Using the Probabilistic Logic Programming Language P-log for Causal and Counterfactual Reasoning and Non-naive Conditioning

Chitta Baral and **Matt Hunsaker**
Department of Computer Science and Engineering
Arizona State University
Tempe, Arizona 85281
$\{$*chitta,hunsaker*$\}$@*asu.edu*

Abstract

P-log is a probabilistic logic programming language, which combines both logic programming style knowledge representation and probabilistic reasoning. In earlier papers various advantages of P-log have been discussed. In this paper we further elaborate on the KR prowess of P-log by showing that: (i) it can be used for causal and counterfactual reasoning and (ii) it provides an elaboration tolerant way for non-naive conditioning.

1 Introduction and Motivation

In recent years there has been a lot of progress in logical knowledge representation and reasoning as well as probabilistic knowledge representation and reasoning. There have been some attempts at developing formalisms that allow both logical as well as probabilistic knowledge representation.

One such approach is Pearl's probabilistic causal models [Pearl, 2000]. While Pearl's formalism is an improvement over traditional propositional probability representations such as Bayes' nets, there are still the limitations that are associated with propositional representation as well as classical (and hence monotonic) reasoning. In recent years approaches to overcome both have been proposed. Proposed approaches to overcome the limitations of propositional representation include probabilistic relational models [Koller, 1999; Getoor *et al.*, 2001], various probabilistic first-order logics [Nilsson, 1986; Bacchus, 1990; Bacchus *et al.*, 1996; Halpern, 1990; 2003; Pasula and Russell, 2001; Poole, 1993], and approaches based on assigning weights to first-order formulas [Paskin, 2002; Richardson and Domingos, 2006]. There have been also many formalisms that integrate logic programming with probability such as Poole's Probabilistic Horn Abduction (PHA) [Poole, 1993], his Independent Choice Logic (ICL) [Poole, 1997; 2000], the LPAD formalism [Vennekens *et al.*, 2004], Bayesian logic programming [Kersting and Raedt, 2000], ALTERID [Breese, 1990; Wellman *et al.*, 1992], probabilistic knowledge bases [Ngo and Haddawy, 1997], stochastic logic programs [Muggleton, 1995; Cussens, 1999], probabilistic constraint logic programs [Riezler, 1998; Santos Costa *et al.*, 2003], interval based probabilistic logic programming [Ng and Subrahmanian, 1992; 1994; Dekhtyar and Dekhtyar, 2004; Lukasiewicz, 1998], dynamically ordered probabilistic choice logic programming [De Vos and Vermeir, 2000], and the recent langauge P-log [Baral *et al.*, 2004; 2005]. Our focus on this paper is on P-log.

Earlier it was shown that P-log has many advantages over the other languages. In particular, P-log allows a wide variety of updates compared to the other approaches; it allows explicit specification of background knowledge; and it allows logical reasoning to dynamically decide on the range of values that a random variable can take. In this paper we show how P-log can also do causal and counterfactual reasoning of the kind in Pearl's probabilistic causal models. We give an encoding of probabilistic causal models in P-log and state the correspondence. Our correspondence shows that while the computation of causal and counterfactuals in PCMs is algorithmic in nature, in the encoded P-log it is automatically captured by the semantics of the language. We then show that P-log's ability to allow logical reasoning to dynamically decide on the range of values that a random variable can take leads to a elaboration tolerant way to perform non-naive conditioning [Halpern, 2003]. We illustrate this with respect to Halpern's second-ace puzzle example.[1]

2 Background

In the next two subsections, we will briefly review the syntax of a simplified version of P-log [Baral *et al.*, 2004] and the syntax and semantics of probabilistic causal models [Pearl, 2000].

2.1 P-log lite: an overview

For simplicity, we will define a subset of the probabilistic logic programming language P-log, which we will call *P-log lite*. We use logic programming terminology of terms, atoms and literals. A P-log lite program Π consists of (i) a set of boolean attributes $\{p, \neg p \; : \; p$ is an atom $\}$, (ii) a *regular part*, (iii) a set of *random selection rules*, (iv) a *probabilistic information* part, and (v) a set of *observations* and *actions*. In the following by $\bar{t}$ we denote a vector of parameters consisting of terms, and by $\widetilde{\neg p}$ we refer to p and by $\tilde{p}$ we refer to $\neg p$.

[1]Earlier work on P-log [Baral *et al.*, 2004] hinted at this with respect to the Monty Hall example. However no general methodologies for "non-naive" conditioning were proposed.

(ii) Regular part: The regular part of a P-log lite program consists of a collection of logic programming rules (without disjunction and under the stable model semantics).

(iii) Random Selection: A *random selection* is a rule of the form

$$random(a(\bar{t})). \qquad (1)$$

Statement (1) says that one of $a(\bar{t})$ or $\neg a(\bar{t})$ is selected at random. In P-log with non-boolean attributes the random statement is more general and is of the form: $random(Attr : \{X \ : \ p(X)\})$.

