Towards Learning Non-recursive LPADs by Transforming Them Into Bayesian Networks

Hendrik Blockeel and Wannes Meert

Katholieke Universiteit Leuven, Department of Computer Science
Celestijnenlaan 200A, 3001 Leuven, Belgium

Abstract. Logic programs with annotated disjunctions, or LPADs, are an elegant knowledge representation formalism that can be used to combine first order logical and probabilistic inference. While LPADs can be written manually, one can also consider the question of how to learn them from data. Methods for learning restricted classes of LPADs have been proposed before, but the problem of learning any kind of LPADs was still open. In this paper, we describe a reduction of non-recursive LPADs with a finite Herbrand universe to Bayesian networks. This reduction makes it possible to learn such LPADs using standard learning techniques for Bayesian networks. Thus the class of learnable LPADs is extended.

1 Introduction

Logic programs with annotated disjunctions, LPADs for short, have been introduced by Vennekens, Verbaeten and Bruynooghe (2004) as a knowledge representation formalism that can be used to combine probabilistic and logic inference. It is a simple and elegant formalism: LPADs are easy to write and interpret, and have formally defined semantics. But while Vennekens et al. define the syntax and semantics of LPADs and illustrate their broad applicability, they do not discuss learning in this framework. Yet learning LPADs is of interest: because LPADs express explicitly the causal structure of stochastic processes, learning them amounts to explicitating this causal structure from observations. It is well known that Bayesian networks, for instance, do not exhibit this property: edges in a Bayesian network do not necessarily indicate a causal relationship.

Riguzzi (2004) proposes an algorithm for learning LPADs. However, only a subclass of LPADs can be learned with his method, and Riguzzi does not discuss whether the subclass is semantically equivalent to the class of all LPADs, that is, whether for each LPAD a semantically equivalent LPAD in the subclass exists.

In this paper we introduce two novel subclasses of LPADs: 1-compliant LPADs and CP-compliant LPADs. We show that each LPAD can be rewritten as a semantically equivalent 1-compliant LPAD; that each 1-compliant LPAD is CP-compliant; and that there is a transformation from (non-recursive, finite-universe) CP-compliant LPADs to Bayesian networks that preserves the LPADs semantics and includes a one-to-one mapping between the LPAD parameters and the Bayesian net parameters.

As a consequence, the many advanced techniques for learning Bayesian networks (both parameters and structure) can be exploited for learning a broad class of LPADs. This class includes LPADs that were not previously learnable.

In the following, we first give the intuitions behind the method (Section 2). Then we treat it more formally: we recall several definitions from the original LPAD paper in Section 3, introduce CP-compliant LPADs in Section 4, and discuss the transformation of LPADs into CP-compliant LPADs in Section 5. In Section 6 we discuss the transformation to Bayesian networks. We briefly discuss learning in Section 7 and wrap up in Section 8.

Note that this work focuses on a specific formal relationship between LPADs and Bayesian networks that can be exploited for learning LPADs. There is no experimental study of learning techniques. Obviously, through the proposed reduction, many well-studied techniques for learning Bayesian networks become available for learning LPADs. Which of these work best in this specific context is a subject for later work.

2 Intuitions

An LPAD can be seen as a set of if-then-rules, where each rule has several possible conclusions, each of which has a certain probability assigned to it. The rule makes exactly one of the conclusions true with the associated probability. For instance,

$$(heads(C) : 0.5) \vee (tails(C) : 0.5) \leftarrow cointoss(C)$$

expresses that if we toss a coin and call the result C, C is heads or tails each with 50% probability.

While in principle the probabilities in the head add up to one, it is possible to write a rule where the sum is less than one; there is then an implicit, anonymous, proposition that is made true with the remaining probability. Thus,

$$(result(C, 6) : 0.167) \leftarrow dieroll(C)$$

expresses that a die roll has 16.7% probability to result in a six. There is then an 83.3% probability to have some other result, but we are not interested in what those other results are.

Multiple rules in an LPAD may lead to the same conclusions. Take, for instance, the following logic program

$$wet \leftarrow gone\_swimming$$
$$wet \leftarrow rain$$

which expresses that if I have gone swimming or if it is raining, my hair is wet.

If we assume that the rules do not hold in 100% of the cases, for instance, I dry my hair immediately after swimming in 30% of the cases, and I use an umbrella when it is raining in 60% of the cases, then we could indicate this as:

$wet : 0.7 \leftarrow gone\_swimming$
$wet : 0.4 \leftarrow rain$

The numbers associated with *wet* now indicate the probability that the event mentioned in the rule body causes my hair to get wet. We will use the convention that when the probability is 1, it may be omitted. The LPAD rule is then equivalent to a regular logic programming clause.

It is part of the semantics of LPADs (see Section 3) that each rule *independently of all other rules* makes one of its head atoms true when triggered. LPADs are therefore particularly suitable for describing models that contain a number of independent stochastic events or causal processes [4]. Consequently, learning LPADs amounts to discovering the causal structure of possibly complex processes.

