AMAI Special Issue for ASPOCP manuscript No.
(will be inserted by the editor)

P-log: refinement, extension, and a new coherency condition.

Evgenii Balai · Michael Gelfond · Yuanlin Zhang

Received: date / Accepted: date

Abstract This paper focuses on the investigation and improvement of knowledge representation language P-log that allows for both logical and probabilistic reasoning. We refine the definition of the language by eliminating some ambiguities and incidental decisions made in its original version, slightly modify the formal semantics to better match the intuitive meaning of the language constructs, and introduce a simple extension of the language which allows to express new and useful types of reasoning. We also define a new class of coherent (i.e., logically and probabilistically consistent) P-log programs which facilitate their construction and proofs of correctness. There are a query answering algorithm, sound for programs from this class, and a prototype implementation which, due to their size, are not included in the paper. They, however, can be found in the dissertation of the first author.

Keywords Answer Set Programming · Probabilistic Inference · Knowledge Representation

PACS 60 · 68

1 Introduction

The language P-log, introduced in [7,8], is capable of combining non-monotonic logical reasoning about agent's beliefs in the style of Answer Set Prolog (ASP) [12] and probabilistic reasoning with Causal Bayesian Networks [21]. In addition to being logically non-monotonic, P-log is also "probabilistically non-monotonic" — an addition of new information can add new possible worlds and substantially change the original probabilistic model. Another distinctive feature of P-log is its

Evgenii Balai
Texas Tech University E-mail: evgenii.balai@gmail.com

Michael Gelfond
Texas Tech University E-mail: michael.gelfond@ttu.edu

Yuanlin Zhang
Texas Tech University E-mail: y.zhang@ttu.edu

ability to reason with a broad range of updates. In addition to standard conditioning on observations, P-log allows conditioning on rules, including defaults and rules introducing new terms, deliberate actions in the sense of Pearl, etc. These and several other features make P-log representations of non-trivial probabilistic scenarios (including such classical "puzzles" as Simpson Paradox, Monty Hall Problem, etc.) very close to their English descriptions which greatly facilitate a difficult task of producing their probabilistic models and sheds some light on subtle issues involved in their solutions. The ability of P-log to represent causal and counterfactual reasoning is discussed in [9]. In [14] P-log was expanded to allow abductive reasoning in the style of CR-Prolog [6] and infinite possible worlds. A P-log prototype query answering system (based on ASP solvers) is described in Zhu's dissertation [25]. The system allowed the use of P-log in a number of applications such as software for finding best probabilistic diagnosis for certain types of failure in the Space Shuttle [25], development and automation of mathematical models of machine ethics [22], methods for combining probabilistic and logical reasoning in robotics [24], etc.

This paper describes improvements of P-log in the following directions.

1. Refine and expand the syntax and semantics of P-log by:
   - clarifying and improving the original semantics of partial functions and activity records,
   - allowing statements used to record activities of a reasoner to occur in the bodies of rules.
2. Define a new class of coherent (i.e, probabilistically and logically consistent) P-log programs which includes a number of classical examples not included in the class defined in [8].

In what follows we briefly elaborate on particular features of the original P-log we found not fully satisfactory and describe the refinements introduced in this paper.

Negative literals in the heads: One of the oversights made in the design of the original P-log was allowing negative literals of the form $f(\bar{x}) \neq y$ in the head of a rule. If f is a total function then the decision does not cause any problems. If, however, f is partial, it leads to a discrepancy between intuitive meaning of the program and its formal semantics. To see that let us consider a program P consisting of rules:

$f : boolean$
$f \neq false$

Intuitively, $f \neq false$ means that f is defined, i.e. has a value and this value is different from $false$. The intuition agrees with some other extensions of ASP with functional symbols, e.g. [5]. Together with declaration $f : boolean$ this should imply that the value of f is $true$. However, the program P has one possible world consisting of a literal $f \neq false$ and hence P does not entail $f = true$. To avoid this discrepancy, the new version of P-log prohibits the use of negative literals in the heads of rules. This restriction does not seem to cause any problems from the knowledge representation perspective. Moreover, disallowing this syntactic feature leads to a substantial simplification of the formal semantics of P-log. Instead of defining possible worlds as sets of literals we can view them simply as (partial) interpretations of the attribute terms from the program's signature (in other words,

as collections of atoms). So far we were not able to find any adverse effect of our restriction on the original syntax.

P-log observations: Another problem with the original P-log is related to the intuitive meaning of P-log observations. According to [8], such observations are used "to record the outcomes of random events, i.e., random attributes, and attributes dependent on them". However, axioms 10 – 13 from the original paper do not faithfully reflect this intuition. Axiom (12), for instance, does not prohibit observations of non-random events. Instead, it simply views $obs(f(\bar{x}) = y)$ as a shorthand for the constraint

$$\leftarrow not\ f(\bar{x}) = y$$

where f is an arbitrary attribute. The new observation simply eliminates some of the possible worlds of the program, which reflects understanding of observations in classical probability theory. This view is also compatible with the treatment of observations in action languages. So if we limit ourselves to the syntax of traditional P-log there are no adverse consequences of expanding the observability of attribute values to a non-random case. Section 3 will contain some examples of the use of this extension.

P-log intervening actions: Let us now attempt to clarify the P-log meaning of the *do* statement. The original paper states: "the statement $do(f(\bar{x}) = y)$ indicates that $f(\bar{x}) = y$ is made true as a result of a deliberate (non-random) action". Note that, here, $f(\bar{x})$ is not required to be declared as random, i.e. its value does not have to be defined by a random experiment. This is not wrong. Even though the original intervening action *do* of Pearl only applies to random attributes (no other types are available in Bayesian Nets), nothing prohibits us from expanding the domain of *do* to non-random ones. After all, this is exactly what we did with observations. But, in the case of intervening actions, such an extension seems to be unwarranted. It is easy to see that for non-random $f(\bar{x})$, $do(f(\bar{x}) = y)$ is (modulo *do*) equivalent to $f(\bar{x}) = y$, which undermines the utility of such statements. In addition, it violates an important principle of language design frequently advocated by N. Wirth and others: *Whenever possible, make sure that each important type of informal statements you want expressible in your formal language corresponds to one language construct.* Moreover, applying *do* to interfere with a random experiment in which the value of random $f(\bar{x})$ with range R is selected from the collection of values satisfying some property p (referred to as dynamic domain of $f(\bar{x})$) causes an ambiguity of an interpretation: should the deliberately assigned value belong to the dynamic domain or arbitrary value of R must be allowed? The formal semantics from [8] corresponds to the second option, but, according to the best recollection of the authors of [8], this is accidental. We believe now that the first approach is more faithful to the intuition behind the notion of dynamic domain. We may avoid this redundancy and ambiguity by introducing a new special attribute term, $truly\_random(a)$, meaning that the value of a was chosen by a genuine random experiment, and slightly modifying the definition of the set of general axioms of P-log as described in Section 2.

Actions and observations in the bodies of rules: Another important modification of the language is allowing literals formed by *do* and *obs* (i.e, actions and observations) to occur in the bodies of P-log rules. We have already mentioned

that the addition of an observation to a program in the original P-log language may only eliminate some of its possible worlds but cannot create a new one. Allowing observations to occur in the bodies of P-log rules changes the situation. Consider a program

$$Q = \begin{cases} \neg a \leftarrow not\ a. \\ a \leftarrow obs(a = true). \end{cases}$$

which has an observation in the body of the second rule. Intuitively, the program has one possible world $\{\neg a\}$. But, the possible world of program Q' obtained by the addition of an observation $obs(a = true)$ to Q is $\{a, obs(a = true)\}$. The new observation creates a possible world which did not exist according to the original program. This extension of the language does not significantly complicate the mathematical semantics of the language but seems to add substantially to its expressive power. More examples that use the new extension can be found in Section 3.

New Class of Coherent Programs and Inference Algorithm. In the original paper [8], the authors define the notion of coherent (logically and probabilistically consistent) programs. They also define sufficient conditions for a program to be coherent. Unfortunately, there are interesting and important coherent programs which do not satisfy those conditions. In this paper we introduce new sufficient conditions and provide a number of examples of coherent programs that satisfy our conditions, but do not satisfy those from [8]. Section 4 contains the details. There are also a query answering algorithm, sound for programs from this newly introduced class of coherent programs, and a prototype implementation [1]. which, due to their size, are not included in the paper. They, however, can be found in the dissertation of the first author [1].

The paper is organized as follows. The first section contains the syntax and semantics of the new version of P-log. The second gives examples of the use of rules with observations and deliberate actions in the body for knowledge representation. The third describes the new class, B, of coherent programs and gives a number of examples of programs from this class which were not covered by previous results. Appendix (which currently can be found in the full version of the paper located at ???) contains the proof of coherency of programs from B. The paper is a substantial extension of the original workshop version [3]. It contains more detailed and accurate description of the language, more examples, definition of class B and the corresponding coherency theorem. Moreover, all examples of programs given in the paper are run on prototype implementation[2].

2 Syntax and Semantics of P-log

A program of the new version of P-log will be defined as a pair consisting of a sorted signature and a collection of P-log rules and causal probability atoms. As usual, the program will define the collection of possible worlds corresponding to the beliefs of a rational reasoner associated with it as well as the probability function

[1] The system can be downloaded from https://github.com/iensen/plog2.0/wiki

[2] The programs, written in syntax suitable for our current implementation, can be found at https://github.com/iensen/plog2.0/tree/master/plogapp/tests/aspocp_amai.

on the sets of these worlds describing degrees of the reasoner's beliefs. We start with defining the sorted signature of P-log and the syntax of P-log programs.

Syntax:

A *sorted signature* Σ of P-log is a tuple $\langle S, O, F\rangle$ where S is a finite non-empty set of sort names, O is a finite set of object constants, and F is a finite non-empty set of functional symbols.

- Every sort name $s \in S$ is assigned the sort denoted by it - a collection of object constants from O. We say that an object o from this collection belongs to (or is an instance of) sort s and write $o \in s$. We assume that Σ contains sorts **N** and **Q** of natural and rational numbers which are mapped into standard representations of these numbers viewed as elements of O. Whenever it is clear from the context, we will abuse the notation by using the same letter to refer to the sort name and the sort denoted by it.
- Every function symbol from F has sort names assigned to its parameters and its range. In what follows we use standard mathematical notation $f : s_1, \ldots, s_n \rightarrow s$ to describe these assignments.
- Set F is partitioned into two parts: *attributes* and *arithmetic* functions, such as $+$, $-$, etc., defined on natural or rational numbers.

Every sorted signature of P-log has three types of variable-free terms:

- Standart *arithmetic terms*.
- *Regular terms* - expressions of the form $f(c_1, \ldots, c_n)$ where f is an attribute $f : s_1, \ldots, s_n \rightarrow s$, c's are constants and, for every i, $c_i \in s_i$. We say that s is the *range* of term $f(c_1, \ldots, c_n)$.
- *Special boolean terms* listed below. Note that $f(\bar{x})$ denotes a regular term, y is a constant from the range of $f(\bar{x})$, and p is a unary boolean attribute.
  - $do(f(\bar{x}), y)$, which reads as "a random experiment assigning value to $f(\bar{x})$ is deliberately interfered with and $f(\bar{x})$ is assigned the value y". (In some cases it is also convenient to use $do(a, f(\bar{x}), y)$, where a stands for an agent performing the interference. Similarly for an attribute *obs* below.)
  - $obs(f(\bar{x}), y, true)$, which reads as "the value of $f(\bar{x})$ is observed to be y" and $obs(f(\bar{x}), y, false)$, which reads as "the value of $f(\bar{x})$ is observed to be different from y".[3]
  - $random(f(\bar{x}), p)$, which says that "$f(\bar{x})$ may take the value from $\{X : p(X)\}$ as the result of a random experiment which is either genuine or deliberately interfered with." For the sake of compatibility with original P-log, we will sometimes write $random(f(\bar{x}) : \{X : p(X)\})$ instead of $random(f(\bar{x}), p)$. The argument p can be omitted, in which case $random(a)$ is understood as $random(a : r\})$, where r is the range of a.
  - $truly\_random(f(\bar{x}))$, which says that "$f(\bar{x})$ takes value as the result of a genuine random experiment (i.e., the one without any outside interference)".

  To maintain compatibility with [8], non-arithmetic terms will be often referred to as *attribute terms*.

[3] To simplify the notation, we sometimes write $obs(f(\bar{x}), y, true)$ and $obs(f(\bar{x}), y, false)$ as $obs(f(\bar{x}), y)$ (or $obs(f(\bar{x}) = y)$) and $\neg obs(f(\bar{x}), y)$ (or $obs(f(\bar{x}) \neq y)$) respectively; if f is boolean, then $obs(f(\bar{x}), true, true)$ will be written as $obs(f(\bar{x}))$.

In what follows "signature" refers to a sorted signature of P-log.