(iv) Probabilistic Information: Information about probabilities of random attributes taking particular values is given by *probability atoms* (or simply *pr-atoms*) which have the form:

$$pr(l) = v. \qquad (2)$$

where l is $a(\bar{t})$ or $\neg a(\bar{t})$, and $pr(a(\bar{t})) = v$ would cause $a(\bar{t})$ with probability v.

(v) Observations and actions: Observations and actions are statements of the respective forms: $obs(l)$ and $do(l)$.

Observations are used to record the outcomes of random events, i.e., random attributes and attributes dependent on them. The statement $do(a(\bar{t}))$ indicates that $a(\bar{t})$ is made true as a result of a deliberate (non-random) action.

Semantics of P-log lite:

The semantics of a P-log lite program Π is given by a collection of the possible sets of beliefs of a rational agent associated with Π, together with their probabilities.

We start with translating the logical part of a P-log lite program Π to an Answer Set Prolog program $\tau(\Pi)$ as described below.

(a) $\tau(\Pi)$ contains the regular part of Π.

(b) For each attribute atom $a(\bar{t})$, $\tau(\Pi)$ has the rules:

$intervene(a(\bar{t})) \leftarrow do(a(\bar{t}))$.
$intervene(a(\bar{t})) \leftarrow do(\neg a(\bar{t}))$.

(c) For each $random(a(\bar{t}))$ in Π, $\tau(\Pi)$ has the Smodels rule:

$1\{a(\bar{t}), \neg a(\bar{t})\}1 \leftarrow not\ intervene(a(\bar{t}))$.

The left hand side of the above rule means that one of $a(\bar{t})$ and $\neg a(\bar{t})$ must be in the answer set.

(d) $\tau(\Pi)$ has all the actions and observations of Π.

(e) For all literals l, $\tau(\Pi)$ has the rules:

$\leftarrow obs(l), not\ l$.
$l \leftarrow do(l)$.

An answer set of $\tau(\Pi)$ is called a **possible world** of Π. The set of all possible worlds of Π is denoted by $\Omega(\Pi)$.

Let W be a possible world of a P-log lite program Π. If Π contains a random selection rule of the form $random(a(\bar{t}))$, then we say that both $a(\bar{t})$ and $\neg a(\bar{t})$ are *possible* in W with respect to Π. For every $W \in \Omega(\Pi)$ and every literal l *possible* in W we now define the corresponding probability $P(W, l)$.

For each l *possible* in W:

If $l \in W$, Π has a *pr-atom* $pr(l) = v$ and W does not contain $intervene(l)$ then $P(W, l) = v$, and $P(W, \tilde{l}) = 1 - v$.

If for any attribute literal l, we do not have pr-atoms for either l or $\tilde{l}$, $l \in W$ and W does not contain $intervene(l)$ then $P(W, l) = 0.5$, and $P(W, \tilde{l}) = 0.5$.

Now we are ready to define the measure, μ_Π, induced by the P-log lite program Π.

Definition 1. *(Measure)* (i) The *unnormalized probability*, $\hat{\mu}_\Pi(W)$, of a possible world W *induced by* Π is $\hat{\mu}_\Pi(W) = \prod_{l \in W} P(W, l)$, where the product is taken over literals for which $P(W, l)$ is defined.

(ii) The *measure* (or the normalized probability), $\mu_\Pi(W)$, of a possible world W *induced by* Π is the unnormalized probability of W divided by the sum of the unnormalized probabilities of all possible worlds of Π, i.e., $\mu_\Pi(W) = \frac{\hat{\mu}_\Pi(W)}{\sum_{W_i \in \Omega} \hat{\mu}_\Pi(W_i)}$

When the program Π is clear from the context we may simply write $\hat{\mu}$ and μ instead of $\hat{\mu}_\Pi$ and μ_Π respectively. $\square$

The truth and falsity of propositional formulas with respect to possible worlds are defined in the standard way. A formula A is true in W is denoted by $W \vdash A$.

Definition 2. *(Probability)*
The *probability* with respect to program Π of a formula A, $P_\Pi(A)$, is the sum of the measures of the possible worlds of Π in which A is true, i.e. $P_\Pi(A) = \sum_{W \vdash A} \mu_\Pi(W)$. $\square$

When Π is clear from the context we may simply write P instead of P_Π. Conditional probability in P-log lite is defined in the usual way. Moreover, under certain consistency conditions (which hold for the examples discussed in this paper) on P-log lite programs T, formulas A, and a set of literals B such that $P_T(B) \neq 0$, we have $P_T(A|B) = P_{T \cup obs(B)}(A)$

2.2 Probabilistic causal models

In this section we present many definitions from [Pearl, 2000] and give our definition of a PCM-probabilistic query.

Causal model: A *causal model* is a triple $M = \langle U, V, F \rangle$ where

(i) U is a set of *background* variables, (also called *exogenous*), that are determined by factors outside the model;

(ii) V is a set $\{V_1, V_2, \ldots V_n\}$ of variables, called *endogenous*, that are determined by variables in the model - that is, variables in $U \cup V$; and

(iii) F is a set of functions $\{f_1, f_2, \ldots f_n\}$, such that each f_i is a mapping from $U \cup (V \setminus V_i)$ to V_i, and such that the entire set F forms a mapping from U to V. In other words, each f_i tells us the value of V_i given the values of all other variables in $U \cup V$, and the entire set F has a unique solution for V, given a realization of U. Symbolically, the set of equations F can be represented by writing $V_i = f_i(PA_i, U_i)$, $i = 1, \ldots, n$, where PA_i (connoting parents) is a subset of variables in $V \setminus V_i$ and U_i stands for a subset of variables in U.