It may be tempting to interpret the numbers as the conditional probability of *wet* given the body, e.g., $Pr(wet|rain) = 0.4$. However, this would be incorrect. The conditional probability that my hair is wet, given that it is raining, is higher than 0.4, because there is a second possible cause for my hair getting wet, namely the swimming. To compute this conditional probability, we need information on the probability that I have gone swimming. For instance, with

$wet : 0.7 \leftarrow gone\_swimming$
$wet : 0.4 \leftarrow rain$
$gone\_swimming : 0.1.$
$rain : 0.3.$

we can say that $Pr(wet|rain) = 0.4 + 0.1 * 0.7 * 0.6 = 0.442$: the rain causes my hair to get wet with probability 0.4 but there is also a probability of 0.1 that I have gone swimming, and hence a probability of 0.1*0.7*0.6 that my hair is wet not because of the rain but because of swimming.

Thus, for head atoms that occur in multiple rules, the relationship between the mentioned probabilities and conditional probabilities is somewhat complex. But it is not unintuitive. The meaning of the probabilities mentioned in the rules is quite simple: they reflect the probability that the body *causes* the head to become true. This is different from the conditional probability that the head is true given the body, but among the two, *the former is the more natural one to express*. Indeed, the former is local knowledge: we can estimate the probability that *rain* causes *wet* without considering any other possible causes for *wet*. To estimate $Pr(wet|rain)$, we need global knowledge: we need to know all possible causes for *wet*, the probability of them occurring, and how they interact with *rain*.

Arguing that the probabilities in the rules are unintuitive when they do not reflect conditional probabilities, some researchers have proposed to focus on LPADs where all probabilities are conditional probabilities. Let us call such LPADs *CP-compliant* LPADs (CP for conditional probabilities). Riguzzi [3] proves that, when in an LPAD any two rules sharing head atoms have mutually exclusive bodies (i.e., there exist no interpretations for which both bodies are

true; we call such an LPAD *ME-compliant*), then it is CP-compliant. Exploiting this property, Riguzzi proposes a learning algorithm for ME-compliant LPADs.

The above example shows that non-ME-compliant LPADs are by no means far-fetched; they may express knowledge in a straightforward and interpretable way. For that reason, limiting ourselves to ME-compliant LPADs seems undesirable.

In this paper we show that *any* LPAD, ME-compliant or not, can be rewritten into a CP-compliant LPAD. If this LPAD is non-recursive and has a finite Herbrand universe, it can in turn be translated into a Bayesian network[1] in which the conditional probability tables contain exactly the conditional probabilities occurring in the CP-compliant LPAD. As a consequence, if we can learn such Bayesian networks, we have a method for learning any non-recursive LPAD with a finite Herbrand universe.

The conversion from LPAD to CP-compliant LPAD goes as follows. To avoid that head atoms occur in the head of multiple rules, we annotate them with an index that uniquely identifies their rule. To preserve the semantics, we add regular logic programming clauses (or, LPAD rules with one head atom with annotation 1) defining the original predicates in terms of the indexed predicates. Thus, the LPAD

$wet : 0.7 \leftarrow gone\_swimming$
$wet : 0.4 \leftarrow rain$

is turned into the semantically equivalent LPAD

$wet_1 : 0.7 \leftarrow gone\_swimming$
$wet_2 : 0.4 \leftarrow rain$
$wet \leftarrow wet_1$
$wet \leftarrow wet_2$

LPADs resulting from this conversion have the property that head atoms can only be shared by multiple rules if their annotation is 1; we call such LPADs 1-compliant. We prove in this paper that the conversion preserves the LPAD's semantics and that all 1-compliant LPADS are CP-compliant.

A non-recursive CP-compliant LPAD can be converted into a Bayesian network as follows. First, the LPAD is grounded; from now on we refer only to the ground LPAD. The Bayesian network will have one variable for each atom in the ground LPAD, *except the indexed atoms*: since all atoms with index i are mutually exclusive, we can represent them with a single variable C_i instead of introducing a separate boolean variable for each of them. C_i takes on value j if rule i selects its j'th head atom, and 0 if no listed atom is selected. For each C_i we then have one node in the Bayesian network, which has as parent nodes the body atoms of rule i, and its CPD reflects that $C_i = j$ with the corresponding probability if all body literals are true, and $C_i = 0$ otherwise. The original head atoms are also nodes in the Bayesian network: if h occurs as the j'th head atom

[1] A recursive LPAD would give rise to cycles in the Bayesian network, which are not allowed. An infinite Herbrand universe would give rise to an infinite Bayesian net.

of rule i, then h has C_i as a parent and its CPD reflects that h is true whenever $C_i = j$.

The CP-compliant LPAD just mentioned is shown as a Bayesian network in Fig. 1.