An *atom* (or *positive literal*) of signature Σ is a statement of one of the forms:

1. $t = y$, which says that "y is the value of t", where t is a non-arithmetic term and y is an object constant from the range of t, or
2. $t_1 \odot t_2$ where t_1 and t_2 are arithmetic terms and $\odot$ is one of the standard arithmetic relations, $=$, $\neq$, $>$, etc. These atoms are called *arithmetic*

Sometimes we use $atoms(\Sigma)$ to denote the set of atoms over signature Σ. A *negative literal* is an expression $\neg(t = y)$, which says that "the value of t is different from y", or, equivalently, $t \neq y$. If t is boolean then $t = true$ and $t = false$ will often be written as t and $\neg t$. An atom of the form $t = y$ is called *special* if t is a special attribute term, otherwise it is called *regular*. If t is of the form $obs(f(\bar{x}), y, B)$ or $do(f(\bar{x}), y)$, the atom is called an *observation* or an *action* correspondingly. A *literal* of Σ is either a positive literal of Σ or a negative literal of Σ. A literal, possibly preceded by the default negation *not*, is called an *extended literal* or simply an *e-literal* of Σ. The e-literal **not** l reads as "l is not believed to be true" (which is, of course, different from "l is believed to be false").

A *P-log rule*, r over signature Σ is of the form:

$$head(r) \leftarrow body(r) \tag{1}$$

where $head(r)$ is an atom of Σ and $body(r)$ is a collection of e-literals of Σ. As discussed in the introduction, compared to the original version, the syntax defined here does not allow negative literals in heads. The head of the rule can optionally be omitted. Such rules are referred to as *constraints*. If $head(r)$ is an observation or an action, we require the *body* to be empty and the rule is called *an activity record*. If $head(r)$ is of the form $random(a : \{X : p(X)\})$, the rule is called *a random selection rule*. A rule which is not an activity record or a random selection rule is called *a regular rule*.

Note that, according to our definition, the rules of P-log do not contain variables. As it is normally done in logic programming, however, variables (denoted by identifier starting with an upper case letter) will be used, but a *rule with variables will be understood as a shorthand for the collection of rules obtained by replacing the variables with the properly sorted object constants of* Σ.

By a *P-log program* we mean a pair consisting of a signature Σ and a collection R of P-log rules and *causal probability statements* (also called *pr-atoms*) – expressions of the form

$$pr(f(\bar{x}) = y \mid B) = v \tag{2}$$

where $f(\bar{x})$ is a regular attribute term such that $y \in range(f(\bar{x}))$, B is a set of e-literals of Σ and $v \in [0, 1]$ is a rational number. The statement says that "if the value of $f(\bar{x})$ is generated randomly and B holds then the probability of the selection of y for the value of $f(\bar{x})$ is v. Moreover, there is a potential existence of a direct causal relationship between B and the possible value of $f(\bar{x})$." We will refer to $f(\bar{x}) = y$ as the *head* of the pr-atom and to B as the *body* of the *pr*-atom. We will refer to v as the *probability assigned by the pr-atom*.

Unless otherwise stated, we will assume that R contains the following rules, called *general P-log axioms*:

– For every attribute term $f(\bar{x})$ of Σ which is not special, the rules:

$$\leftarrow not\ f(\bar{x}) = Y,\ obs(f(\bar{x}), Y, true) \tag{3}$$

$$\leftarrow not\ f(\bar{x}) \neq Y,\ obs(f(\bar{x}), Y, false). \tag{4}$$

The rules are often referred to as *reality check axioms*. Intuitively, they prohibit observations of undefined attribute terms as well as observations which contradict the agent's beliefs. Note that, unlike those from the original version, the observed attribute terms are not restricted, explicitly or implicitly, to random attribute terms and the attribute terms dependent on them.

– For every random atom $random(f(\bar{x}), p)$ of Σ such that $range(f) = \{y_1, \ldots, y_k\}$, the rules :

$$f(\bar{x}) = y_1\ \mathbf{or}\ \ \ldots\ \mathbf{or}\ \ f(\bar{x}) = y_k\ \ \leftarrow random(f(\bar{x}), p)^4 \tag{5}$$

$$\begin{array}{l} truly\_random(f(\bar{x})) \leftarrow random(f(\bar{x}), p\}), \\ \qquad\qquad\qquad not\ do(f(\bar{x}), y_1), \ldots, not\ do(f(\bar{x}), y_k) \end{array} \tag{6}$$

$$\leftarrow f(\bar{x}) = Y,\ \mathbf{not}\ \ p(Y), random(f(\bar{x}), p) \tag{7}$$

$$\begin{array}{l} \leftarrow not\ f(\bar{X}) = Y, \\ \quad do(f(\bar{X}), Y). \end{array} \tag{8}$$

Intuitively, the rules (5) and (7) guarantee that if $random(f(\bar{x}), p)$ is true, then $f(\bar{x})$ is assigned the value satisfying condition p, rule (6) makes sure that $truly\_random(f(\bar{x}))$ is true iff the value of $f(\bar{x})$ is assigned as the result of a truly random experiment, i.e. an experiment without any intervention, and rule (8) guarantees that the atoms made true by interventions are indeed true.

– The rule

$$\begin{array}{l} \leftarrow not\ random(f(\bar{x}), p), \\ \quad \ldots, \\ \quad not\ random(f(\bar{x}), p), \\ \quad do(f(\bar{x}), Y). \end{array} \tag{9}$$

where $random(f(\bar{x}), p), \ldots, random(f(\bar{x}) : \{X : p_n(X)\})$ are all special attribute terms of Σ formed by $random$ with $f(\bar{x})$ as the first argument. Intuitively, the rule guarantees that an attempt to apply do to a non-random $f(\bar{x})$ leads to inconsistency. As discussed in the introduction, this axiom was not present in the original P-log.

– Versions of axioms containing obs and do with an additional "agent" parameter.

In addition, for every rule r which is not a general axiom, we disallow literals formed by $truly\_random$ and $random$ to occur in the body of r. Elements of a program such as terms, e-literals, rules, programs, etc., are called *ground* if they contain no variables and no names of arithmetic functions.

As was mentioned in the introduction, syntax differs from the syntax defined in [8] in the following ways:

[4] Disjunction here is a so called shifted disjunction [10] and hence, can be eliminated.

a) Partial attributes are explicitly allowed in the language.
b) Special attribute terms like *obs* and *do* are allowed to occur in rules' bodies.
c) Negative literals are not allowed in the rules' heads.

More details on the proposed changes and a more general definition of P-log syntax can be found in [1,3].

Semantics

Now we are ready to define the semantics of P-log programs. Since every variable-free program can be translated into a ground one by evaluating its arithmetic expressions in what follows by a program we will mean a ground program of P-log.

The semantics of such a program Π is given by the collection $\mathcal{W}(\Pi)$ of possible worlds of Π and the probability function P_Π[5], defined on the sets of these worlds. The former correspond to possible sets of beliefs of a rational agent associated with the program, and the latter specifies degrees of such beliefs. The definition of $\mathcal{W}(\Pi)$ is very similar to the definition of answer sets of logic programs. An usual, *interpretation* over signature Σ is a (possibly partial) mapping I from the attribute terms of Σ into values from their corresponding ranges. We assume that on arithmetic symbols I coincides with their standard interpretation and that every interpretation contains $s(x)$ for every sort s and $y \in s$. Since attributes of P-log can be partial, special care should be taken in defining satisfiability relation ($\models$) for atoms preceded by $\neg$ and **not**. In order to define the realtion, we will use the notation for elements of Σ: a denotes a ground attribute term, l denotes a literal, el denotes an extended literal, and B denotes a set of extended literals.

The *satisfiability relation* is defined as follows:

1. $I \models a = y$ if $I(a) = y$,
2. $I \models \neg(a = y)$ if $I(a) = y'$ where $y' \neq y$,
3. $I \models$ **not** l if $I \not\models l$,
4. $I \models B$ if for every $el \in B$, $I \models el$,
5. $I \models l \leftarrow B$ if $I \not\models B \vee I \models l$, and
6. $I \models \leftarrow B$ if $I \not\models B$.

We say that an atom A of Σ is *true* in I if $I \models A$ and *false* in I if $I \models \neg A$. If A is neither *true* nor *false* in I then it is *undefined* in I. We will often represent an interpretation I as the set of non-arithmetic atoms satisfied by I. Note, that an interpretation in which a and b are undefined satisfies rule $a \leftarrow \neg b$ but does not satisfy rule $a \leftarrow$ **not** b. Let Π^I denote the standard *reduct* [11] of a program Π with respect to interpretation I. Then I is a possible world of Π if

1. Every rule of Π^I is satisfied by I.
2. There is no interpretation I_0 such that $I_0 \subsetneq I$ and I_0 satisfies the rules of Π^I.

The following examples illustrate the definition. Recall that, in addition to the explicitly stated rules, every program below contains general P-log axioms 3–9. To simplify the specification of programs' signatures, we use standard mathematical declarations of functions, $f : s_1, \ldots, s_n \rightarrow s$ and sorts $s = \{t_1, \ldots, t_m\}$. If $n = 0$,

[5] When Π is clear from the context we may simply write P instead of P_Π.

we simply write $f : s$. We follow our system's convention and start the sort names with #. Also, we will assume that the boolean sort

$$\#boolean = \{true, false\}$$

is defined for every program.

Example 1 Consider the program P_1:

```
a,b,c: #boolean.
random(a).
b :- c, -a.
do(a, false).
random(c).
```

It is not difficult to see that the program has two possible worlds W_1 and W_2:

$$W_1 = \{\neg a, b, c, do(a, false), random(a), truly\_random(c), random(c)\}$$

and

$$W_2 = \{\neg a, \neg c, do(a, false), random(a), truly\_random(c), random(c)\}.$$

□

Example 2 Consider the program P_2:

```
a: #boolean.
obs(a=true).
```

P_2 has no possible worlds. Note that $W = \{obs(a = true)\}$ is not a possible world, because of the reality check axiom (3).

If we add the rule

$$random(a)$$

to P_2, it will have one possible world

$$W = (\{obs(a = true), a, truly\_random(a), random(a)\}),$$

where the observation $obs(a = true)$ is consistent with the belief in a.

If however we were to replace $random(a)$ by

$$\neg a$$

P_2 would become inconsistent again, because any interpretation that satisfies both rules $\neg a$ and $obs(a, true)$ will violate reality check axiom (3).

□

With one small exception which we will mention shortly the definition of P_{Π} in our paper is the same as that in [8]. We'll illustrate it by a simple example. More detailed information can be found in the original paper [8]. Consider a program P_3:

```
#s = {1,2,3}.
a : #s.
b : #boolean.
random(a).
random(b).
pr(a=1) = 1/2.
```

The program has six possible worlds corresponding to choices of values for a and b. For every possible world W and every attribute a such that $W \models truly\_random(a)$ we first define *causal probability*, $P(W, a = y)$ for every possible outcome y of a (for the precise definition of possible outcome we refer the reader to [8]). Let us consider a possible world W_1 containing $a = 1$ and $b = true$. Causal probability $P(W_1, a = 1)$ is directly determined by the *pr*-atom of the program and is equal to $1/2$; the value of $P(W_1, b = true)$ is determined by the indifference principle which says that possible values of random attribute are assumed to be equally probable if we have no reason to prefer one of them to any other, i.e., $P(W_1, b = true) = 1/2$. Now consider a possible world W_2 containing $a = 2$ and $b = true$. From the pr-atom of the program we know that the causal probability of a being equal to 2 or to 3 is $1/2$. Hence, by the indifference principle we have $P(W_2, a = 2) = 1/4$. Now we are ready to compute unnormalized measure, $\hat{\mu}_{\Pi}(W)$ defined as the product of causal probabilities of random atoms from W. In our case, $\hat{\mu}_{P_3}(W_1) = 1/2 \times 1/2 = 1/4$ and $\hat{\mu}_{P_3}(W_2) = 1/4 \times 1/2 = 1/8$. As expected, the measure, $\mu(W)$ is the unnormalized probability of W divided by the sum of the unnormalized probabilities of all possible worlds of Π. Suppose now that Π is a P-log program having at least one possible world with nonzero unnormalized probability. The *probability*, $P_{\Pi}(E)$, of a set E of possible worlds of program Π is the sum of the measures of the possible worlds from E. The *probability* with respect to program Π of a literal l of Π, $P_{\Pi}(l)$, is the sum of the measures of the possible worlds of Π that satisfy l.

As discussed in [8] the corresponding functions are only defined for programs which satisfy three conditions on possible worlds of a program Π. Roughly speaking, the conditions ensure that for every W there is at most one random selection rule determining possible values of a in W, at most one *pr*-atom which can be used to define $P(W, a = y)$, and that $P(W, a = y)$ is not defined for y outside of the set of a's possible values determined by the random selection rule for a. The first condition, is strengthened compared to its original version.

Condition 1 (Unique selection rule) If Π contains rules r_1 and r_2, such that none of them is an instance of a general axiom and, for some attribute term a

- $head(r_1)$ is of the form $a = y$ or $random(a, p)$, and
- $head(r_2)$ is of the form $a = y$ or $random(a, p)$,

then no possible world of Π satisfies $body(r_1)$ and $body(r_2)$.