We will refer to the realization of a set of variables $Y \subseteq V$ given the causal model M and the background variable realization of $U = u$ as $Y_M(u)$. $Y(u)$ will be used as shorthand for $Y_M(u)$, but we will explicitly use the subscript M_x for realizations of *submodels* defined below.

Example 1. We have a simple causal model consisting of a background variable U, endogenous variables A and B, and the set of the following causal functions:

$A = U \wedge B$. $\qquad$ $B = U \vee A$.

For each value of U, there is a unique solution for A and B. That is, for $U = 1$, $A = B = 1$ and for $U = 0$, $A = B = 0$. □

Submodel: Let M be a causal model, X be a set of variables in V, and x be a particular realization of X. A *submodel* M_x of M is the causal model $M_x = \langle U, V, F_x \rangle$ where $F_x = \{f_i : V_i \notin X\} \cup \{X = x\}$.

Submodels are useful for representing the effect of local actions and hypothetical changes. M_x represents the model that results from a minimal change to make $X = x$ hold true under any realization of U.

Effect of Action: Let M be a causal model, X be a set of variables in V, and x be a particular realization of X. The *effect of action* do(X = x) on M is given by the submodel M_x.

Potential Response: Let X and Y be two subsets of variables in V. The *potential response* of Y to action do(X = x), denoted $Y_{M_x}(u)$, is the solution for Y of the set of equations F_x, where u is a particular realization of U.

Counterfactual: Let X_i and Y be two subsets of variables in V. The *counterfactual* sentence "The value that Y would have obtained, had X_i been forced to be x_i" is interpreted as denoting the potential response $Y_{M_{x_i}}(u)$, where u is a particular realization of U.

Probabilistic causal model: A *probabilistic causal model* (PCM) is a pair $\langle M, P(u) \rangle$ where M is a causal model and $P(u)$ is a probability function defined over the domain of U.

Because the set of functional equations forms a mapping from U to V, the probability distribution P also induces a probability distribution over the endogenous variables.

For every set of variables $Y \subseteq V$, we have: $P(Y = y) = \sum_{\{u \mid Y(u)=y\}} P(u)$

The probability of a counterfactual statement is defined using the *potential response*, $Y_{M_x}(u)$, induced by the submodel M_x. It can be expressed in a similar manner: $P(Y_{M_x} = y) = \sum_{\{u \mid Y_{M_x}(u)=y\}} P(u)$

PCM probabilistic query: Joint probabilities of counterfactuals that are conditional on observations will be the focus of our counterfactual reasoning. Let $Y_{M_{x_1}} = y_1, \ldots, Y_{M_{x_m}} = y_m$ be counterfactuals and let B and C be subsets of V. A *PCM probabilistic query* is of the form P($Y_{M_{x_1}} = y_1, \ldots, Y_{M_{x_m}} = y_m$, B = b | $C = c$), and its value is given by:

$$\sum_u P(Y_{M_{x_1}} = y_1, \ldots, Y_{M_{x_m}} = y_m, B = b \mid u) P(u \mid c)$$

3 Encoding Pearl's firing squad in P-log lite

In this section we will consider Pearl's firing squad example and show how one can encode this example in P-log lite and ask counterfactual queries.

The firing squad PCM: The PCM of the *firing squad* has two background variables U and W, and endogenous variables A, B, C, and D; which stand for

U = court orders the execution; C = captain gives a signal; A = rifle A shoots; B = rifle B shoots; D = the prisoner dies; W = rifle A pulls the trigger out of nervousness.

The causal relationships between the variables are described by the functional equations $C = U$; $\quad A = C \vee W$; $\quad B = C$; $\quad D = A \vee B$. In addition the probabilities of the background variables are given as $P(U) = p$ and $P(W) = q$.

3.1 Example of an observation assimilation query

Consider the query that asks the probability that B shot given that the prisoner is dead? In PCM formalism this will be expressed as $P(B|D)$.

To translate this query to an equivalent P-log lite query we will use a program Π_1 that captures both the causal relationships and the probabilistic arguments found in the *firing squad* PCM. The program Π_1 will have explicit rules for the positive and negative boolean variables. The P-log lite predictive query is not conditioned on the fact that the prisoner is dead. Instead, the fact that the prisoner is dead is added to Π_1 as an observation rule $obs(D)$. The P-log lite program Π_1 consists of the following:

1. Declarations for each background variable:
   $U : boolean. \qquad random(U).$
   $W : boolean. \qquad random(W).$
2. Declarations for each endogenous variable:
   $C : boolean. \qquad A : boolean.$
   $B : boolean. \qquad D : boolean.$
3. Pr atoms and logic programming rules
   $pr(U) = p. \qquad pr(W) = q.$
4. Reminder of the logic programming rules
   $C \leftarrow U. \qquad\qquad\qquad \neg C \leftarrow \neg U.$
   $A \leftarrow C. \quad A \leftarrow W. \qquad \neg A \leftarrow \neg C \wedge \neg W.$
   $B \leftarrow C. \qquad\qquad\qquad \neg B \leftarrow \neg C.$
   $D \leftarrow A. \quad D \leftarrow B. \qquad \neg D \leftarrow \neg A \wedge \neg B.$

Result 1. $P(B|D) = P_{\Pi_1 \cup obs(D)}(B) = \frac{p}{1-(1-p)(1-q)}$

3.2 Example of an intervention query

Now let us consider the following intervention query: What is the probability that the prisoner is dead given that A is not allowed to shoot? In PCM formalism this is expressed as $P(D_{M_{\neg A}})$.

To translate this query to an equivalent P-log lite query we will use a program Π_2 which is similar to Π_1 with respect to the parts (1)-(3) of Π_1, but differs on part (4) by having rules which are blocked when an intervention is done. In this the nonmonotonic operator *not* of logic programs comes in handy.

Part (4) of Π_2

$C \leftarrow U,\ not\ do(C),\ not\ do(\neg C).$
$A \leftarrow C,\ not\ do(A),\ not\ do(\neg A).$
$A \leftarrow W,\ not\ do(A),\ not\ do(\neg A).$

$B \leftarrow C,\ not\ do(B),\ not\ do(\neg B).$
$D \leftarrow A,\ not\ do(D),\ not\ do(\neg D).$
$D \leftarrow B,\ not\ do(D),\ not\ do(\neg D).$

$\neg C \leftarrow \neg U,\ not\ do(C),\ not\ do(\neg C).$
$\neg A \leftarrow \neg C, \neg W,\ not\ do(A),\ not\ do(\neg A).$
$\neg B \leftarrow \neg C,\ not\ do(B),\ not\ do(\neg B).$
$\neg D \leftarrow \neg A, \neg B,\ not\ do(D),\ not\ do(\neg D).$

Result 2. $P(D_{M_{\neg A}}) = P_{\Pi_2 \cup do(\neg A)}(D) = p$

3.3 Example of a counterfactual query

We now consider the following counterfactual query from [Pearl, 2000]: "What is the probability that the prisoner would be alive if A had not shot given that the prisoner is in fact dead?"

In PCM this counterfactual query is expressed as either $P(\neg D_{M_{\neg A}} | D)$ or as $P(\neg D_{M_{\neg A}} | D_M)$, where M is the original causal model.

Thus the PCM counterfactual query has **two** causal models in the probabilistic query. The conditional variable D_M, which is an equivalent representation of variable D, is a member of the original causal model M, while the variable $Y_{M_{\neg A}}$ is a member of the submodel $M_{\neg A}$.

Our encodings in Π_1 refers to the model M and our encoding in Π_2 refers to the model $M_{\neg A}$. Now we need both. We achieve this by carefully using the indices 0 and 1 for the endogenous variables. We do not use any index for the exogenous variables as they remain the same in both models.

Note that in Π_1 we do not apply do() rules to the original model M, since by definition submodels are the result of an *effect of action* do() on the original model M. Therefore all variables that are members of the M do not need intervention rules in the translation.

The P-log lite encoding of the above example will consist of the program Π_3 given below. To ask the query all one needs to do is to add $obs(D(0))$ and $do(\neg A(1))$ to Π_3 and compute the probability of $\neg D(1)$. In the syntax of P-log lite this is expressed as $P_{\Pi_3 \cup obs(D(0)) \cup do(\neg A(1))}(\neg D(1))$.

The intuition behind adding $obs(D(0))$ is that D is observed with respect to the original model M which is encoded by the index 0. The intuition behind adding $do(\neg A(1))$ is that $\neg A$ is established (through an action) resulting in a new model $M_{\neg A}$ which is encoded by the index 1. The intuition behind asking the probability of $\neg D(1)$ is that the query is asked about the model $M_{\neg A}$ which, as we mentioned earlier, is encoded by the index 1. Now we describe the program Π_3. Its constituents are as follows:

- Declarations for each background variable:

  $U : boolean. \qquad random(U).$
  $W : boolean. \qquad random(W).$

- Probability atoms corresponding to the background variables U and W.

  $pr(U) = p. \qquad pr(W) = q.$

- Indices to enumerate the models M and $M_{\neg A}$:

  index = $\{0, 1\}$.

- Declarations for each endogenous variable:

  $C : index \rightarrow boolean. \qquad A : index \rightarrow boolean.$
  $B : index \rightarrow boolean. \qquad D : index \rightarrow boolean.$

- Rules that allow us to reason about the model M.

  We use the index 0 for the endogenous variables. The background variables do not have an index as they remain unchanged with respect to the original model and its sub-models.