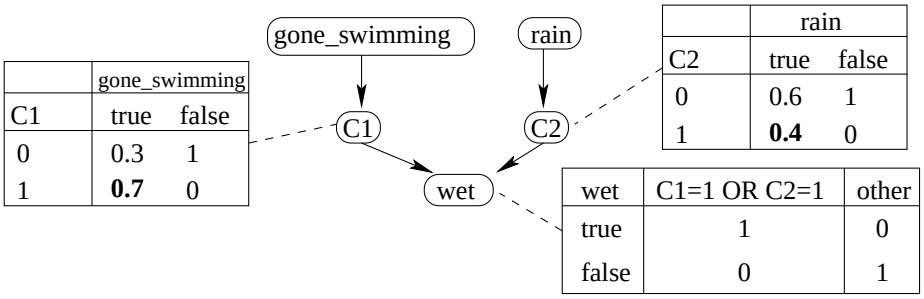

Fig. 1. A Bayesian network corresponding to our example CP-compliant LPAD. The probabilities in boldface are those occurring explicitly in the LPAD.

In the remainder of this text, we define the transformation more formally, show that it always yields CP-compliant (though not ME-compliant) LPADs, and show how to construct from a CP-compliant LPAD a semantically equivalent Bayesian network. Learning (non-recursive, finite-universe) LPADs will thus be formally reduced to learning Bayesian networks, for which many techniques already exist.

3 LPADs: syntax and semantics

We start with a brief overview of the syntax and semantics of LPADs, as given by Vennekens et al. [5]. We assume familiarity with standard logic programming terminology (atoms, literals, Herbrand universe, Herbrand base, variable substitutions, groundings, etc.). See Lloyd [1] for an introduction on this.

An LPAD consists of a set of rules of the following form:

$$(h_1 : \alpha_1) \vee (h_2 : \alpha_2) \vee \ldots \vee (h_n : \alpha_n) \leftarrow b_1, b_2, \ldots, b_m.$$

with h_i atoms and b_i literals in the logical sense, all $\alpha_i \in [0, 1]$, and $\sum_{i=1}^{n} \alpha_i = 1$. We call the set of all $(h_i : \alpha_i)$ the head of the rule c, denoted $head(c)$, and the set of all b_i the body, denoted $body(c)$. If the head contains only one atom $h : 1$, we may write it as h.

The semantics of an LPAD is defined using its grounding. We denote the set of all ground LPADs with $\mathcal{P}_\mathcal{G}$. Given an LPAD P, $\mathcal{I}_P$ is the set of all Herbrand interpretations of P. The Herbrand base of P is denoted H_P. The semantics of an LPAD is defined as a probability distribution on $\mathcal{I}_P$, as follows.

Definition 1. *Let $P \in \mathcal{P}_\mathcal{G}$. An admissible probability distribution π on $\mathcal{I}_P$ is a mapping from $\mathcal{I}_P$ to $[0,1]$ such that $\sum_{I \in \mathcal{I}_P} \pi(I) = 1$.*

Definition 2. *Let $P \in \mathcal{P}_\mathcal{G}$. A selection σ is a function that selects one pair $(h : \alpha)$ from each rule of P, i.e., $\sigma : P \to (H_P \times [0,1])$ such that for each $c \in P$, $\sigma(c) \in head(c)$. With $\sigma(c) = h : \alpha$, we also write $\sigma_{atom} = h$ and $\sigma_{prob} = \alpha$. The set of all selections σ is denoted by $\mathcal{S}_P$.*

Definition 3. *Let $P \in \mathcal{P}_\mathcal{G}$ and $\sigma \in \mathcal{S}_P$. The instance P_σ chosen by σ is defined as $P_\sigma = \{\sigma_{atom}(c) \leftarrow body(c) | c \in P\}$.*

Definition 4. *Let $P \in \mathcal{P}_\mathcal{G}$ and $\sigma \in \mathcal{S}_P$. The probability of σ is*

$$C_\sigma = \prod_{c \in P} \sigma_{prob}(c).$$

This definition of the probability of a selection implies that the selection of a head atom in one rule is stochastically independent from the selection of head atoms in all other rules.

The following definition defines the LPADs to which we can give meaning:

Definition 5. *An LPAD P is sound iff for each $\sigma \in \mathcal{S}_P$, the well founded model of P_σ, denoted $WFM(P_\sigma)$, is two-valued.*

Since we only consider two-valued well-founded models, we can represent the well-founded model as a single interpretation. We will use this convention in the remainder of the paper.

Given an interpretation I, we denote the set of all $\sigma \in \mathcal{S}_P$ for which $WFM(P_\sigma) = I$ as $\mathcal{S}_P^I$. The semantics of a sound LPAD is then defined as follows.

Definition 6. *Let $P \in \mathcal{P}_\mathcal{G}$ be a sound LPAD. For each of its interpretations $I \in \mathcal{I}_P$, the probability $\pi_P^*(I)$ assigned by P to I is the sum of the probabilities of all selections that lead to I, i.e.,*

$$\pi_P^*(I) = \sum_{\sigma \in \mathcal{S}_P^I} C_\sigma.$$

Vennekens et al. [5] prove that if P is a sound LPAD in $\mathcal{P}_\mathcal{G}$, then π_P^* is an admissible probability distribution over $\mathcal{I}_P$. This defines the semantics of any sound LPAD.

We next recall the definition of the probability of a logic formula, again from Vennekens et al.