□

The original condition allows for a program with a possible world W to contain rules $r_1 : random(a) \leftarrow B_1$. and $r_2 : a = y \leftarrow B_2$. s.t. W satisfies both B_1 and B_2, while the new one prohibits such programs. We believe that the new condition better captures the intuition of a unique value selection for random attribute

terms. Moreover, it is not clear whether or not a should be considered random in a possible world which satisfies the bodies of both of the rules r_1 and r_2.

The two later conditions remain unchanged. We restate them here for completeness. The second condition says:

Condition 2 (Unique probability assignment)
If Π contains a random selection rule

$$random(a(\bar{t}) : \{Y : p(Y)\}) \leftarrow B \tag{10}$$

along with two different probability atoms

$$pr(a(\bar{t}) \mid B_1) = v_1 \text{ and } pr(a(\bar{t}) \mid B_2) = v_2 \tag{11}$$

then no possible world of Π satisfies B, B_1, and B_2.

□

The justification of this condition is as follows. If B is satisfied by a possible world W of Π, then the random experiment defined by rule (10) is active and $a(\bar{t})$ takes a value in W from the range $\{Y : p(Y)\}$. In this case, if B_1 and B_2 are both satisfied by W, we have two different assignments to probability of $a = y$ which are either contradictory (if $v_1 \neq v_2$) or repeat the same information (if $v_1 = v_2$). Both situations are undesired.

Finally, the last condition is:

Condition 3 (No probabilities assigned outside of dynamic range)
If Π contains a random selection rule

$$random(a(\bar{t}) : \{Y : p(Y)\}) \leftarrow B_1 \tag{12}$$

along with probability atom

$$pr(a(\bar{t}) = y \mid B_2) = v \tag{13}$$

then no possible world of Π satisfies B_1 and B_2 but does not satisfy $p(y)$.

□

The rule guarantees that, if a possible world W satisfies B_1, B_2, but does not satisfy $p(y)$, then pr-atom (13), assigning probability to a value outside of the range $\{X : p(X)\}$ of possible values of $a(\bar{t})$ in W, is not active in W.

3 Reasoning with Observations and Actions

In this section we present several examples of the use of observations and actions in the bodies of rules for types of reasoning with both, logical and probabilistic information which do not seem to be naturally expressible in P-log's original version. We start with an example of P-log formalization of diagnostic reasoning from symptoms[6] to causes.

[6] By a symptom we mean a sign reflecting the presence of some abnormal state of affairs. We assume that a symptom can be observed by a patient or by an outside observer, and ignore the difference between the two types of observations commonly made in the medical profession.

Example 3 [reasoning from symptoms to causes]
Suppose that an experienced diagnostician knows that a certain symptom, s, has exactly two possible causes c_1 and c_2. This information can be expressed in P-log by a program P_4:

```
c1, c2, s: #boolean.
¬c1 :- not obs(s).
¬c2 :- not obs(s).
s :- c1.
s :- c2.
¬s :- not s.
```

The first two rules say that the causes are not true in case of the absence of an observation of the symptom – a natural default we use in our actions before becoming aware of a problem by experiencing its symptoms. The next three rules give the complete list of possible causes for s. As expected, according to P_4, the probabilities of s, c_1 and c_2 are 0. It is, however, important to note that an update of P_4 by an observation of the truth of any attribute of P_4 leads to inconsistency. This is not necessarily an unwelcome outcome for the observations of causes – after all, causes are normally not directly observable and need to be derived from the observations of symptoms and the background knowledge. This shall not however happen for the observation of the symptom s. The following informal argument is possible in this case: *Since we are given a complete list of possible causes of s and s is observed to be true we have a good reason to believe that at least one of its possible causes is true which should prevent us from using closed world assumptions for causes. Instead we should think of them as random attributes which may or may not be true. Consequently, possible states of the world will be those which contain causes explaining our symptom.* According to this reasoning, the program describing the agent's knowledge should have possible worlds $W_1 = \{c_1, \neg c_2, s\}$, $W_2 = \{\neg c_1, c_2, s\}$, and $W_3 = \{c_1, c_2, s\}$.

The missing knowledge used by the reasoner to go from observations of a symptom to its causes can be represented by expanding P_4 by the rules:

$$R = \begin{cases} random(c_1) \leftarrow obs(s) \\ random(c_2) \leftarrow obs(s) \end{cases}$$

which have observations in their bodies. It is easy to check that program

$$P_5 = P_4 \cup R \cup \{obs(s)\}$$

is consistent and has three possible worlds W_1, W_2, and W_3 described above. The program assigns probabilities $2/3$ to c_1 and to c_2 and probability 1 to s.

It is important to notice that, in this example, the addition of an observation of the symptom causes the reasoner to consider worlds which were not considered possible before the update. Consequently, the new observation changes probabilities of some events from 0 to a positive number (even to 1) – the type of non-monotonicity which is not possible in the original P-log.

It may be tempting to replace P_5 by program P_5', obtained from Π_5 by replacing R with collection of rules:

$$R' = \begin{cases} random(c_1) \leftarrow s \\ random(c_2) \leftarrow s \end{cases}$$

This, however, will not work, since the resulting program will be inconsistent. This is not surprising, since there is a substantial difference between s and $obs(s)$. The first is a fact and can be used by a reasoner to justify his belief in s. The second is a constraint which cannot be used for this purpose. As the result, allowing observations in the bodies of rules is essential for the type of reasoning discussed in this example.

Example 4 [Explaining away]
Let us consider the story and its formalization by program P_5 from the previous example. Now assume that, by checking some available statistics, the diagnostician acquire knowledge about probabilities of c_1 and c_2. These probabilities can be added to the program by extending it with the set PA of causal probability atoms:

$$PA = \begin{cases} pr(c_1) = 0.05 \\ pr(c_2) = 0.01 \end{cases}$$

The probabilities assigned to c_1 and c_2 by the new program,

$$P_6 = P_5 \cup PA$$

are now approximately 0.8 and 0.2. So c_1 is the most likely cause of the symptom.

Finally, let us consider the case when after some direct or indirect observation the diagnostician establishes that c_2 is true. The probabilities assigned to c_1 and c_2 by program

$$P_7 = P_6 \cup \{obs(c_2)\}$$

are now 0.05 and 1 respectively. The latter observation is an example of a probabilistic phenomena called "explaining away" [20, 23]: when you have competing possible causes for some event, and the chances of one of those causes increases, the chances of the other causes must decline since they are being "explained away" by the first explanation. Thirty years ago, the ability of probabilistic reasoning to naturally exhibit explaining away phenomena served as a substantial argument against modeling knowledge with rule-based systems which did not have this property. We believe explaining away is an important phenomena and that languages combining logical and probabilistic reasoning should allow its formalization.

It is worth noting that, even though we were not able to do that in original P-log, we could do it in its extension [14] based on CR-Prolog [6] – extension of ASP by so called consistency-restoring rules. Consider program P_6' obtained from P_6 by replacing R with the rules:

$$R_2 = \begin{cases} random(c_1) \leftarrow include\_causes \\ random(c_2) \leftarrow include\_causes \end{cases}$$

adding a consistency-restoring rule:

$$include\_causes \stackrel{+}{\leftarrow}$$

and replacing each of the defaults

$$\neg c_1 \leftarrow not\ obs(s)$$
$$\neg c_2 \leftarrow not\ obs(s)$$

with classical closed worlds assumptions for c_1 and c_2:

$$\neg c_1 \leftarrow not\ c_1$$
$$\neg c_2 \leftarrow not\ c_2$$

It is easy to check that P_6' is logically and probabilistically equivalent to P_6. Similarly, we can obtain a CR-Prolog program equivalent to P_7.

This, however, requires the programmer to learn the semantics of CR-Prolog. Moreover, currently there is no reasoning system that implements P-log with consistency-restoring rules, and the task of developing and efficiently implementing such a system seems to be non-trivial. In contrast, implementing P-log with rules containing actions and observations in their bodies seems to be less daunting.

We believe that the above examples show a fairly seamless combination of logical and probabilistic reasoning in search of causal explanations of a symptom.

The next example shows the use of observations in the body of rules to prevent or to allow observations of undefined value of an attribute term $f(\bar{x})$. For the purpose of this discussion we will say that $f(\bar{x})$ is undefined in program Π if Π has no rule containing an occurrence of $f(\bar{x})$ in the head.

Example 5 [Observing unknown values of attributes]
Consider a program $R_0 = \{f : \#boolean\}$. Since the value of f is undefined by the program, a rational agent associated with it has no reason to believe that f is true. Accordingly $P_{R_0}(f) = 0$. Similarly for $\neg f$. It is possible, however, that f is *directly observable*, i.e. the agent has some means of accurately measuring its actual value. This can be represented by a boolean attribute *directly_observable*. Let a program R_1 be obtained from R_0 by adding a rule,

$$f = y \leftarrow directly\_observable(f), obs(f = y)$$

which guarantees correctness of the observation of the value of f. Now suppose, that f was observed to be true. To record this information one will naturally expand R_1 by $obs(f = true)$. The resulting program, R:

```
f : #boolean.
f = Y :- directly_observable(f),
         obs(f=Y).
directly_observable(f).
obs(f=true).
```

is consistent and allows us to conclude that probability of f after the observation is 1. It is easy to see that the second rule of the program cannot be omitted. Without it, the program will violate reality check axiom 3 and will be inconsistent.

The next example demonstrate the use of rules with a deliberate action *do* in the bodies.

Example 6 [Defining attributes in terms of deliberate actions]
Some software systems have strict authorization procedures which restrict the ability of people and devices to interfere with random experiments generating values for one or more of their random attributes. In such cases any such interference can

cause an alarm. This can be achieved by having a boolean attribute *alarm* (possible connected with some physical device), a list of agents authorized to interfere with random attribute $f(\bar{x})$ and rule

$$alarm \leftarrow do(A, f(\bar{x})), \textbf{not} \ \ authorized(A, f(\bar{x})).$$

4 DYNAMICALLY CAUSALLY ORDERED P-LOG PROGRAMS

In this section we introduce a class $\mathcal{B}$ of P-log programs and show that every program from $\mathcal{B}$ is coherent, i.e., defines the collection of possible worlds corresponding to possible sets of beliefs of a rational agent associated with it, and probability function numerically specifying the strength of the agent's beliefs. It is not difficult to see that even a program satisfying Conditions 1–3 from Section 2 may be incoherent. This can be caused by a well known logical inconsistency of the underlying ASP program, by allowing non-determinism of attributes not declared as random, and by some other factors.

The precise notion of coherency, introduced in [8], is given by the following:

Definition 1 (Program Coherency)
A P-log program Π is called *coherent* if

1. Π is *consistent*, i.e. has at least one possible world.
2. Let Π' be obtained from Π by removing all observations and actions. Then
   (a) probability function $P_{\Pi'}$ is defined.
   (b) for every selection rule

$$random(f(\bar{x}), p) \leftarrow K$$

and every probability atom

$$pr(f(\bar{x}) = y \mid_c B) = v$$

of Π, if $P_{\Pi'}(B \cup K) \neq 0$ then

$$P_{\Pi' \cup obs(B) \cup obs(K)}(f(\bar{x}) = y) = v.$$

□

Conditions (1) and (2a) are self-explanatory. Condition (2b) insures that causal probabilities, given by pr-atoms of program Π, agree with corresponding conditional probabilities defined by its a priori part Π'.

Example 7 (Incoherent Programs)
Clearly, a program containing facts p and $\neg p$ is incoherent logically, and has no possible worlds. Probabilitic incoherency is illustrated by program P_8:

```
a,b: #boolean.
random(a).
pr(a) = 0.3.
b:- not -b, a.
-b :- not b, a.
```

It is easy to see that the program has three possible worlds with non-normalized probabilistic measures 0.3, 0.3, and 0.7, and hence the probability $P_{P_8}(a) = 6/13$ which is different from the causal probability 0.3 given by the pr-atom of the program. Intuitively, the program is incoherent because a non-deterministic attribute b is not declared as random. If we were to replace the last two rules by $random(b) \leftarrow a$, the program would regain coherency.

Even though Definition 1 captures the intuitive notion of coherency it is not always easy to use in practice. So, to facilitate the process of writing meaningful programs and proving their coherency, [8] introduces the notions of *causally ordered* and *unitary* programs which are comparatively easy to check and shows that every program satisfying these two conditions is coherent.

A causally ordered program Π allows an ordering, $a_1, \ldots, a_k$ of attributes of Π based on a simple dependency relation, defined as follows: (a) Attribute term a_2 "immediately depends" on a_1 if it occurs in the head of a rule or pr-atom whose body contains an occurrence of a_2. (Note, that B is the body of a pr-atom $pr(f = y|_c B)$ and hence f depends on any attribute term occurring in B). (b) Dependency is defined as the reflexive, transitive closure of this relation. For causally ordered programs the ordering must be strict on random attributes.

If such an ordering is given, then the possible worlds of Π can be constructed gradually from possible worlds of programs $\Pi_0, \Pi_1, \ldots, \Pi_n$ such that $\Pi_n = \Pi$, $\Pi_i \subseteq \Pi_{i+1}$, and each Π_i where $i > 0$, corresponds to a random attribute a_i. Each Π_i is associated with the language, L_i, of attribute terms used to form its rules. L_0 consists of all attribute terms not dependent on any random attribute of Π, L_1 consists of attribute terms not dependent on any random attribute term except (possibly) a_1, etc. (In what follows we often abuse the terminology and refer to the set L of attribute terms occurring in program Π as the *signature* of Π.) To ensure that non-determinism is only possible for random attributes the non-random base, Π_0 of Π is required to have exactly one possible world.