  $C(0) \leftarrow U. \qquad \neg C(0) \leftarrow \neg U.$
  $A(0) \leftarrow C(0). \qquad \neg A(0) \leftarrow \neg C(0) \wedge \neg W.$
  $A(0) \leftarrow W.$
  $B(0) \leftarrow C(0). \qquad \neg B(0) \leftarrow \neg C(0).$
  $D(0) \leftarrow A(0). \qquad \neg D(0) \leftarrow \neg A(0) \wedge \neg B(0).$
  $D(0) \leftarrow B(0).$

- Rules that allow us to reason about the model $M_{\neg A}$.

  Since doing an action (as an assignment to a variable) necessitates surgery to the model so that the parents of the variable that is assigned no longer affect the variable; we do it logically by adding "*not* do" in the bodies of the rules. As a result we have the following rules:

  $C(1) \leftarrow U,\ not\ do(C(1)),\ not\ do(\neg C(1)).$
  $A(1) \leftarrow C(1),\ not\ do(A(1)),\ not\ do(\neg A(1)).$
  $A(1) \leftarrow W,\ not\ do(A(1)),\ not\ do(\neg A(1)).$
  $B(1) \leftarrow C(1),\ not\ do(B(1)),\ not\ do(\neg B(1)).$
  $D(1) \leftarrow A(1),\ not\ do(D(1)),\ not\ do(\neg D(1)).$
  $D(1) \leftarrow B(1),\ not\ do(D(1)),\ not\ do(\neg D(1)).$

  $\neg C(1) \leftarrow \neg U,\ not\ do(C(1)),\ not\ do(\neg C(1)).$
  $\neg A(1) \leftarrow \neg C(1) \wedge \neg W,\ not\ do(A(1)),$
  $\qquad not\ do(\neg A(1)).$
  $\neg B(1) \leftarrow \neg C(1),\ not\ do(B(1)),\ not\ do(\neg B(1)).$
  $\neg D(1) \leftarrow \neg A(1) \wedge \neg B(1),\ not\ do(D(1)),$
  $\qquad not\ do(\neg D(1)).$

To answer $P_{\Pi_3 \cup obs(D(0)) \cup do(\neg A(1))}(\neg D(1))$ all one needs to do is to add the intervention and observation facts do(¬A(1)), and obs(D(0)), respectively and compute the probability of $\neg D(1)$ with respect to the resulting theory.

Result 3. $P(\neg D_{M_{\neg A}} | D)$=
$P_{\Pi_3 \cup obs(D(0)) \cup do(\neg A(1))}(\neg D(1)) = \frac{(1-p)\times q}{1-(1-p)(1-q)}$

4 A general encoding of PCM to P-log lite

We now generalize the encoding illustrated in the previous section to arbitrary PCM theories and for queries that may refer to multiple hypothetical models. We will refer to our encoding[2] as T. To accommodate multiple submodels, the P-log lite rules will use an index variable S, where $S = 0$ will correspond to the original model of the PCM theory and, $S = 1 \ldots m$, will correspond to the m sub-models necessitated by the probabilistic query. The encoding T will consist of the following.

Step 1: Enumerate the models and submodels necessitated by the probabilistic query $0, \ldots, m$. The variable S used in the

[2]Our encoding of a PCM theory and a query will be independent of each other.

endogenous variables rules is in the domain of $index$:
$index = \{0, \ldots, m\}$

Step 2: For each background variable $u \in U$, for which the PCM has the probability $p(u) = q, T$ will have the following:

$u : boolean. \qquad pr(u) = q. \qquad random(u).$

Step 3: For each endogenous variable v, with $x_1, \ldots, x_n \in U \cup (V \setminus v)$ and with the associated function $f_v(x_1, \ldots, x_n)$, we do the following:

1. Add the following declaration:
   $v : index \rightarrow boolean.$

2. Let the disjunctive normal form of $f_v(x_1, \ldots, x_n)$ be $c_1 \vee \ldots \vee c_n$, where each c_i is a conjunction of literals made up of from $x_1, \ldots, x_n$. In the translation the parameter S is added to each endogenous variable x_i of $x_1, \ldots, x_n$, such that x_i becomes x_i(S). The P-log lite program T will have the following rules:

   $v(S) \leftarrow c_1, \textit{not do}(v(S)), \textit{not do}(\neg v(S)).$

   $\vdots$

   $v(S) \leftarrow c_n, \textit{not do}(v(S)), \textit{not do}(\neg v(S)).$

3. Let the disjunctive normal form of $\neg f_v(x_1, \ldots, x_n)$ be $d_1 \vee \ldots \vee d_n$, where each d_i is a conjunction of literals made up of from $x_1, \ldots, x_n$. In the translation the parameter S is added to each endogenous variable x_i of $x_1, \ldots, x_n$, such that x_i becomes x_i(S). The P-log lite program T will have the following rules:

   $\neg v(S) \leftarrow d_1(S), \textit{not do}(v(S)), \textit{not do}(\neg v(S)).$

   $\vdots$

   $\neg v(S) \leftarrow d_n(S), \textit{not do}(v(S)), \textit{not do}(\neg v(S)).$

Step 4: Encoding the query $P(b, Y_{M_{x_1}} = y_1, \ldots, Y_{M_{x_m}} = y_m | c)$

4.1: For each counterfactual statement $Y_{M_{x_i}} = y_i$:

For each variable $v \in$ V that is a member of the realization of $X_i = x_i$: if v is positive, we add $do(v(S = i))$; else (i.e., v is negative) we add $do(\neg v(S = i))$.