Definition 7. *For any logic formula ϕ, the set of Herbrand models of ϕ is denoted and defined as*

$$\mathcal{I}_P^\phi = \{I \in \mathcal{I}_P | I \models \phi\}.$$

Definition 8. *Let P be a sound LPAD in $\mathcal{P}_\mathcal{G}$. The probability of ϕ according to P, denoted $\pi^*_P(\phi)$, is defined as*

$$\pi^*_P(\phi) = \sum_{I \in \mathcal{I}^\phi_P} \pi^*_P(I).$$

We add the notion of conditional probability:

Definition 9. *Let P be a sound LPAD in $\mathcal{P}_\mathcal{G}$. The conditional probability of ϕ given ψ, according to P, is denoted and defined as*

$$\pi^*_P(\phi|\psi) = \pi^*_P(\phi \wedge \psi)/\pi^*_P(\psi)$$

if $\pi^*_P(\psi) > 0$ (and undefined otherwise).

When P is clear from the context, we will often denote $\pi^*_P(\phi)$ as $Pr(\phi)$ and $\pi^*_P(\phi|\psi)$ as $Pr(\phi|\psi)$.

As said before, one should take care not to interpret α_i $Pr(h_i|B)$, the conditional probability of h_i given the body B. However, LPADs generally do have the property that $Pr(h_i|B) \geq \alpha_i$ [5].

4 CP-compliant LPADs

Riguzzi [3] presents a learning algorithm for a subclass of LPADs, which we will refer to as ME-compliant LPADs. The definition is as follows.

Definition 10 (ME-compliant LPADs). *An ME-compliant LPAD is an LPAD in which for each two rules $H_1 \leftarrow B_1$ and $H_2 \leftarrow B_2$ it holds that (a) H_1 and H_2 do not share any atoms, or (b) B_1 and B_2 are mutually exclusive.*

Under these conditions, it holds for each annotated head atom $h_i : \alpha_i$ in a rule that $Pr(h_i|B) = \alpha_i$ with B the body of the rule. We call LPADs fulfilling this property CP-compliant.

Definition 11 (CP-compliant LPADs). *A CP-compliant LPAD is an LPAD in which for each rule $H \leftarrow B$ it holds that $\forall(h_i : \alpha_i) \in H : Pr(h_i|B) = \alpha_i$.*

CP-compliance is important because conditional probabilities can easily be estimated from data: if an LPAD is CP-compliant, then its parameters can be estimated equally easily. This property is exploited by Riguzzi to learn the α_i parameters of ME-compliant LPADs from data.

Now we introduce a different subclass of LPADs, which (as we shall prove) also has the property that all parameters to be estimated are conditional probabilities. We call this subclass 1-compliant LPADs. The name refers to the property that each head atom either occurs only in the head of a single rule, or its annotation is 1 in all the heads where it occurs.

Definition 12 (1-compliant LPADs). *A 1-compliant LPAD is an LPAD in which for each atom h that occurs in the head of a rule, it holds that either h occurs in only one rule (i.e., it cannot be unified with any atom in the head of any other rule), or it always occurs with an annotation of 1.*

1-compliant LPADs are of the same form as the transformed LPAD we mentioned in our intuitive treatment.

In the syntactic sense, our CP-compliant LPADs are neither a subclass nor a superclass of Riguzzi's ME-compliant LPADs. Riguzzi allows several rules to share head atoms as long as their bodies are mutually exclusive, which is generally not allowed in CP-compliant LPADs. On the other hand, we allow rules to share head atoms even if their bodies are not mutually exclusive, as long as the probabilities of these atoms are one.

Riguzzi shows that ME-compliant LPADs are CP-compliant. We now show that 1-compliant LPADs are CP-compliant.

Theorem 1. *In a 1-compliant LPAD, for each rule of the form $h_1 : \alpha_1 \vee \ldots \vee h_n : \alpha_n \leftarrow B$, $Pr(h_i|B) = \alpha_i$. That is, each α_i can be interpreted as the conditional probability that its atom is true, given that the rule body is true.*

Proof. According to the definition of a 1-compliant LPAD, for each $h_i : \alpha_i$ in a rule head with body B, it holds that either (a) h_i does not occur in any other rule heads, or (b) $\alpha_i = 1$.
Case (a): $Pr(h_i|B) = \alpha_i$ follows from Riguzzi's proof of Theorem 1 [3]. While the theorem states that for any rule, $\alpha_i = Pr(h_i|B)$ if all the rules (in the whole LPAD) sharing head atoms have mutually exclusive bodies, the proof in fact just exploits the mutual exclusion property for the rule for which the equality is proven. Case (a) implies this property.
Case (b): We know from the definition of LPADs and their semantics that $\alpha_i \leq Pr(h_i|B) \leq 1$. If $\alpha_i = 1$, this implies $Pr(h_i|B) = \alpha_i$.

5 Transforming LPADs to 1-compliant LPADs

An algorithm for transforming LPADs into 1-compliant LPADs is shown in Table 1. The algorithm just adds an index i to the predicate names of all the head atoms of each rule c_i, and adds rules stating that the original (unindexed) version of the atom must be true if its indexed version is true.