The construction of possible worlds of Π starts with the possible world W_0 of Π_0 and proceeds recursively by considering a possible world W of Π_i with a random selection rule for a_{i+1} whose body is satisfied in W, and building new possible worlds of Π_{i+1} corresponding to possible values of a_{i+1}. For a program to be causally ordered the possible outcomes of that selection shall not be constrained by logical rules or other random selections. The precise definition of causally ordered programs can be found in [8]. We hope that it can be sufficiently illustrated by the following example.

Example 8 (Causally Ordered Programs)
Consider program, P_9:

```
a, b, c, d, f : #boolean.
f.
random(a) :- not b, f.
pr(a) = 0.3.
c :- a.
d :- -a.
```

The possible worlds of the program can be obtained by first considering $L_0 = \{b, f\}$ consisting of attribute terms not depending on any random attributes of P_9 and

program $\Pi_0 = \{f.\}$ whose rules contain only literals that are formed by terms of L_0. Clearly, $W_0 = \{f\}$. Next, we consider $L_1 = L_0 \cup \{a, c, d\}$ (which, in this case, consists of all attribute terms of the program), the corresponding Π_1 which is equal to P_9, and random selection rule,

```
random(a) :- not b, f
```

whose body is satisfied by W_0. There are two possible outcomes for a, and two possible worlds $\{f, a, c\}$ and $\{f, \neg a, d\}$ corresponding to these outcomes. Since non-determinism comes only from the selection rule and no outcomes are constrained, the program is causally ordered. If P_9 were expanded by

```
e :- a. e :- not e
```

then outcome $a = true$ would become impossible, and the program would not be causally ordered. It is also easy to see that program P_8 is not causally ordered since the non-determinism of b is not directly caused by any random selection rule.

It is easier to describe the notion of unitary program which simply requires *the sum of probabilities assigned by pr-atoms for the outcomes of a random experiment not to exceed 1.* (If every value in the range of an attribute has its probability assigned by the pr-atoms, the sum must be exactly 1.) For example, in P_9, we have pr-atom for value a but not $\neg a$. Since $pr(a) = 0.3$ not exceeding 1, the program is unitary. Note, that addition of another *pr*-atom,

```
pr(-a) = 0.2.
```

would make the program non-unitary.

Since P_9 is both causally ordered and unitary, the theorem from [8] establishes its coherency.

Even though many useful P-log programs which can be found in the literature are causally ordered, there are simple and important coherent programs which do not belong to this class. Consider, for instance, the following example (more realistic examples will be given at the end of this section).

Example 9 (Coherent program which is not causally ordered.)
Consider program P_{10}

```
a, b, c, d, f : #boolean.
f.
random(a) :- not b, f.
random(b) :- a, not f.
pr(a) = 0.3.
c :- a.
d :- -a.
```

obtained from P_9 by adding the rule

```
(*) random(b) :- a, not f.
```

Since, according to the original definition of dependency, a and b depend on each other there is no ordering of random attributes which would allow to view P_{10} as the result of gradual construction of programs Π_0, Π_1, ..., corresponding to this ordering. Hence, P_{10} is not causally ordered. However, it is easy to check that P_{10} is coherent. It has possible worlds $W_0 = \{f, a, c\}$ and $W_1 = \{f, \neg a, d\}$ and probability $P_{P_{10}}(a) = 0.3$ which coincides with that of P_{10}. This, of course, can be immediately seen from the fact that, since $f \in P_{10}$, the newly added rule is useless and can therefore be removed from the program without changing its meaning. Moreover, this fact can be discovered in the process of gradual computation of possible worlds of P_{10}. We should start with computing a possible world of a program $\Pi_0 = \{f\}$ of signature $L_0 = \{f\}$. Since f belongs to all possible worlds of P_{10} the uselessness of rule (*) becomes apparent and the rule can be removed.

In what follows we define the class $\mathcal{B}$ of dynamically causally ordered unitaery programs which include program P_{10} from Example 9. The main idea is to replace causal ordering of attributes of the program by an ordering based on the dependency relation between attributes which ignores useless rules. To give a precise definition of such rules we need to generalize the satisfiability relation between an interpretation I over signature L and a set B of e-literals. The definition of satisfiability from section 2 only considers B consisting of e-literals formed by attributes from L (symbolically, $B \subseteq$ e-lit(L)). In what follows we allow B contain e-literals over some other signature.

Definition 2 (Extended Satisfiability Relation)

1. Interpretation I of signature L *satisfies* a set B of e-literals (not necessarily over L) if $B \subseteq$ e-lit(L) and every e-literal from B is satisfied by I,
2. I *falsifies* B if at least one e-literal from B belongs to e-lit(L) and is not satisfied by I.

As an example, consider a set $B = \{a, \mathbf{not}\ b\}$ and interpetation $I_0 = \{a\}$ of $L_0 = \{a\}$. Since $(\mathbf{not}\ b) \notin$ e-lit(L_0), I_0 does not satisfy B. However, interpretation $I_1 = \{a\}$ of $L_1 = \{a, b\}$ satisfies B. Similarly, interpretation $I_2 = \{a, b\}$ of $L_1 = \{a, b\}$, falsifies B. (It is important to notice the difference between the terms "falsify" and "does not satisfy" – B is not satisfied by I_0, but is not falsified by it.) Of course, if B consists of e-literals over L_0 then the new definition coincides with the old one.

We'll also need some notation. By $L_{base}(\Pi)$ we denote the set of attribute terms of program Π which do not depend on any of its random attribute terms; $R_{base}(\Pi)$ denotes the collection of rules of Π whose attribute terms belong to L_{base}. (Whenever possible we omit the parameters of L_{base} and R_{base}.) Program with signature $L_{base}(\Pi)$ and rules $R_{base}(\Pi)$ is called the *base* of Π.

To eliminate useless rule of Π we introduce the following notion:

Definition 3 (Reduct)

Reduct, *red*, is a partial function from programs to programs such that

– $red(\Pi)$ is defined iff the base of Π has exactly one possible world (denoted by W_{base}),

– $red(\Pi)$ is the program obtained from Π by removing all pr-atoms and rules whose bodies are falsified by W_{base}.

□

Example 10 [Reduct]
Consider program P_{10} from example 9. It is easy to see that $L_{base}(P_{10}) = \{f\}$ and $R_{base}(P_{10}) = \{f.\}$. The base has exactly one possible world $W_{base} = \{f\}$, and hence, $red(P_{10})$ is defined

Clearly, *red* eliminates the useless third rule of P_{10}. The body $\{a, not\ f\}$ is falsified by W_{base} (which is, of course, an interpretation of L_{base}). Hence, $red(P_{10})$ includes every rule of P_{10} except the third one. (One may note that, as expected, $red(P_{10}) = P_9$.)

For the following program P

```
a, b: #boolean.
a :- not b.
b :- not a.
```

$L_{base}(P) = \{a, b\}$, and the base of the program includes all the rules of P. Since the base has two possible worlds, $red(P)$ is not defined.

Intuitively, the new ordering, based on the notion of simplification captured by function red[7] is defined in several steps. We start with a strict ordering α of random attributes of Π (often referred to as *probabilistic leveling*), and use the dependency relation to extend it to (not necessarily strict) ordering of all atributes of Π (referred to as *total leveling*). (Note that the original probabilistic ordering of the program does not necessarily reflect the dependency relation between random attributes.) In a manner similar to that in the definition of causally ordered programs this total leveling defines layers $\Pi_0, \ldots, \Pi_n$ of Π, referred to as *dynamic structure of Π induced by α*. If α is such that possible worlds of Π can be gradually constructed from the layers of the dynamic structure induced by α then Π is dynamically causally ordered via α. This intuition is captured by the following definitions.

Definition 4 (Total Leveling)
Let Π be a program such that $red(\Pi)$ is defined, and α be a probabilistic leveling of the random attributes of Π. We expand α to total leveling $|\ |$ as follows: (a) For a random attribute a, $|a| = \alpha(a)$, and (b) for a non-random attribute term a,

1. If a is of the form $random(b, p)$, then $|random(b, p)| = |b|$.
2. Otherwise:
   (a) $|a| = 0$ iff a does not depend on any random attribute term of Π in $red(\Pi)$.
   (b) $|a| = i$ iff i is the level of the random attribute b of Π such that
      i. a depends on b in $red(\Pi)$ and

[7] Note that we consider this specific simplification which is used in the corresponding algorithm and implementation. It can be easily generalized by removing other useless rules, e.g. rules whose bodies contain contrary literals, etc., but it is not clear that implementation of such a generalization will improve the algorithm efficiency.

ii. there is no random attribute c with level j such that $j > i$ and a depends on c in $red(\Pi)$.

We will say that total leveling $|\ |$ is *determined* by probabilistic leveling α.
Total levelings are extended to the e-literals of Σ as follows:

$$|not\ a = y| = |not\ a \neq y| = |a = y| = |a \neq y| = |a|$$

□

Example 11 (Total Leveling)
Consider a probabilistic leveling $|\ |$ of the random attributes of P_{10} such that $|b| = 1, |a| = 2$. In example 10 we showed that $red(P_{10})$ (and therefore total leveling determined by $|\ |$) is defined. It looks as follows. Since f does not depend on a or b, its level is 0. $random(a)$ has the same level as a, i.e., 2. Similarly, $|random(b)| = 1$. Since c, d depend on both b and a in $red(P_{10})$, their level is 2, the larger one of that of a and b. This leveling is expanded to e-literals as follows: $|not\ b| = |b| = 1$, etc.

Now we are ready to define the layers of Π defined by an ordering of random attribute terms.

Definition 5 (Dynamic Structure)
Let Π be a program such that $red(\Pi)$ is defined and $|\ |$ be the total ordering determined by a probabilistic leveling $a_1, \ldots, a_k$ of the random attributes of Π.

We say that $\langle L_0, \Pi_0\rangle, \ldots, \langle L_k, \Pi_k\rangle$ is the *dynamic structure* of Π *induced by* $a_1, \ldots, a_k$ if for every $0 \leq i \leq k$,

1. L_i consists of all attribute terms of Π whose levels are i or less in $|\ |$,
2. for every rule or pr-atom r of Π, Π_i contains r iff all the literals occurring in r are formed by attribute terms of L_i.

Each pair $\langle L_i, \Pi_i\rangle$ $(i \in 0..k)$ is called a *layer*. When convenient, we drop the signatures and simply write $\Pi_0, \ldots, \Pi_n$.

Example 12 (Dynamic Structure)
Consider program P_{10} and probabilistic leveling $|b| = 1$ and $|a| = 2$ from example 11. We have already shown that $red(P_{10})$ and the corresponding total leveling are defined. The corresponding dynamic structure consists of three layers shown in the table below.

Table 1: The Dynamic Structure of P_{10} induced by $|b| = 1$ and $|a| = 2$

$L_0 = \{f\}$	Π_0 is
	`f.`
$L_1 = \{f, b, random(b)\}$	Π_1 is
	`f.`
$L_2 = \{f, b, random(b)$ $a, random(a), c, d\}$	Π_2 is

	`f.`
	`random(a) :- not b, f.`
	`random(b) :- a, not f.`
	`pr(a) = 0.3.`
	`c :- a.`
	`d :- -a.`

Now we are ready to give the definition of a dynamically causally ordered (dco) program. The definition will consist of three parts. We will first define dco programs via a given probabilistic leveling. We next define dco programs not containing activity records. Finally, we will define arbitrary dco programs.

Definition 6 (DCO via Probabilistic Leveling)
Let Π be a program not containing activity records.

Π is *dynamically causally ordered (dco) via a probabilistic leveling* $a_1, \ldots, a_k$ of the random attributes of Π if

1. $red(\Pi)$ is defined,
2. the dynamic structure $\langle L_0, \Pi_0\rangle, \ldots, \langle L_k, \Pi_k\rangle$ induced by $a_1, \ldots, a_k$ satisfies conditions
   (a) $\langle L_0, \Pi_0\rangle$ has a unique possible world and
   (b) for every $i \in \{1..k\}$, if W_{i-1} is a possible world of program $\langle L_{i-1}, \Pi_{i-1}\rangle$ then
      i. if r is a rule or a pr-atom of Π with a_i in the head, and r is not a ground instance of a general axiom of Π then the body of r is either falsified or satisfied by W_{i-1},
      ii. if r is a random selection rule, $random(a_i : p) \leftarrow B$, and W_{i-1} satisfies B then
         – there is $y \in range(a_i)$ such that $p(y) \in W_{i-1}$,
         – for every $y \in range(a_i)$, $p(y) \in atoms(L_{i-1})$ and, if $p(y) \in W_{i-1}$ then the program $W_{i-1} \cup \Pi_i \cup \{\leftarrow not\ a_i = y\}$ has exactly one possible world,
      iii. If W_{i-1} falsifies bodies of all random selection rules with a_i in the head then $W_{i-1} \cup \Pi_i$ has exactly one possible world.