4.2: For each variable $w \in c$: if w is positive, we add $obs(w(0))$; else (i.e., w is negative) we add $obs(\neg w(0))$. □

Before presenting a theorem that formally relates PCM theories and queries with our encoding, we need to consider a notion of "being able to compute unique solutions of a causal model using three valued iteration". Consider the causal model from Example 1 which had the functions: $A = U \wedge B$. $B = U \vee A$. In Example 1 we showed that for each value of U, there is a unique solution for A and B. In this case, these unique solutions can be computed in an iterative fashion. Suppose we initially assign the value $false$ to U. We then assign unknown to A and B. We now use 3-valued logic on the causal functions and compute the values of A and B. After the first iteration A has value $false$ and B has value $unknown$. After the second iteration we reach a fixpoint where A and B are both assigned to $false$. This is what we refer to as "being able to compute unique solutions of a causal model using three valued iteration."

Having unique solution does not necessarily mean that it can be obtained using the above mentioned three valued iteration. Following is a counterexample[3]:

Let $A = \neg B \vee \neg C \vee U$, $\;B = \neg A \wedge \neg C$, and $C = A \vee \neg B$. When U is false we can not derive the value of A, B, or C by a three valued iteration. But, there is the unique solution $A = true$, $B = false$, and $C = true$ that satisfies the causal functions.

Theorem 1. Given a PCM $\mathcal{M} = \langle M, P(U) \rangle$, let P denote the probabilistic query with respect to it and let the P-log lite program T be obtained by the encoding of $\mathcal{M}$ as described in the previous section. Let b, c, $y_1, \ldots, y_m$ and $x_1, \ldots, x_m$ be realizations of subsets of the endogenous variables V. Also let $Y_{M_{x_i}} = x_i$ be counterfactual statements, where M_{x_i} is a submodel of M. If all realizations of $U = u$ give unique solutions for endogenous variables in the submodels M_{x_i} and the unique solutions can be computed in an iterative fashion starting from the background variables we have the following:
$P(b, Y_{M_{x_1}} = y_1, \ldots, Y_{M_{x_m}} = y_m | c) =$
$P_{T \cup obs(c(0)) \cup do(x_1(1)) \cup \ldots \cup do(x_m(m)))}$
$(b(0), y_1(1), \ldots, y_m(m)).$

Discussion on the theorem and its proof:

Here the encoding has three main aspects: (i) the encoding of the PCM equations, (ii) taking into account the severing of connections between variables when an action is performed, and (iii) considering models and submodels in parallel during counterfactual reasoning. For part (i) the notion of "unique solutions being computed in an iterative fashion" is important as that leads to a correspondence between the unique solutions and the answer sets. To do (ii) we use the negation as failure operator "not" and to do (iii) we introduce multiple time lines. Overall, the proof is based on splitting the logic programming translation of T to many layers, where the bottom layer consists of the enumeration of the background variables.

4.1 PCM, PAL and P-log lite

A related work is [Tran and Baral, 2004], where an encoding of PCM in an action language PAL is given. The encoding here to a probabilistic logic programming language P-log lite is different from the encoding in [Tran and Baral, 2004]. There the key issue was to encode the severing of connection between variables when an action is performed. It was achieved by introducing an ab predicate and having effect axiom $make(v_i)$ $causes$ $\{ab(v_i), v_i\}$ and a static causal law $\neg ab(v_i)$ $causes$ $v_i \Leftrightarrow f_i(PA_i, u_i)$ to capture the PCM equation $v_i = f_i(PA_i, u_i)$.

5 Non-naive conditioning in P-log

We now show how P-log provides an elaboration tolerant way to perform non-naive conditioning in that it shows a

[3]This has significance beyond AI, as the recent thesis [Riedel, 2003] (awarded the best Ph.D thesis of Electrical Engineering at the California Institute of Technology for the year 2003) mentions three valued iteration but does not point out that the iteration does not necessarily lead to the unique solution.

way to incorporate new observations through simple addition of knowledge rather than a surgery of existing knowledge. Halpern in [Halpern, 2003], uses the term naive conditioning to refer to conditioning with respect to the 'naive' state space one usually constructs when trying to reason. He discusses several examples, where naive conditioning gives the wrong answer. Following is his second-ace puzzle:

A deck has four cards: the ace and the deuce of hearts and spades. After a fair shuffle, two cards are dealt to Alice. It is easy to see that, at this point, there is a probability of 1/6 that Alice has both aces, a probability of 5/6 that Alice has at least one ace, a probability of 1/2 that Alice has ace of spades and a probability of 1/2 that Alice has ace of hearts.

Alice then says, "I have an ace." Conditioning on this information (by discarding the possibility that Alice was dealt no aces), Bob computes the probability that Alice holds both aces to be 1/5. This seems reasonable. Next, Alice says, "I have the ace of spades." Conditioning on this new information, Bob now computes the probability that Alice holds both aces to be 1/3. But suppose that Alice had instead said, "I have the ace of hearts." It seems that a similar argument again shows that the conditional probability that Alice holds both aces is 1/3.