Example 1. Consider the following LPAD:

$(a : 0.5) \vee (b : 0.5) \leftarrow c$
$(b : 0.2) \vee (c : 0.8) \leftarrow d$

The i-th rule is transformed by just adding an index i to each atom in the head:

$(a_1 : 0.5) \vee (b_1 : 0.5) \leftarrow c$
$(b_2 : 0.2) \vee (c_2 : 0.8) \leftarrow d$

function Transform(P: LPAD) **returns** CP-compliant LPAD
$P' := \emptyset$
for each rule $(h_{i1} : \alpha_{i1} \vee \ldots \vee h_{in_i} : \alpha_{in_i} \leftarrow B_i) \in P$:
let h'_{ij} be h_{ij} with its predicate name p changed into p_i
$P' := P' \cup \{h'_{ij} : \alpha_{i1} \vee \ldots \vee h'_{in_i} : \alpha_{in_i} \leftarrow B_i\}$
$P' := P' \cup \bigcup_{j=1}^{n_i} \{h_{ij} \leftarrow h'_{ij}\}$
return P'

Table 1. Algorithm for transforming LPADs into CP-compliant LPADs

and the following rules are added:

$a \leftarrow a_1$
$b \leftarrow b_1$
$b \leftarrow b_2$
$c \leftarrow c_2$

Theorem 2. *The transformation yields a 1-compliant LPAD.*

Proof. The resulting program consists of two types of rules: rules with indexed atoms in the head (type 1 rules) and rules with original atoms in the head (type 2 rules). A type 1 rule cannot share a head atom with any other rule: not with a type 2 rule because it has only indexed atoms in the head (and type 2 rules contain only original atoms), and not with other type 1 rules because the indexes differ. Only type 2 rules can therefore share head atoms, but they all have a single head atom with annotation 1. Consequently, the conditions for 1-compliance are fulfilled.

We now prove that the transformation preserves the semantics of the LPAD, in the sense that any logic formula ϕ defined over the original LPAD has the same probability according to the transformed LPAD.

Theorem 3. *Let $P \in \mathcal{P}_\mathcal{G}$ be a sound LPAD, and let P' be the transformed version of P. For each formula ϕ defined over P,*

$$\pi^*_P(\phi) = \pi^*_{P'}(\phi)$$

Proof. First, we expand the left hand side of the equation:

$$\pi^*_P(\phi) = \sum_{I \in \mathcal{I}^\phi_P} \sum_{\sigma \in \mathcal{S}^I_P} \prod_{r \in P} \sigma_{prob}(r)$$

Define $\mathcal{S}^\phi_P$ as the set of all selections σ for which $WFM(P_\sigma) \models \phi$; that is, $\mathcal{S}^\phi_P = \bigcup\{\mathcal{S}^I_P | I \models \phi\}$. We can then shorten the above expression to

$$\pi^*_P(\phi) = \sum_{\sigma \in \mathcal{S}^\phi_P} \prod_{r \in P} \sigma_{prob}(r)$$

Similarly, for the right hand side we have

$$\pi^*_{P'}(\phi) = \sum_{\sigma \in \mathcal{S}^\phi_{P'}} \prod_{r \in P'} \sigma_{prob}(r)$$

So we need to prove

$$\sum_{\sigma \in \mathcal{S}^\phi_{P}} \prod_{r \in P} \sigma_{prob}(r) = \sum_{\sigma \in \mathcal{S}^\phi_{P'}} \prod_{r \in P'} \sigma_{prob}(r) \tag{1}$$

We can define a one-to-one correspondence between $\mathcal{S}_P$ and $\mathcal{S}_{P'}$ as follows. Let $\sigma \in \mathcal{S}_P$ and $\sigma' \in \mathcal{S}_{P'}$ be such that

$$\sigma(P) = \{(h_{1s_1} : \alpha_{1s_1}), \ldots, (h_{ms_m} : \alpha_{ms_m})\}$$
$$\sigma'(P') = \left\{ (h'_{1s_1} : \alpha_{1s_1}), \ldots, (h'_{ms_m} : \alpha_{ms_m}), \bigcup_{i=1}^{m} \bigcup_{j=1}^{n_i} \{h_{ij}\} \right\}$$

where m is the number of rules, n_i is the number of head atoms in rule i and $s_i \in [1, n_i]$. This is clearly a one-to-one correspondence because both σ and σ' map one-to-one to a vector $(s_1, s_2, \ldots, s_m)$. Fig. 2 illustrates this correspondence more graphically.

To prove Equation 1, it suffices to show that (1) this one-to-one-correspondence carries over to $\mathcal{S}^\phi_P$ and $\mathcal{S}^\phi_{P'}$, that is, $\sigma \in \mathcal{S}^\phi_P \Leftrightarrow \sigma' \in \mathcal{S}^\phi_{P'}$, and (2) the probabilities associated with corresponding selections are the same.

(1) We need to prove $\sigma \in \mathcal{S}^\phi_P \Rightarrow \sigma' \in \mathcal{S}^\phi_{P'}$ and $\sigma \notin \mathcal{S}^\phi_P \Rightarrow \sigma' \notin \mathcal{S}^\phi_{P'}$. But since $\sigma \notin \mathcal{S}^\phi_P$ is equivalent to $\sigma \in \mathcal{S}^{\neg\phi}_P$, it suffices to prove that the first implication holds for any formula ϕ.