□

Example 13 (DCO Program via a Probabilistic Leveling)
Consider program P_{10} and its dynamic structure induced by probabilistic leveling $|b| = 1, |a| = 2$ from example 12. We will verify that P_{10} is dynamically causally ordered via $|b| = 1, |a| = 2$.

The first condition, existence of $red(P_{10})$, was established earlier. Consider dynamic structure from table 1.

The first level $\langle L_0, \Pi_0\rangle$, has a unique possible world $W_0 = \{f\}$ - condition (2a) is satisfied.

Let us now check (2b) for level $i = 1$, i.e. for random attribute $a_1 = b$. The rule The only rule with random attribute $a_1 = b$ in the head mentioned in condition (i) of (2b) is the third rule of P_{10}. Clearly, it is falsified by W_0. Condition (ii) for $a_1 = b$ is true vacuously. Condition (iii) for $a_1 = b$ is satisfied since $W_0 \cup \Pi_1$ does have exactly one possible world $W_1 = W_0$. To check condition (2b) for $a_2 = a$, consider all rules with random attribute a in the head. The only one such rule is the second rule of P_{10}. Its body is $\{not\ b, f\}$, it is satisfied by W_1 which proves condition (i) of (2b) for $a_2 = a$. To establish (ii) of (2b) for $a_2 = a$ let us first recall that, by definition of interpretation, every interpretation contains $s(y)$ where s is a sort and $y \in s$. Hence, $boolean(true) \in W_{i-1}$ which satisfies the first clause of (ii) for a. To establish the second clause of (ii) it is sufficient to notice that each of the programs $W_1 \cup \Pi_2 \cup \{\leftarrow not\ a = true\}$ and $W_1 \cup \Pi_2 \cup \{\leftarrow not\ a = false\}$ has

exactly one possible world. Condition (iii) for $a_2 = a$ is true vacuously. Therefore, P_{10} is dynamically causally ordered via the given probabilistic leveling.

Definition 7 (Dynamically Causally Ordered Programs - I)
Let Π be a program not containing activity records. Π is *dynamically causally ordered* if Π is dynamically causally ordered via some probabilistic leveling of the random attributes of Π. The probabilistic leveing is called a *dynamic causal probabilistic leveling of* Π. □

In the example 13, P_{10} is dynamically causally ordered because it is so via the probabilistic leveling $|b| = 1, |a| = 2$. The probabilistic leveling is a dynamic causal probabilistic leveling of P_{10}.

Definition 8 (Dynamically Causally Ordered Programs - II)
Let Π be an arbitrary program, and Π' be the program obtained from Π by removing all activity records. Π is *dynamically causally ordered* if Π' is dynamically causally ordered. A *dynamic causal probabilistic leveling of* Π is a dynamic causal probabilistic leveling of Π'.

□

We have completed the definition of class $\mathcal{B}$ of dynamically causally ordered unitary programs.

Theorem 1 *[Main Result]*
Every P-log program from class $\mathcal{B}$ is coherent.

The proof of Theorem 1 is given in appendix 6. An important intermediate result of the proof is a formulation of splitting set theorem for P-log, originally introduced in [19] for logic programs, is given in subsection 6 of the proof.

5 Examples

In this section, we present examples that show the difference between the class $\mathcal{B}$ of dynamically causally ordered and the class of unitary causally ordered programs. First three examples are rather natural P-log programs from class $\mathcal{B}$ which are not causally ordered. The last example is causally ordered by not dynamically causally ordered.

5.1 Die

A program for the following example is dynamically causally ordered but not causally ordered.

We throw a die until we get outcome 1 or make 5 throws. What's the probability that we will make 5 throws?

A very natural P-log representation of the story is given below:

```
% Sorts
#outcome = 1..6.
#step = 1..5.


% Attributes
throw : #step -> #outcome.
made_5th_throw: #boolean.

% Rules

% the outcome of the die at step 1 is random
random(throw(1)).

% if the value of the die at the previous step, T2, was not 1,
% then the outcome of the die at current step, T, is random
random(throw(T)) :- throw(T2) != 1, T = T2+1.

% the fifth throw was made if the die takes some value, X, at step 5
made_5th_throw :- throw(5) = X.
```

Let us denote the set of all ground instances of this program by P^d. It is not difficult to check that $P_{P^d}(made\_5th\_throw) = (5/6)^4$ which answers the problem's question.

Note, however, that P^d contains rules like

```
random(throw(2)) :- throw(3) != 1, 2 = 3
```

hence $throw(2)$ and $throw(3)$ depend on each other, and P^d is not causally ordered. Fortunately, one can easily verify that P^d is dynamically causally ordered via probabilistic leveling $throw(1), \ldots, throw(5)$. It is easy to see that the base of the program is empty, hence $W_{base} = \emptyset$. Recall that every possible world contains standard interpretation of arithmetic terms which are not displayed in our notation. Hence, W_{base} falsifies $2 = 3$ and eliminates the above rule as well as all other similar rules of P^d. It is not difficult to check that conditions of the definition 6 are satisfied and, hence, the program belongs to our class $\mathcal{B}$.

5.2 Random Tree

Consider a tree defined by a collection of facts of the form $arc(X, Y)$ where Y is the parent of X. Each node of the tree is assigned a value. If a node is a leaf, the assigned value is selected (uniformly) at random from $\{1, 2, 3, 4, 5, 6\}$. If a node is not a leaf, its value is selected randomly from the values of the node's children.

The natural program representing the story is:

```
% Sorts
#node = {1,2,3,4,5}.
#value = {1,2,3,4,5,6}.
```

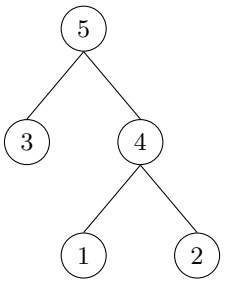
```
% Attributes
arc: #node, #node -> #boolean.
value_of : #node -> #value.
possible_value: #value, #node -> #boolean.
leaf: #node -> #boolean.

% Rules

% Tree arcs. arc(i,j) means there is an arc from i to j
arc(4,5).
arc(3,5).
arc(2,4).
arc(1,4).

% Definition of leaves:

% Node X not a leaf if there is a directed arc with the end in X
leaf(X) = false :- arc(Y,X).

% Otherwise, X is a leaf.
leaf(X) = true :- not leaf(X) = false.

% Random selections:

% Every leaf node takes a value at random
random(value_of(N)) :- leaf(N).

% Every non-leaf node X takes a value from the set of possible
% values {X:possible_value(X,N)}
random(value_of(N):{X:possible_value(X,N)}) :- -leaf(N).

% Value N is possible in non-leaf node X if it a value
% of its child
possible_value(X,N) :- arc(N1,N), value_of(N1) = X.
```

Let P^t denote the set of ground instances of the rules of the program. Its random attribute terms are: $value\_of(1), \ldots, value\_of(5)$. P^t contains rules

```
random(value_of(1):{X:possible_value(X,1)}) :- -leaf(1).
possible_value(3,1) :- arc(4,1), value_of(4) = 3.
```

Clearly, rules obtained from the two rules above by replacing 1 by 4 and 4 by 1 also belong to P^t. Hence, $value\_of(1)$ and $value\_of(4)$ are dependent on each other. Hence, the program is not causally ordered.

To show that P^t is dynamically causally ordered we need to find proper probabilistic leveling of its attribute terms. To do that consider the tree described by the program:



Intuitively, the values of its leaves do not "depend" on each other or any other random attributes. Hence $value\_of(1)$, $value\_of(2)$ and $value\_of(3)$ can be ordered arbitrarily, say $|value\_of(1)| = 1$, $|value\_of(2)| = 2$ and $|value\_of(3)| = 3$. Since $value\_of(4)$ "depends" only on $value\_of(1)$, and $value\_of(2)$, its level should be greater than $value\_of(1)$ and $value\_of(2)$. Since $|value\_of(3)| = 3$, we can have $|value\_of(4)| = 4$. Since $value\_of(5)$ "depends" on $value\_of(3)$ and $value\_of(4)$, we have $|value\_of(5)| = 5$.

Let us now show that P^t is dynamically causally ordered via this leveling $|\ |$. Clearly, $L_{base}(P^t)$ is $\{arc(X,Y) : X,Y \in \#node\} \cup \{leaf(X) : X \in \#node\}$ and the base of P^t is

```
arc(4,5).
arc(3,5).
arc(2,4).
arc(1,4).
leaf(X) = false :- arc(Y,X).
leaf(X) = true :- not leaf(X) = false.
```

which has the unique possible world W_{base}:
$\{arc(4,5), arc(3,5), arc(2,4), arc(1,4), leaf(1), leaf(2), leaf(3), \neg leaf(4), \neg leaf(5)\}$.
Hence, $red(P^t)$ is defined.

Let $\langle L_0, \Pi_0\rangle, \ldots, \langle L_5, \Pi_5\rangle$ be the dynamic structure induced by $|\ |$. It is easy to see that $L_0 = L_{base}(P^t)$ and hence $W_0 = W_{base}$ is the unique possible world of $\langle L_0, \Pi_0\rangle$. To show that the structure satisfies condition (2b) of definition 6 we start with level $i = 1$ and random attribute $value\_of(1)$. The rules with $value\_of(1)$ in their heads are

```
random(value_of(1)) :- leaf(1).
random(value_of(1):{X:possible_value(X,1)}) :- -leaf(1).
possible_value(X,1) :- arc(N1,1), value_of(N1) = X.
```

The first rule is satisfied by W_0, the second falsified by W_0, and instance of the third rule for any $X \in 1..6, N1 \in 1..5$ is falsified by W_0 because $arc(N1,1)$ is not satisfied by W_0. Hence, condition (i) of the definition 6 of DCO via probabilistic leveling is satisfied. For the first rule, for every value y from the range of $value\_of(1)$, $W_0 \cup \Pi_1 \cup \{\leftarrow not\ value\_of(1) = y\}$ has exactly one possible world. Hence condition (ii) of definition 6 holds. Condition (iii) of definition 6 is true vacuously.

Similarly, we can show that for any level $i \in 2..5$, conditions (i) to (iii) of definition 6 hold. Hence P^t is dynamically causally ordered via probabilistic leveling $|\ |$.

5.3 Blood Type Problem

This is a typical blood type problem frequently used in probability and statistics classes. The particular problem description is based on section 4.1.3 in [25].

The ABO blood group system distinguishes four types of bloods: ***A, B, AB*** *and* ***O****. The type of blood of each individual is determined by two genes inherited from his/her parents (one gene is inherited from each parent). The pair of genes is also called a genotype. There are three types of genes:* ***a****,* ***b*** *and* ***o****, and 6 corresponding genotypes:* ***ao,bo,ab,aa,bb,oo****. The genotypes* ***ao, bo, ab, aa, bb, oo*** *are distributed in generation* 1 *with probabilities 0.24, 0.24, 0.18, 0.09, 0.09, 0.16 correspondingly. The corresponding blood type of a person for each combination of inherited genes (which determines his/her genotype) is given in Table 2.*

Table 2: ABO blood group system

Mother's gene \ Father's gene	**a**	**b**	**o**
a	**A**	**AB**	**A**
b	**AB**	**B**	**B**
o	**A**	**B**	**O**

If an individual A has genes of types X and Y, and an individual B has genes of types F and H, their child will have one of the pairs of genes (X,F), (Y,F), (X,H), (Y,H); where each pair is inherited with probability 0.25.