Is this reasonable? When Bob learns that Alice has an ace, he knows that she must have either the ace of hearts or the ace of spades. Why should finding out which particular ace it is raise the conditional probability of Alice having two aces? Put another way, if this probability goes up from 1/5 to 1/3 whichever ace Alice says she has, and Bob knows that she has an ace, then why isn't it 1/3 all along?

Halpern [Halpern, 2003] analyzes the above example and points out that it is important to know to what question Alice answered: "I have ace of spades." He goes on to say that if she is asked (Q1) to name an ace if she has any and it is given that if she has both aces she will pick one of their colors with equal probability, naive conditioning does not lead to the right answer. We agree with Halpern. In this case instead of the naive possible worlds, *one needs to take into account the new question* and create a new set of possible worlds. *P-log allows us to do that within the language in a nice way.* On the other hand if Alice's answer is with respect to the question (Q2) which asks if Alice has the ace of spades, then the assimilation of Alice's answer is different, and in this case naive conditioning works.

The second-aces puzzle in P-log:

Consider the following encoding Π_1 of the second-ace puzzle in P-log.[4]

$card1$ and $card2$ have the domain $card = \{1s, 2s, 1h, 2h\}$.

$random(card1)$.

$\neg p2(X) \leftarrow card1 = X. \quad p2(X) \leftarrow not\ \neg p2(X)$.

$random(card2 : \{X : p2(X)\})$.

[4]Here we use full P-log rather than P-log lite that we presented earlier. Among the differences, in full P-log the literals are more general and attributes can take values from a declared domain rather than being just true or false.

$has\_alice(X) \leftarrow card1 = X.$
$has\_alice(X) \leftarrow card2 = X.$

Result 4. Probabilities w.r.t. Π_1
(i) $P_{\Pi_1}(has\_alice(1s) \wedge has\_alice(1h)) = 1/6$
(ii) $P_{\Pi_1}(has\_alice(1s) \vee has\_alice(1h)) = 5/6$
(iii) $P_{\Pi_1}(has\_alice(1s)) = 1/2$
(iv) $P_{\Pi_1}(has\_alice(1s) \wedge has\_alice(1h) | has\_alice(1s)) = 1/3$

Now we will consider that Alice's response of "I have ace of spades" is with respect to the question Q1. In that case we propose that the new information is incorporated to Π_1 by adding the following to Π_1:

$random(o : \{X : p(X)\})$.
$p(1s) \leftarrow has\_alice(1s). \quad p(1h) \leftarrow has\_alice(1h)$.

We refer to the resulting P-log program as Π_2.

Result 5. $P_{\Pi_2}(has\_alice(1s) \wedge has\_alice(1h) | o = 1s) = 1/5$

A general methodology:

From the encodings in the previous section one can draw the lesson that when an observation x about a particular concept is made, one can incorporate that observation correctly and avoid possible error associated with naive conditioning by:

1. Adding rules for a new predicate $p(X)$ that define the value that o can take.
2. Adding the random declaration for a new attribute o: $random(o : \{X : p(X)\})$.
3. Adding the observation $obs(o = x)$.

Moreover the steps (1) and (2) are useful for defining the values that an attribute can take. It is used in Π_1 when defining the values the attribute $card2$ can take, after $card1$ has been assigned a value.

6 Conclusion

In this paper we have shown how to do non-naive conditioning and encode Pearl's probabilistic causal models (PCMs) in P-log lite. We show how one can reason about simple observations, observations as a result of answers given to specific queries, interventions and counterfactuals, in P-log lite. There is a distinct difference between how reasoning about interventions and counterfactuals is done in PCMs and in P-log lite. In the former, surgery is done on a given PCM theory to obtain sub-models due to interventions, while in the later one just needs to add *do* facts, and let the semantics take care of the reasoning. Another difference between the two approaches is that P-log, because of its superior logical knowledge representation aspect, allows one to specify more nuanced theories. For example, with respect to the firing squad example, one can express knowledge such as "Rifleman B normally follows the captain's order". An elaboration tolerant representation of "normally" will allow easy elaboration of new knowledge such as: "B, as a conscientious objector to death penalty, defies his captain with respect to shooting." Such representation and elaboration is straightforward in P-log, while the corresponding PCM theory will need surgery.

7 Acknowledgement

This work was supported by a contract from ARDA and NSF grant 0412000. We thank Michael Gelfond and Nelson Rushton for discussions on this topic.

References

[Bacchus *et al.*, 1996] F. Bacchus, A. Grove, J. Halpern, and D. Koller. From statistical knowledge bases to degrees of belief. *Artificial Intelligence*, 87:75–143, 1996.

[Bacchus, 1990] F. Bacchus. *Representing and reasoning with uncertain knowledge*. MIT Press, 1990.

[Baral *et al.*, 2004] C. Baral, M. Gelfond, and N. Rushton. Probabilistic reasoning with answer sets. In *Proceedings of LPNMR7*, pages 21–33, 2004.