So assume $\sigma \in \mathcal{S}^\phi_P$; this implies there is an I such that $\sigma \in \mathcal{S}^I_P$ and $I \models \phi$. Now define $I' = I \cup \left\{h'_{ij} | h_{ij} \in \sigma_{atom}(P)\right\}$. We will prove that (1a) $\sigma' \in S_{P'}(I')$ and (1b) $I' \models \phi$; from this follows $\sigma' \in \mathcal{S}^\phi_{P'}$.

(1a) σ' is defined in such a way that P_σ contains $h_{ij} \leftarrow B_i$ if and only if $P'_{\sigma'}$ contains the clauses $\left\{h_{ij} \leftarrow h'_{ij}, h'_{ij} \leftarrow B_i\right\}$. Thus, whenever h_{ij} can be derived in P_σ, it can be derived in $P'_{\sigma'}$, and vice versa. In addition, whenever h_{ij} can be derived in P_σ, h'_{ij} can be derived in $P'_{\sigma'}$. This proves that $WFM(P_\sigma) = I$ if and only if $WFM(P'_{\sigma'}) = I'$, in other words, $\sigma \in \mathcal{S}^I_P \Leftrightarrow \sigma' \in S_{P'}(I')$.

(1b) The formula ϕ refers only to original (non-indexed) predicates. Since I', restricted to non-indexed predicates, equals I, $I' \models \phi$ if and only if $I \models \phi$.

This proves the one-to-one correspondence between $\mathcal{S}^\phi_P$ and $\mathcal{S}^\phi_{P'}$.

(2) If we multiply all the σ_{prob} as defined by σ and σ' we get:

$$\prod_{r \in P} \sigma_{prob}(r) = \alpha_{1s_1} \ldots \alpha_{ms_m}$$
$$\prod_{r \in P'} \sigma'_{prob}(r) = \alpha_{1s_1} \ldots \alpha_{ms_m} \cdot \underbrace{1 \ldots 1}_{\sum_{i=1}^{m} n_i \text{times}}$$

$\underline{(a:0.3)} \vee (b:0.4) \vee (c:0.3) \leftarrow B_1$	$\underline{(a_1:0.3)} \vee (b_1:0.4) \vee (c_1:0.3) \leftarrow B_1$
$\underline{(a:0.1)} \vee (d:0.5) \vee \underline{(e:0.1)} \leftarrow B_2$	$\underline{(a_2:0.1)} \vee (d_2:0.5) \vee \underline{(e_2:0.1)} \leftarrow B_2$
$(d:0.5) \vee \underline{(f:0.5)} \leftarrow B_3$	$(d_3:0.5) \vee \underline{(f_3:0.5)} \leftarrow B_3$
	$\underline{a} \leftarrow a_1$
	$\underline{a} \leftarrow a_2$
	$\underline{b} \leftarrow b_1$
	$\underline{c} \leftarrow c_1$
	$\underline{d} \leftarrow d_2$
	$\underline{d} \leftarrow d_3$
	$\underline{e} \leftarrow e_2$
	$\underline{f} \leftarrow f_3$

Fig. 2. An illustration of the one-to-one correspondence between selections in P and in P'. For the program P to the left, the 1-compliant version P' is shown to the right. For each selection σ for P there is precisely one selection σ' for P' according to the defined correspondence, and they have the properties that the probabilities of σ and σ' are equal and $\mathrm{WFM}(P'_{\sigma'})$ restricted to H_P equals $\mathrm{WFM}(P_\sigma)$.

Thus the probability of σ and σ' is the same. This concludes the proof.

Corollary 1. *Any LPAD can be transformed into a 1-compliant, and therefore CP-compliant, LPAD.*

Corollary 2. *ME-compliant LPADs can be transformed into 1-compliant LPADs.*

6 A reduction to Bayesian networks

We now turn to the transformation of a 1-compliant LPAD into a Bayesian network. From here on we assume LPADs to be non-recursive and have a finite Herbrand universe.

First, the LPAD needs to be grounded. From here one, when we refer to the LPAD, we mean the ground LPAD.

The nodes in the Bayesian network are the original LPAD atoms (which become boolean variables) as well as so-called choice variables C_i (which are included instead of the indexed atoms, as explained before). Each C_i takes integer values in the interval $[0, n_i]$ where n_i is the number of head atoms of LPAD rule i. When $C_i = j$, this means the j'th atom of rule i has been selected. $C_i = 0$ will be used to denote that no listed head atom was selected, which may be either because the rule body is false, or because an anonymous atom was selected.