Here is a natural P-log program representing the story for three people: Mary, Todd, and John.

```
%% Blood Type Problem
%% Sorts
#person={mary, todd, john}.
#gene ={g_a,g_b,g_o}.
#genotype = {g(g_a, g_b), g(g_a, g_o), g(g_b,g_o),
             g(g_a, g_a), g(g_b, g_b), g(g_o,g_o)}.
#bloodtype={b_a,b_b,b_o,b_ab}.
#generation = {1,2}.

%% Attributes
genotype_of: #person -> #genotype.
bloodtype_of: #person -> #bloodtype.
mother_of: #person -> #person.
father_of: #person -> #person.
generation_of: #person -> #generation.
possible_combination: #genotype, #genotype, #genotype -> #boolean.
belongs_to: #gene, #genotype -> #boolean.

%% Rules
% generations
```

```
generation_of(john) = 2.
generation_of(mary) = 1.
generation_of(todd) = 1.

% family tree
mother_of(john)=mary.
father_of(john)=todd.


% blood_type(X)=G : the blood type of person X
% determined by the genes he or she inherits from parents
% as described in table 1

bloodtype_of(X)=b_a :- genotype_of(X) = g(g_a,Y), Y!=g_b.
bloodtype_of(X)=b_b :- genotype_of(X) = g(g_b,Y), Y!=g_a.
bloodtype_of(X)=b_ab :- genotype_of(X) = g(g_a,g_b).
bloodtype_of(X)=b_o :- genotype_of(X) = g(g_o,g_o).

% the genotypes of the parents of a person X in the old generation are
% distributed as it is given in the problem statement

random(genotype_of(X)):- generation_of(X) = 1.

pr(genotype_of(X) = g(g_a,g_o)|generation(X) = 1) = 24/100.
pr(genotype_of(X) = g(g_b,g_o)|generation(X) = 1) = 24/100.
pr(genotype_of(X) = g(g_a,g_b)|generation(X) = 1) = 18/100.
pr(genotype_of(X) = g(g_a,g_a)|generation(X) = 1) = 9/100.
pr(genotype_of(X) = g(g_b,g_b)|generation(X) = 1) = 9/100.
pr(genotype_of(X) = g(g_o,g_o)|generation(X) = 1) = 16/100.


% the genotypes of a person in the new generation are randomly
% inherited from his/her parents

random(genotype_of(P):{G:possible_genotype(P,G)}) :-
                                        generation_of(P) = 2.

possible_genotype(P,G) :- father_of(P)=F,
                          mother_of(P)=M,
                          genotype_of(F) = U,
                          genotype_of(M) = V,
                          possible_combination(G,U,V).



% possible_combination(G,U,V) is true if G can be the genotype of a
% child whose parents have genotypes U and V

possible_combination(g(G1,G2),U,V) :- belongs_to(G1,U),
```

```
                                        belongs_to(G2,V).
possible_combination(g(G1,G2),U,V) :- belongs_to(G2,U),
                                        belongs_to(G1,V).

% belongs_to(G,GT) is true if gene G belongs to the pair of genes in
% genotype GT
belongs_to(G,g(G,X)).
belongs_to(G,g(X,G)).
```

Let P^b denote the set of ground instances of the rules of the program. The random attribute terms are those related to genotypes: $genotype\_of(mary)$, $genotype\_of(todd)$ and $genotype\_of(john)$. P^b contains the following rules

```
random(genotype_of(mary):{G:possible_genotype(mary,G)}) :-
       generation_of(mary) = 2
```

and

```
possible_genotype(mary,g(g_a, g_b)) :-
       father(mary) = john,
       mother(mary) = todd,
       genotype_of(john) = g(g_a, g_b),
       genotype_of(todd) = g(g_a, g_b),
       possible_combination(g(g_a, g_b),g(g_a, g_b),g(g_a, g_b)).
```

Clearly, rules obtained from the two rules above by replacing $mary$ by $john$ and $john$ by $mary$ also belong to P^b. Hence, $genotype\_of(mary)$ and $genotype\_of(john)$ are dependent on each other. Hence, the program is not causally ordered.

However, one can verify that P^b is dynamically causally ordered via probabilistic leveling $|\ |$:

$|genotype\_of(mary)| = 1$,
$|genotype\_of(todd)| = 2$,
$|genotype\_of(john)| = 3$.

It is easy to check that $red(P^b)$ is defined. Let $\langle L_0, \Pi_0\rangle, \ldots, \langle L_3, \Pi_3\rangle$ be the dynamic structure induced by $|\ |$. Π_0 consists of

```
generation(mary)=1.
generation(todd)=1.
generation(john)=2.
mother_of(john)=mary.
father_of(john)=todd.
```

Π_0 has a unique possible world W_0 consisting of the atoms above. Thus, we have condition (2a). To prove condition (2b) for level 1, note that the rules with $genotype\_of(mary)$ in their heads are

```
random(genotype_of(X)):- generation_of(X) = 1.
random(genotype_of(mary):{G:possible_genotype(mary,G)}) :-
                                        generation_of(mary) = 2.
```

The body of the first rule is satisfied by W_0 while that of the second rule is falsified. For each value y from the range of $genotype\_of(mary)$, $W_0 \cup \Pi_1 \cup \{\leftarrow not\ genotype\_of(mary) = y\}$ has exactly one possible world. So, for level 1, conditions (i) to (iii) of the definition 6 are satisfied. Similarly, we can verify these conditions for levels 2 and 3. Hence, the program P^b is dynamically causally ordered.

5.4 A Causally Ordered Program which is Not Dynamically Causally Ordered

Consider the program P:

```
a,b,h,x,y:#boolean.
p: #boolean -> #boolean

h.
p(true).
p(false).
random(x:{X:p(X)}).
random(y:{X:p(X)}) :- x.
a:- not b, x.
b:- not a, x.
a :- not h, y.
:- a,y.
:- a,-y.
```

We first illustrate that the program is causally ordered by using the following leveling:
$|h| = |p| = 0$,
$|x| = 1$,
$|y| = |a| = |b| = 2$.
This leveling results in a dynamic structure $\langle L_0, \Pi_0\rangle, \ldots, \langle L_2, \Pi_2\rangle$. To verify that the structure satisfies definition of causally ordered program from [8] we first need to check that Π_0 has exactly one possible world. Obviously, this is $W_0 = \{h, p(true), p(false)\}$. Next we have to show that
$W_0 \cup \Pi_1 \cup \{\leftarrow not\ x = true\}$ and $W_0 \cup \Pi_1 \cup \{\leftarrow not\ x = false\}$
have unique possible worlds W_{11} and W_{12} respectively. Clearly,
$W_{11} = W_0 \cup \{x = true\}$ and $W_{12} = W_0 \cup \{x = false\}$.
Since W_{11} satisfies the body of
"`random(y:X:p(X)) :- x`"
we need to show that
$W_{11} \cup \Pi_2 \cup \{\leftarrow not\ y = true\}$ and $W_{11} \cup \Pi_2 \cup \{\leftarrow not\ y = false\}$
have unique possible worlds W_{21} and W_{22}. Constraints of the program ensure that the rule "`a:- not b, x.` " becomes useless in both programs, and hence
$W_{21} = W_1 \cup \{y, b\}$ and $W_{22} = W_1 \cup \{\neg y, b\}$. Finally, W_{12} does not satisfy the body of the random rule for y. So the last thing to show is that $W_{12} \cup \Pi_2$ has unique possible world. Clearly, it is W_{12}. Thus, program P is causally ordered.

Now we show that the program is not dynamically causally ordered. There are only two different probabilistic levelings:

$|y|_1 = 1$ and $|x|_1 = 2$,
$|y|_2 = 2$ and $|x|_2 = 1$.
In both cases $red(P)$ is defined, Π_0 is the same and has unique possible world W_0, and hence conditions (1) and (2a) of definition 6 are satisfied.

The program will not be dynamically causally ordered via the probabilistic leveling $|y|_1 = 1$ and $|x|_1 = 2$. The fifth rule of P has y in its head, but its body is neither satisfied nor falsified by W_0. This violates condition (i) of definition 6.

Now consider the probabilistic leveling of $|x|_2 = 1$ and $|y|_2 = 2$. Note that the total leveling $|\ |$, used for showing that the program is causally ordered, is not the total leveling determined by $|\ |_2$. The definition of the total leveling expanded from a probabilistic leveling requires that a non-random attribute term must be assigned a level where it depends on the random attribute term at this level. Since a and b do not depend on y, their level cannot be 2 as in $|\ |$.

Since $h \in W_0$ the rule

```
a :- not  h, y
```

is not in $red(P)$. As a result, a and b will depend on x only and thus have the same level as x in the total leveling determined by $|\ |_2$. Let $\langle L_0^2, \Pi_0^2\rangle, \ldots, \langle L_2^2, \Pi_2^2\rangle$ be the dynamic structure induced by the total leveling $|\ |_2$. We will show $P_1 = W_0 \cup \Pi_1^2 \cup \{\leftarrow not\ x = true\}$ does not have a unique possible world. Π_1^2 will contain

```
a:- not b, x.
b:- not a, x.
```

We can verify that P_1 has two possible worlds $W_0 \cup \{x = true, a\}$ and $W_0 \cup \{x = true, b\}$. It violates the condition (ii) of definition 6. Hence, the program is not dynamically causally ordered via the probabilistic leveling $|x|_2 = 1$ and $|y|_2 = 2$.

In summary, the program is not dynamically causally ordered via any probabilistic leveling and thus not dynamically causally ordered.

6 Conclusion

In this paper we

- Refined the syntax and semantics of P-log by eliminating some ambiguities and incidental decisions made in its original version and narrowing the distance between the intuitive meaning of the language constructs and their formal semantics.
- Introduced a simple extension of the language allowing rules with observations and actions in their bodies. Showed how the extension can be used to express new and useful types of reasoning, including reasoning from symptoms to causes which combines logic and probability and accounts for common probabilistic phenomena of "explaining away".
- Defined a new class, $\mathcal{B}$, of dynamically causally ordered unitary programs, and showed that such programs are coherent (i.e., logically and probabilistically consistent). The result facilitates construction of P-log programs and proofs of their coherency. We also gave examples of natural programs from $\mathcal{B}$ not belonging to previously known coherency class of causally ordered unitary programs from [8]. Even though the class $\mathcal{B}$ is broad and contains all examples of P-log

programs we found in the literature, we showed that there are causally ordered and unitary programs which do not belong to this class.

We believe that this work made P-log clearer and more expressive knowledge representation language, and added some new insights into methodology of its use.

There is a query answering algorithm sound for programs from the class $\mathcal{B}$ (see [1]). Prototype implementation of this algorithm was used to run all relevant examples from this paper. Our future plans include expanding the language by the more powerful type system in the style of [4] and by aggregates and other set related constructs from [16,15]. Such an expansion does not require any additional theoretical work but may have a substantial influence on the implementation. This will further improve P-log as a tool for teaching. To make P-log more suitable for large applications one need to improve efficiency of its reasoning systems. Substantial improvements can be achieved by optimizing the algorithm and data structures used in the current implementation. It would also be interesting to investigate other approaches to P-log inference. One can develop approximate inference algorithms (possibly based on Monte-Carlo sampling methods). It is also possible to use inference methods for a recently introduced new formalism $\mathrm{LP}^{\mathrm{MLN}}$ [17]. The results from [18] and [2] would allow us to use inference in P-log for programs in $\mathrm{LP}^{\mathrm{MLN}}$, and vice versa.

Acknowledgements We would like to thank Leroy Mason and Nelson Rushton for useful discussions on subjects related to this paper.

References

1. Balai, E.: Investigating and extending p-log. Ph.D. thesis, Texas Tech University (2017)
2. Balai, E., Gelfond, M.: On the relationship between P-log and $\mathrm{LP}^{\mathrm{mln}}$. In: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016, pp. 915–921 (2016). URL http://www.ijcai.org/Abstract/16/134
3. Balai, E., Gelfond, M.: Refining and generalizing p-log - preliminary report. In: Proceedings of the 10th Workshop on Answer Set Programming and Other Computing Paradigms co-located with the 14th International Conference on Logic Programming and Nonmonotonic Reasoning, ASPOCP@LPNMR 2017, Espoo, Finland, July 3, 2017. (2017). URL http://ceur-ws.org/Vol-1868/p6.pdf
4. Balai, E., Gelfond, M., Zhang, Y.: Towards answer set programming with sorts. In: Logic Programming and Nonmonotonic Reasoning, 12th International Conference, LPNMR 2013, Corunna, Spain, September 15-19, 2013. Proceedings, pp. 135–147 (2013)
5. Balduccini, M.: Answer set solving and non-herbrand functions. In: Proceedings of the 14th International Workshop on Non-Monotonic Reasoning (NMR'2012)(Jun 2012) (2012)
6. Balduccini, M., Gelfond, M.: Logic programs with consistency-restoring rules. In: International Symposium on Logical Formalization of Commonsense Reasoning, AAAI 2003 Spring Symposium Series, pp. 9–18 (2003)
7. Baral, C., Gelfond, M., Rushton, N.: Probabilistic reasoning with answer sets. In: International Conference on Logic Programming and Nonmonotonic Reasoning, pp. 21–33. Springer (2004)
8. Baral, C., Gelfond, M., Rushton, N.: Probabilistic reasoning with answer sets. Theory and Practice of Logic Programming **9**(01), 57–144 (2009)
9. Baral, C., Hunsaker, M.: Using the probabilistic logic programming language p-log for causal and counterfactual reasoning and non-naive conditioning. In: IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007, pp. 243–249 (2007)