[Baral *et al.*, 2005] C. Baral, M. Gelfond, and N. Rushton. Probabilistic reasoning with answer sets, 2005. Draft: http://www.public.asu.edu/∼cbaral/papers/plog05.pdf.

[Breese, 1990] J. Breese. Construction of belief and decision networks. Technical report, Tech. Memorandom 90, Rockwell International Science Center, Palo Alto, CA, 1990.

[Cussens, 1999] J. Cussens. Loglinear models for first-order probabilistic reasoning. In *Proceedings of the Fifteenth Conference on UAI*, pages 126–133, 1999.

[De Vos and Vermeir, 2000] M. De Vos and D. Vermeir. Dynamically ordered probabilistic choice logic programming. In *Proceedings of the 20th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS2000)*, pages 227 – 239, 2000.

[Dekhtyar and Dekhtyar, 2004] A. Dekhtyar and M. Dekhtyar. Possible worlds semantics for probabilistic logic programs. In *ICLP*, pages 137–148, 2004.

[Fitting, 1985] M. Fitting. A Kripke-Kleene semantics for logic programs. *Journal of Logic Programming*, 2(4):295–312, 1985.

[Getoor *et al.*, 2001] L. Getoor, N. Friedman, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In *Relational data mining*, pages 307–335. Springer, 2001.

[Halpern, 1990] J. Halpern. An analysis of first-order logics of probability. *ArtificialIntelligence*, 46:311–350, 1990.

[Halpern, 2003] J. Halpern. *Reasoning about Uncertainty*. MIT Press, 2003.

[Kersting and Raedt, 2000] Kristian Kersting and Luc De Raedt. Bayesian logic programs. In J. Cussens and A. Frisch, editors, *Proceedings of the Work-in-Progress Track at the 10th International Conference on Inductive Logic Programming*, pages 138–155, 2000.

[Koller, 1999] D. Koller. Probabilistic relational models. In *ILP99*, pages 3–13, 1999.

[Lukasiewicz, 1998] T. Lukasiewicz. Probabilistic logic programming. In *ECAI*, pages 388–392, 1998.

[Muggleton, 1995] S. Muggleton. Stochastic logic programs. In L. De Raedt, editor, *Proceedings of the 5th International Workshop on Inductive Logic Programming*, page 29. Department of Computer Science, Katholieke Universiteit Leuven, 1995.

[Ng and Subrahmanian, 1992] Raymond T. Ng and V. S. Subrahmanian. Probabilistic logic programming. *Information and Computation*, 101(2):150–201, 1992.

[Ng and Subrahmanian, 1994] Raymond T. Ng and V. S. Subrahmanian. Stable semantics for probabilistic deductive databases. *Information and Computation*, 110(1):42–83, 1994.

[Ngo and Haddawy, 1997] Liem Ngo and Peter Haddawy. Answering queries from context-sensitive probabilistic knowledge bases. *Theoretical Computer Science*, 171(1–2):147–177, 1997.

[Nilsson, 1986] N. Nilsson. Probabilistic logic. *Artificial Intelligence*, 28:71–87, 1986.

[Paskin, 2002] M. Paskin. Maximum entropy probabilistic logic. Tech. Report UCB/CSD-01-1161, Computer Science Division, Univ. of California, Berkeley, CA, 2002.

[Pasula and Russell, 2001] H. Pasula and S. Russell. Approximate inference for first-order probabilistic languages. In *Proceedings of IJCAI 01*, pages 741–748, 2001.

[Pearl, 2000] J. Pearl. *Causality*. Cambridge University Press, 2000.

[Poole, 1993] David Poole. Probabilistic horn abduction and bayesian networks. *Artificial Intelligence*, 64(1):81–129, 1993.

[Poole, 1997] D. Poole. The independent choice logic for modelling multiple agents under uncertainty. *Artificial Intelligence*, 94(1-2):7–56, 1997.

[Poole, 2000] D. Poole. Abducing through negation as failure: Stable models within the independent choice logic. *Journal of Logic Programming*, 44:5–35, 2000.

[Richardson and Domingos, 2006] M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62:107–136, 2006.

[Riedel, 2003] M. Riedel. *Cyclic Combinational Circuits*. PhD thesis, California Institute of Technology, 2003.

[Riezler, 1998] S. Riezler. *Probabilistic constraint logic programming*. PhD thesis, University of Tubingen, Tubingen, Germany, 1998.

[Santos Costa *et al.*, 2003] V. Santos Costa, D. Page, M. Qazi, and J. Cussens. CLP(BN): Constraintlogic programming for probabilistic knowledge. In *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*, pages 517–524, 2003.

[Tran and Baral, 2004] Nam Tran and Chitta Baral. Encoding probabilistic causal models in probabilistic action language pal. In *AAAI'04*, 2004.

[Vennekens *et al.*, 2004] J. Vennekens, S. Verbaeten, and M. Bruynooghe. Logic programs with annotated disjunctions. In *ICLP*, pages 431–445, 2004.

[Wellman *et al.*, 1992] M. Wellman, J. Breese, and R. Goldman. From knowledge bases to decision models. *Knowledge Engineering Review*, pages 35–53, 1992.