The conditional probability distribution (CPD) of each C_i has a very simple structure, represented by the following function:

$P(C_i = j|\text{all parents true}) = \alpha_j$, for all $j > 0$
$P(C_i = 0|\text{all parents true}) = 1 - \sum_j \alpha_j$
$P(C_i = 0|\text{not all parents true}) = 1$
$P(C_i = j|\text{not all parents true}) = 0$, for all $j > 0$

The CPD of a non-choice variable always expresses a logical or, since the second part of our LPAD is essentially a standard logic program. For instance, in the example, b is true if $C_1 = 2$ or $C_2 = 1$. This can be represented with a simple CPD function:

$P(h_i = true|\text{all parents false}) = 0$
$P(h_i = true|\text{not all parents false}) = 1$

(the probabilities for h_i=*false* are the complements).

Example 2. For the LPAD of Example 1, which in 1-compliant form became

$a_1 : 0.5 \vee b_1 : 0.5 \leftarrow c$
$b_2 : 0.2 \vee c_2 : 0.8 \leftarrow d$
$a \leftarrow a_1$
$b \leftarrow b_1$
$b \leftarrow b_2$
$c \leftarrow c_2$,

the corresponding Bayesian net is shown in Fig.3.

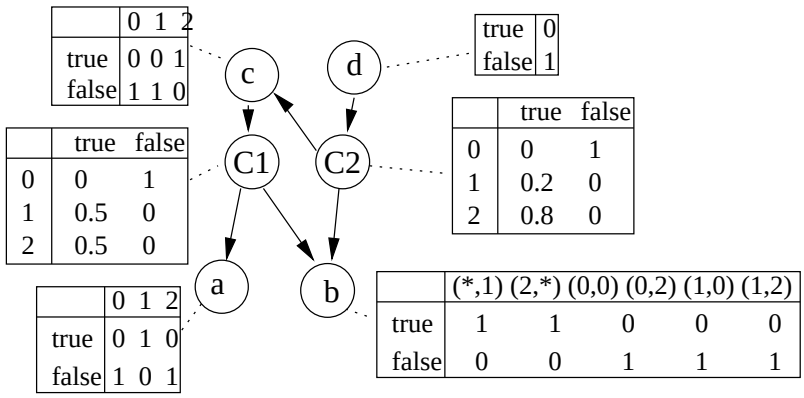

Fig. 3. Bayesian net corresponding to the example LPAD

The algorithm to produce a Bayesian net from a 1-compliant LPAD is shown in Table 2.

Thus, given an LPAD with certain parameters, it can be transformed into a Bayesian network with a specific structure, consisting of two kinds of variables: atom variables and choice variables. The CPD's for the atom variables have a fixed structure that is independent of the probabilities in the LPAD; they always express a logical or. The CPD's for the choice variables have a fixed structure and contain only 0's, 1's, and the LPAD probabilities α_{ij}.

function BN(P: 1-compliant LPAD) **returns** a Bayesian net
$N := \emptyset$ // *nodes of the BN*
$E := \emptyset$ // *edges of the BN*
for each rule $(h'_{i1} : \alpha_{i1} \vee \ldots \vee h'_{in_i} : \alpha_{in_i} \leftarrow b_{i1}, \ldots, b_{im_i}) \in P$:
$N := N \cup \{C_i\} \cup \bigcup_j \{b_{ij}\}$
$E := E \cup \bigcup_j \{(b_{ij}, C_i)\}$
associate with C_i a CPD as follows:
$P(C_i = j|\text{all parents true}) = \alpha_{ij}$, for all $j > 0$
$P(C_i = 0|\text{all parents true}) = 1 - \sum_j \alpha_{ij}$
$P(C_i = 0|\text{not all parents true}) = 1$
$P(C_i = j|\text{not all parents true}) = 0$, for all $j > 0$
for each clause $(h_{ij} \leftarrow h'_{ij}) \in P$:
$N := N \cup \{h_{ij}\}$
$E := E \cup \bigcup_j \{(C_i, h_{ij})\}$
for each $l \in N$ that is not a C_i:
associate with l a CPD as follows:
$C := \bigvee_{ij:(l \leftarrow h'_{ij}) \in P} C_i = j$
$P(l = true|C) = 1$
$P(l = false|C) = 0$
$P(l = true|\neg C) = 0$
$P(l = false|\neg C) = 1$
return (N, E, CPD)

Table 2. Algorithm for transforming non-recursive CP-compliant LPADs into Bayesian networks. By convention, h_{ij} refers to original atoms in this description, and h'_{ij} to indexed atoms.

7 Perspectives on learning LPADs

The previous sections contain the main contribution of the paper. In the following we briefly discuss the perspectives on learning LPADs that our results entail.

7.1 Learning the parameters of an LPAD

Given an LPAD with unknown values for the probabilities, we can learn these probabilities using the following approach: (1) ground the LPAD; (2) reduce the ground LPAD to a Bayesian net, as outlined above; (3) learn the CPDs of the Bayesian net, taking into account constraints on these CPDs; (4) map these CPDs onto the LPAD probabilities.

Our previous discussion leaves only step 3 to be discussed. First, note that the Bayesian net contains unobserved variables: indeed, none of the C_i are observed in the data. There are well-known procedures for parameter estimation in such Bayesian nets, e.g., EM-MAP [2].