10. Dix, J., Gottlob, G., Marek, V.W.: Reducing disjunctive to non-disjunctive semantics by shift-operations. Fundam. Inform. **28**(1-2), 87–100 (1996). DOI 10.3233/FI-1996-281205. URL https://doi.org/10.3233/FI-1996-281205
11. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In: Logic Programming, Proceedings of the Fifth International Conference and Symposium, Seattle, Washington, August 15-19, 1988 (2 Volumes), pp. 1070–1080 (1988)
12. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New Generation Computing **9**(3/4), 365–386 (1991)
13. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New generation computing **9**(3-4), 365–385 (1991)
14. Gelfond, M., Rushton, N.: Causal and probabilistic reasoning in P-log. In: R. Dechter, H. Geffner, J. Halpern (eds.) A tribute to Judea Pearl, pp. 337–359. College Publications (2010)
15. Gelfond, M., Zhang, Y.: Vicious circle principle and logic programs with aggregates. TPLP **14**(4-5), 587–601 (2014). DOI 10.1017/S1471068414000222. URL http://dx.doi.org/10.1017/S1471068414000222
16. Gelfond, M., Zhang, Y.: Vicious circle principle and formation of sets in ASP based languages. In: Logic Programming and Nonmonotonic Reasoning, 14th International Conference, LPNMR (2017)
17. Lee, J., Wang, Y.: Weighted rules under the stable model semantics. In: Principles of Knowledge Representation and Reasoning: Proceedings of the Fifteenth International Conference, KR 2016, Cape Town, South Africa, April 25-29, 2016., pp. 145–154 (2016). URL http://www.aaai.org/ocs/index.php/KR/KR16/paper/view/12901
18. Lee, J., Yang, Z.: LP$^{\mathrm{MLN}}$, weak constraints, and p-log. In: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA., pp. 1170–1177 (2017)
19. Lifschitz, V., Turner, H.: Splitting a logic program. In: Logic Programming, Proceedings of the Eleventh International Conference on Logic Programming, Santa Marherita Ligure, Italy, June 13-18, 1994, pp. 23–37 (1994)
20. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1988)
21. Pearl, J.: Causality: Models, Reasoning, and Inference. Cambridge University Press (2000)
22. Pereira, L.M., Saptawijaya, A.: Programming Machine Ethics, 1st edn. Springer Publishing Company, Incorporated (2016)
23. Wellman, M.P., Henrion, M.: Explaining'explaining away'. IEEE Transactions on Pattern Analysis and Machine Intelligence **15**(3), 287–292 (1993)
24. Zhang, S., Stone, P.: Corpp: Commonsense reasoning and probabilistic planning, as applied to dialog with a mobile robot. In: Proceedings of the 29th Conference on Artificial Intelligence (AAAI) (2015)
25. Zhu, W.: Plog: Its algorithms and applications. Ph.D. thesis, Texas Tech University (2012)

Appendix: Proof of the Main Result

We prove Theorem 1 in 3 steps. In section A.1 we describe a translation τ from P-log programs into ASP programs and show the relationship between the possible worlds of a given P-log program Π and answer sets of its translation $\tau(\Pi)$. Then, in section A.2 we formulate splitting set theorem for P-log originally defined in [19] for Answer Set Prolog programs. Finally, in section A.3 we prove theorem 1 using the results from sections A.1 and A.2. The proof refines many of the results used in [8] to prove the coherency of causally ordered unitary programs in the original P-log language.

A.1 Translation from P-log to ASP

For every P-log program Π, not necessarily containing general axioms, with signature Σ we define an ASP program $\tau(\Pi)$ whose answer sets correspond to possible worlds of Π. More precisely, τ is defined on elements of Π as follows:

1. if $f(\bar{x}) = y$ is a literal of Σ, $\tau(f(\bar{x}) = y)$ is $f(\bar{x}, y)$;
2. if $f(\bar{x}) \neq y$ is a literal of Σ, $\tau(f(\bar{x}) \neq y)$ is $\neg f(\bar{x}, y)$;
3. if r is a rule of Π, $\tau(r)$ is an ASP rule obtained from r by replacing all occurrences of literals in the rule with their translations;
4. if Π is a P-log program with signature Σ, $\tau(\Pi)$ is an ASP program consisting of
   (a) the rules in the set $\{\tau(r) \mid r \text{ is a rule of } \Pi\}$; and
   (b) the rules of the form
   $$\neg f(\bar{x}, y_1) \leftarrow f(\bar{x}, y_2) \tag{14}$$
   for each two atoms $f(\bar{x}) = y_1$ and $f(\bar{x}) = y_2$ of Σ such that $y_1 \neq y_2$;
5. if A is a set of atoms of Σ, then $\tau(A)$ is the set of ASP literals
   $$\{f(\bar{x}, y) \mid f(\bar{x}) = y \in A\} \cup \{\neg f(\bar{x}, y) \mid f(\bar{x}) = y_1 \in A \wedge y_1 \neq y \wedge y \in range(f)\}$$
6. If L is a set of literals of Σ, then $\tau(L)$ is the set of ASP literals:
   $$\tau(\{f(\bar{x}) = y \mid f(\bar{x}) = y \in L\}) \cup \{\tau(f(\bar{x}) \neq y) \mid f(\bar{x}) \neq y \in L\}$$

Lemma 1 If I is an interpretation of Σ, then I satisfies a literal l of Σ if and only if $\tau(I)$ satisfies $\tau(l)$

Proof
$\Rightarrow$

1. if l is of the form $f(\bar{x}) = y$ and I satisfies $f(\bar{x}) = y$, $\tau(I)$ contains an atom $\tau(l) = f(\bar{x}, y)$.
2. If l is of the form $f(\bar{x}) \neq y$, and I satisfies l, by definition of satisfiability there must exists an atom $f(\bar{x}) = y_1$, where $y_1 \neq y$, such that I satisfies $f(\bar{x}) = y_1$. Therefore, from part 5 of the definition of τ, $\tau(I)$ contains $\neg f(\bar{x}, y)$.

$\Leftarrow$

1. if l is of the form $f(\bar{x}, y)$ and $\tau(I)$ satisfies $f(\bar{x}, y)$, then, by construction of $\tau(I)$, we have $f(\bar{x}) = y \in I$.
2. If l is of the form $\neg f(\bar{x}, y)$, and $\tau(I)$ satisfies l, by construction of $\tau(I)$, I must contain an atom $f(\bar{x}) = y_1$ for $y_1 \neq y$. Therefore, by definition of satisfiability, I satisfies $f(\bar{x}) \neq y$.

Lemma 2 Let Π be a P-log program not necessarily containing all general axioms. An interpretation W of Π is a possible world of Π if and only if $\tau(W)$ is an answer set of $\tau(\Pi)$

Proof

$\Rightarrow$ Let W be a possible world of Π. We prove that $\tau(W)$ is an answer set of $\tau(\Pi)$.

1) We show that $\tau(W)$ is a consistent set of ASP literals. We prove by contradiction. Suppose $\tau(W)$ is inconsistent. Thus, there exists an ASP atom $f(\bar{x}, y)$ such that $\tau(W)$ contains both $f(\bar{x}, y)$ and $\neg f(\bar{x}, y)$. By definition of $\tau(W)$, it implies that $I(f(\bar{x})) = y$ and there exists $y_1 \neq y$ such that $I(f(\bar{x})) = y_1$. Thus, since I is a mapping by definition, we have a contradiction.
2) We show that $\tau(W)$ satisfies the rules of the reduct $\tau(\Pi)^{\tau(W)}$. By R_a we denote the rules of $\tau(\Pi)$ described in 4.a) of the definition of τ; by R_b we denote rules described in 4.b) of the same definition. Clearly, $\tau(\Pi)^{\tau(W)} = R_a^{\tau(W)} \cup R_b^{\tau(W)}$.
   (a) We show that $\tau(W)$ satisfies the rules of $R_a^{\tau(W)}$. Let r be a rule in $R_a^{\tau(W)}$ such that $\tau(W)$ satisfies the body of r. We prove that $\tau(W)$ satisfies the head of r. From lemma 1, the definition of τ and the definition of R_a, we conclude that there is a rule r' in Π^W such that $r = \tau(r')$. By lemma 1 and the definition of $\tau(W)$, W satisfies the body of r'. Since W is a possible world of Π, W satisfies the head of r'. By lemma 1 and the definition of $\tau(r')$, $\tau(W)$ satisfies the head of r.
   (b) We show that $\tau(W)$ satisfies the rules of $R_b^{\tau(W)}$. Let r be a rule in $R_b^{\tau(W)}$ given below
   $$\neg f(\bar{x}, y_1) \leftarrow f(\bar{x}, y_2)$$
   such that $\tau(W)$ satisfies $f(\bar{x}, y_2)$. We need to show that $\tau(W)$ satisfies $\neg f(\bar{x}, y_1)$.
   By lemma 1, since $\tau(W)$ satisfies $f(\bar{x}, y_2)$, W satisfies $f(\bar{x}) = y_2$, that, by definition of $\tau(W)$, implies that $\tau(W)$ satisfies $\neg f(\bar{x}, y)$.
3) We show that $\tau(W)$ is minimal, that is, there does not exist a set of literals A such that A satisfies the rules of $\tau(\Pi)^{\tau(W)}$ and A is a proper subset of $\tau(W)$. We prove by contradiction. Suppose there is A such that
   $$A \text{ satisfies all the rules in } \tau(\Pi)^{\tau(W)} \tag{15}$$
   and
   $$A \text{ is a proper subset of } \tau(W) \tag{16}$$
   Since $A \subsetneq \tau(W)$, by construction of $\tau(W)$ and the fact that W is an interpretation, A does not contain a pair of atoms $f(\bar{x}, y_1)$, $f(\bar{x}, y_2)$ for $y_1 \neq y_2$. Thus, we can construct interpretation I of Σ such that I maps $f(\bar{x})$ to y if and only if $f(\bar{x}, y)$ belongs to A.
   In a) we show that I satisfies the rules of Π^W. In b) we show that $I \subsetneq W$, thus, obtaining a contradiction (by the definition of possible world, W should be a minimal interpretation satisfying Π^W).
   (a) We prove that I satisfies the rules of Π^W. Let r be a rule of Π^W such that I satisfies the body of r. We need to show that I satisfies the head of r. First we prove that A satisfies the body of $\tau(r)$. Since r belongs to Π^W, r does not contain literals preceded by default negation.
      – Let l be a literal of the form $f(\bar{x}, y)$ belonging to the body of $\tau(r)$. Since I satisfies the body of r, $I(f(\bar{x})) = y$. By construction of I, A satisfies $f(\bar{x}, y)$.
      – Let l be a literal of the form $\neg f(\bar{x}, y)$ belonging to the body of $\tau(r)$. Since I satisfies the body of r, $I(f(\bar{x})) = y_1$, where $y_1 \neq y$. By construction of I, $f(\bar{x}, y_1)$ belongs to A. Since A satisfies the rules of $\tau(\Pi)^{\tau(W)}$,

including the rule

$$\neg f(\bar{x}, y) \leftarrow f(\bar{x}, y_1)$$

Therefore, A satisfies $\neg f(\bar{x}, y)$.

By definition of reduct and from lemma 1 it follows that $\tau(r)$ belongs to $\tau(\Pi)^{\tau(W)}$. Therefore, since A satisfies the body of $\tau(r)$, and A satisfies the rules of $\tau(\Pi)^{\tau(W)}$, it follows that A satisfies the head of $\tau(r)$. Therefore, there exists an ASP literal $f(\bar{x}, y)$ in the head of $\tau(r)$ satisfied by A. By construction of I, I satisfies literal $f(\bar{x}) = y$. By definition of $\tau(r)$, the head of r contains the literal $f(\bar{x}) = y$. Therefore, I satisfies the head of r.

(b) We prove that $I \subsetneq W$. By construction of I, I contains a literal $f(\bar{x}) = y$ if an only if $f(\bar{x}, y) \in A$. By definition of $\tau(W)$, W contains a literal $f(\bar{x}) = y$ if and only if $f(\bar{x}, y) \in \tau(W)$. For a set of ASP literals S, by S^+ we denote the subset of S containing all positive literals of S and by S^- we denote the subset of S containing all negative literals in S (that is, literals of the form $\neg f(\bar{y})$). It is sufficient to show that $A^+ \subsetneq \tau(W)^+$. We prove by contradiction

i. Suppose A^+ is not a proper subset of $\tau(W)^+$

ii. Since A is a proper subset of $\tau(W)$, A^+ is a subset of A and $\tau(W)^+$ is a subset of $\tau(W)$, from i. we have

$$|\tau(W)^+| = |A^+| \tag{17}$$

(and, even more precisely, $\tau(W)^+ = A^+$)

iii. By definition of $\tau(W)$,

$$|\tau(W)^-| = \sum_{f(\bar{x}) \in \{f(\bar{x}) | \exists y : W(f(\bar{x})) = y\}} (|range(f(\bar{x}))| - 1) \tag{18}$$

iv. For each positive ASP literal $f(\bar{x}, y_2)$ in A and for each y_1 in $range(f)$ such that $y_1 \neq y_2$, there is rule (14) in $\tau(\Pi)$. Since A satisfies the rules of $\tau(W)$, in particular, those of the form (14), from (17) and the construction of $\tau(W)$, it follows that the number of negative literals in A is bounded below as follows:

$$|A^-| \geq \sum_{f(\bar{x}) \in \{f(\bar{x}) | \exists y : W(f(\bar{x})) = y\}} (|range(f(\bar{x}))| - 1) \tag{19}$$

v. By combining equation (18) and inequality (19), we get

$$|A^-| \geq |\tau(W)^-| \tag{20}$$

vi. From equation (17) and inequality (20) we have

$$\begin{aligned} |A| &= |A^+| + |A^-| \\ &= |\tau(W)^+| + |A^-| \\ &\geq |\tau(W)^+| + |\tau(W)^-| \\ &= |\tau(W)| \end{aligned} \tag{21}$$

that contradicts our original assumption (16) stating that A is a proper subset $\tau(W)$.

4) From 1)- 3) it follows that $\tau(W)$ is an answer set of $\tau(\Pi)$.