Second, we want the CPDs that are being learned to have a specific structure. As explained before, the CPDs of the atom variables are fixed (they express a logical "or"). In the CPDs of the choice variables C_i, only one row of values is

to be learned, the other rows contain 0 and 1. To learn these CPDs, it suffices to initialize the 0 and 1 elements with their right value: the Bayesian update rule can only update values strictly between 0 and 1.

Finally, note that when grounding an LPAD, a single rule is typically transformed in a set of rules, all with the same α parameters. These α's occur in multiple places in the Bayesian net. When learning the parameters of the Bayesian net, we need to take into account that all the parameters corresponding to one α must have the same value. This can be done by forcing these parameters to be their average after each iteration of the EM algorithm.

7.2 Learning both structure and parameters of an LPAD

We consider the following task: given a set of predicates, learn an LPAD that may contain any of these predicates. This involves a search over possible LPAD structures, which can be done either in LPAD space or in Bayesian net (BN) space. Typically, a greedy algorithm such as the ones below would be used:

LPAD := $\emptyset$
while LPAD is not good enough:
    S := refinements(LPAD)
    LPAD := argmax$_{L \in S}$ eval(L)
return LPAD

BN := initial Bayesian net
while BN is not good enough:
    S := refinements(BN)
    BN := argmax$_{L \in S}$ eval(L)
return LPAD(BN)

In LPAD space, a refinement operator similar to Riguzzi's [3] could be used for the search, though alternatives can be explored. In BN space, since Bayesian nets corresponding to LPADs are a subset of all possible Bayesian nets, one needs to ensure that the Bayesian net returned maps to a valid LPAD. To this aim, the refinement operator can be redefined so that only LPAD-compatible Bayesian nets are generated.

The *eval* function is typically based on the likelihood of the data given the candidate model. Computing this involves (in LPAD space) grounding the candidate LPAD and transforming it into a Bayesian network; then (in both cases) estimating the parameters of the network and computing the likelihood.

Some experiments with a first implementation of the above described parameter and structure learning approaches, which space restrictions prevent us from detailing here, indicate that at least learning small LPADs such as the ones shown in this paper is quite feasible, and suggest that the repeated conversion of LPADs into Bayesian nets may make the LPAD search much slower than the BN search. We plan a more detailed experimental comparison of the approaches as future work.

8 Conclusions

The main contribution of this paper is the definition of a reduction of non-recursive finite-universe LPADs to Bayesian networks. This reduction is defined

in two steps: in a first step, an LPAD is transformed into a so-called 1-compliant LPAD. In a second step, the latter is transformed into a Bayesian network.

The reduction provides some novel insights regarding the meaning of the α parameters in an LPAD. In particular, we have shown that while the probabilities in LPADs cannot generally be interpreted as conditional probabilities within the universe of atoms occurring in the LPAD, it is always possible to transform the LPAD into another LPAD where this property does hold.

The reduction also offers several perspectives with respect to learning LPADs. First, the only existing methods for learning LPADs, up till now, handled only a restricted type of LPADs, so-called ME-compliant LPADs. Using the reduction proposed here, that restriction is lifted. Second, the reduction makes the extensive expertise on learning Bayesian networks available for learning LPADs.

In future work we intend to have a closer look at the many existing methods for learning Bayesian networks, and evaluate how suitable these approaches are from the point of view of learning LPADs (i.e., how well they work in the presence of constraints on the parameters that are being learned).

Our reduction still leaves open the question of how to learn recursive LPADs. One approach may be based on translating the LPAD to an undirected graphical model, removing the problem that cycles are not allowed. Another approach would be to abandon the reductionist approach and develop an algorithm specifically for learning LPADs. Both approaches will be investigated in the future.

9 Acknowledgements

H.B. is a post-doctoral fellow of the Fund for Scientific Research of Flanders (FWO-Vlaanderen). Research supported by the Flemish Institute for the Promotion of Innovation by Science and Technology in Flanders (IWT-Vlaanderen) and project GOA/2003/08 (B0516) on Inductive Knowledge Bases. The authors thank Kristian Kersting, Jan Ramon, Joost Vennekens and Maurice Bruynooghe for their valuable inputs.

References

1. J.W. Lloyd. *Foundations of Logic Programming*. Springer-Verlag, 2nd edition, 1987.
2. R. Neapolitan. *Learning Bayesian Networks*. Prentice Hall, Upper Saddle River, NJ, USA, 2003.
3. F. Riguzzi. Learning logic programs with annotated disjunctions. In *Proceedings of the 14th International Conference on Inductive Logic Programming*, volume 3194 of *Lecture Notes in Artificial Intelligence*, pages 270–287. Springer-Verlag, 2004.
4. J. Vennekens, M. Denecker, and M. Bruynooghe. Extending the role of causality in probabilistic modeling. In *Proceedings of the 11th International Workshop on Non-monotonic Reasoning*, pages 183–190, 2006.
5. Joost Vennekens, Sofie Verbaeten, and Maurice Bruynooghe. Logic programs with annotated disjunctions. In *Logic Programming, 20th International Conference, ICLP 2004, Proceedings*, volume 3132 of *Lecture Notes in Computer Science*, pages 431–445. Springer, 2004.