$\Leftarrow$ Let A be an answer set of $\tau(\Pi)$ and I be an interpretation of Π such that $A = \tau(I)$. We prove that I is a possible world of Π. In 5) we show that I satisfies the rules of Π^I and in 6) we show the minimality of I.

5) We prove that I satisfies the rules of Π^I. Let r be a rule of Π^I such that I satisfies the body of r. From lemma 1 and the definitions of reduct in P-log and ASP it follows that the rule $\tau(r)$ belongs to $\tau(\Pi)^A$, and moreover, A satisfies the body of $\tau(r)$. Since A is an answer set of $\tau(\Pi)$, A satisfies the head of r. By definitions of $\tau(r)$ and $\tau(I)$ and lemma 1, this means that I satisfies the head of r and, therefore r itself.
6) We prove that I is minimal, that is, there does not exist an interpretation I' such that I' satisfies the rules of Π^I and $I' \subsetneq I$. We prove by contradiction. Suppose such I' exists. In a) we show that $\tau(I')$ satisfies the rules of $\tau(\Pi)^A$ and in b) we show that $\tau(I')$ is a proper subset of A, thus, obtaining a contradiction to the fact that A is an answer set of $\tau(\Pi)$.
   (a) We prove that $\tau(I')$ satisfies the rules of $\tau(\Pi)^A$. Let r be a rule of $\tau(\Pi)^A$ such that $\tau(I')$ satisfies the body of r. We show that $\tau(I')$ satisfies the head of r. From lemma 1 and the definition of τ and the definition of reducts in ASP and P-log it follows that Π^I contains a rule r' such that $r = \tau(r')$. From 1 we have that I' satisfies the body of r'. Since I' satisfies the rules of Π^I, I' satisfies the head of r'. Therefore, from lemma 1 it follows that $\tau(I')$ satisfies the head of $r = \tau(r')$, and, therefore, the rule r itself.
   (b) We prove that $\tau(I')$ is a proper subset of $A = \tau(I)$. By definition of I',

$$I' \subsetneq I \tag{22}$$

Thus, by definition of τ,

$$\tau(I')^+ \text{ is a proper subset of } \tau(I)^+ \tag{23}$$

From (23) it follows immediately

$$|\tau(I')^+| < |\tau(I)^+| \tag{24}$$

By definition of $\tau(I)$ and $\tau(I')$, we have

$$\tau(I)^- = \bigcup_{f(\bar{x})=y \in I} \{\neg f(\bar{x}) = y_1) | y_1 \in range(f(\bar{x})) \wedge y_1 \neq y\} \tag{25}$$

$$\tau(I')^- = \bigcup_{f(\bar{x})=y \in I'} \{\neg f(\bar{x}) = y_1) | y_1 \in range(f(\bar{x})) \wedge y_1 \neq y\} \tag{26}$$

From (22), (25) and (26) it follows that

$$\tau(I')^- \subseteq \tau(I)^- \tag{27}$$

From (23) and (27) and the fact that $\tau(I) = \tau(I)^+ \cup \tau(I)^-$ and $\tau(I') = \tau(I')^+ \cup \tau(I')^-$ we get

$$\tau(I') \text{ is a proper subset of } \tau(I) \tag{28}$$

Proposition 1 Let Π be a P-log program and W_1, W_2 be two possible worlds of Π. It is not true that $W_1 \subsetneq W_2$.

Proof We prove by contradiction. Let W_1 and W_2 be two possible world of a program Π such that

$$W_1 \subsetneq W_2 \tag{29}$$

By Lemma 2,

$$\tau(W_1) \text{ and } \tau(W_2) \text{ are answer sets of } \tau(\Pi). \tag{30}$$

By definition of τ,

$$\tau(W_1) = W_1 \cup \bigcup_{f(\bar{x})=y \in W_1} \{f(\bar{x}) \neq y_1 | y_1 \in range(f(\bar{x})) \wedge y_1 \neq y\} \tag{31}$$

$$\tau(W_2) = W_2 \cup \bigcup_{f(\bar{x})=y \in W_2} \{f(\bar{x}) \neq y_1 | y_1 \in range(f(\bar{x})) \wedge y_1 \neq y\} \tag{32}$$

From (29), (31) and (32) it follows that

$$\tau(W_1) \subsetneq \tau(W_2) \tag{33}$$

(33) and (30) contradict the theorem about minimality of answer sets of ASP programs (see Lemma 1 in [13]).

A.2 Splitting Set Theorem For P-log

In this section we present the P-log version of the original Splitting Set Theorem from [19]. The adoption requires change in the definition of splitting set (see Definition 12). Other definitions follow [19] and are presented for completeness.

Let Π be a program with signature Σ, and X and U be sets of literals of Σ. As in [19], we will define the *bottom* and the *top* of a program Π with respect to X and U, denoted by $b_U(\Pi)$ and $e_U(\Pi \setminus b_U(P), X)$ correspondingly.

For a rule r of the form

$$l \leftarrow l_1, \ldots, l_k, not\ l_{k+1}, \ldots, not\ l_m \tag{34}$$

where $l_1, \ldots, l_m$ are literals, we will introduce notations:

$$pos(r) = \{l_1, \ldots, l_k\}$$

$$neg(r) = \{l_{l+1}, \ldots, l_m\}$$

$$head(r) = l$$

$$lit(r) = head(r) \cup pos(r) \cup neg(r)$$

Definition 9 (Bottom w.r.t. U)
The bottom of Π w.r.t U, denoted by $b_U(\Pi)$, is a program such that:

1. $b_U(\Pi) = \{r \mid r \in \Pi \text{ and } lit(r) \subseteq U\}$

2. the signature of $b_U(\Pi)$ consists of all attribute terms of Σ which form literals from U

□

We next define Top. For rule r such that $pos(r) \cap U$ is satisfied by X and every literal from $(neg(r) \cap U)$ is not satisfied by X, we define $R_U(r)$ to be the rule such that:

$$head(R_U(r)) = head(r), pos(R_U(r)) = pos(r) \setminus U, neg(R_U(r)) = neg(r) \setminus U$$

Definition 10 (Top w.r.t. X and U)
The top of Π w.r.t X and U, denoted by $e_U(\Pi, X)$, is a program such that:

1. the rules of $e_U(\Pi, X)$ are

$$\{R_U(r) \mid r \in \Pi \text{ and } pos(r) \cap U \text{ is satisfied by } X \text{ and every e-literal from } \{not\ l \mid l \in neg(r) \cap U\} \text{ is not satisfied by } X\}$$

2. the signature of $e_U(\Pi, X)$ consists of all attribute terms of Σ which do not form a literal in U.

□

We borrow the definition of a solution to Π w.r.t U:

Definition 11 (Solution to Π w.r.t U)
Let Π be a P-log program. A solution to Π w.r.t U is a pair $\langle X, Y \rangle$ of sets of literals such that:

1. X is possible world of $b_U(\Pi)$
2. Y is a possible world of $e_U(\Pi \setminus b_U(\Pi), X)$
3. $X \cup Y$ is consistent

□

We will next define a splitting set for P-log programs:

Definition 12 (Splitting set)
A *splitting set* for a P-log program Π is any set U of literals of Π's signature such that,

1. for every rule r of Π if $head(r) \in U$, then $pos(r) \cup neg(r) \subseteq U$,
2. if a literal formed by attribute term $f(\bar{x})$ belongs to U, then all the literals of Σ formed by $f(\bar{x})$ belong to U.

□

Finally, we state the splitting set theorem for P-log:

Theorem 2 *[Splitting Set Theorem]*
Let U be a splitting set for a program Π. A set A of literals is a possible world of Π if and only if $A = X \cup Y$ for some solution $\langle X, Y \rangle$ to Π with respect to U.

□

Note that the condition 2 from Definition 12, absent from the original definition of splitting set, is necessary for the correctness of the theorem. For instance, consider the program:

```
#s: {1,2}.
p: #boolean.
f:   #s.
p:- f != 1.
f = 2.
```

If condition 2 is not used, $U = \{p, f \neq 1\}$ would be a splitting set of Π. However, the theorem then wouldn't hold for U. The program has only one possible world $\{f = 2\}$. However, there does not exists a solution $\langle X, Y\rangle$ with respect to U, such that $A = X \cup U$, because, the program $b_U(\Pi)$, which contains only one rule

```
p:- not f != 1.
```

has exactly one possible world $\{p\}$, and, therefore, $X \cup Y$ must contain p for any solution $\langle X, Y\rangle$ of Π with respect to U.

Proof (Proof for theorem 2)
In 1 we show that $\tau(U)$ is a splitting set (as defined in [19]) for the ASP program $\tau(\Pi)$. In 2 we show $\tau(b_U(\Pi)) = b_{\tau(U)}(\tau(\Pi))$. In 3 we show $e_{\tau(U)}(\tau(\Pi) \setminus b_{\tau(U)}(\tau(\Pi)), X') = \tau(e_U(\Pi \setminus b_U(\Pi), X))$. In 4 we use the results from 1-3 to prove that if A a possible world of Π, then $A = X \cup Y$ for some solution $\langle X, Y\rangle$ to Π with respect to U. In 5 we use the results from 1-3 to prove that if $A = X \cup Y$ for some solution $\langle X, Y\rangle$ to Π with respect to U, then A is a possible world of Π.

1. We show that $\tau(U)$ is a splitting set (as defined in [19]) for the program $\tau(\Pi)$. Let r be a rule of $\tau(\Pi)$ such that:

$$\text{the head of } r \text{ is included into } \tau(U) \tag{35}$$

   We need to show that all the literals occurring in the body of r are included into $\tau(U)$. We consider two possible cases:
   (a) there exists r' of Π such that $r = \tau(r')$. In this case, by construction of $\tau(r')$ and clause 1 of Definition 12 we have that every literal occurring in $pos(r) \cup neg(r)$ is included into U. Thus, by definition of $\tau(U)$ and from $r = \tau(r')$, every literal from the body of r is included into $\tau(U)$.
   (b) r is of the form $\neg f(x, y_1) \leftarrow f(x, y)$ where $y_1 \neq y$. In this case, by construction of $\tau(U)$ from (35) we have that $f(x) \neq y_1 \in U$. Therefore, by clause 2) of definition 12 we have that $f(x) = y \in U$. By definition of $\tau(f(x) = y)$ and $\tau(U)$ we have that $f(x, y) \in \tau(U)$. Therefore, every literal in the body of r belongs to $\tau(U)$.
2. We prove:

$$\tau(b_U(\Pi)) = b_{\tau(U)}(\tau(\Pi)) \tag{36}$$

   We prove (36) in two directions:

$$\tau(b_U(\Pi)) \subseteq b_{\tau(U)}(\tau(\Pi)) \tag{37}$$

$$b_{\tau(U)}(\tau(\Pi)) \subseteq \tau(b_U(\Pi)) \tag{38}$$

We start from (37). Suppose r is a a rule such that

$$r \in \tau(b_U(\Pi)) \tag{39}$$

There are two possible cases:

- r is a translation of a rule of $b_U(\Pi)$. In this case, since $b_U(\Pi) \subseteq \Pi$, r is a translation of a rule in Π. Therefore,

  $$r \text{ belongs to } \tau(\Pi) \tag{40}$$

  Also, since every literal occurring in every rule in $b_U(\Pi)$ is from U, and r is a translation of a rule in $b_U(\Pi)$, we have that:

  $$\text{every literal occurring in } r \text{ is from } \tau(U) \tag{41}$$

  From (41) and (40) we have

  $$r \in b_{\tau(U)}(\tau(\Pi)) \tag{42}$$

  Therefore, (37) holds.
- r is of the form $\neg f(\bar{x}, y_1) \leftarrow f(\bar{x}, y_2)$ In this case, by definition of τ from (39) we have $f(\bar{x}) = y_2$ and $f(\bar{x}) = y_1$ are atoms of $b_U(\Pi)$, which means U contains literals l_1 and l_2 formed by $f(\bar{x}) = y_1$ and $f(\bar{x}) = y_2$ respectively. Since U is a splitting set, by condition 2 we have that U contains atoms $f(\bar{x}) = y_1$ and $f(\bar{x}) = y_2$. Therefore, $\tau(U)$ contains literals $\neg f(\bar{x}, y_1)$ and $f(\bar{x}, y_2)$, $r \in b_{\tau(U)}(\tau(\Pi))$, and (37) holds.

We next show (38). Suppose r is a a rule such that

$$r \in b_{\tau(U)}(\tau(\Pi)) \tag{43}$$

There are two possible cases:

- r is a translation of some rule r' of Π. In this case, from (43) we have that all literals from r are from $\tau(U)$. Therefore, $lit(r') \subseteq U$, $r' \in b_U(\Pi)$ and $r \in \tau(b_U(\Pi))$. Therefore, (38) holds.
- r is of the form $\neg f(\bar{x}, y_1) \leftarrow f(\bar{x}, y_2)$. By construction of $b_{\tau(U)}(\tau(\Pi))$ we have that:

  $$\neg f(\bar{x}, y_1) \in \tau(U) \tag{44}$$

  $$f(\bar{x}, y_2) \in \tau(U) \tag{45}$$

  and

  $$r \in \tau(\Pi) \tag{46}$$

  From (44) and (45) we have:

  $$\text{all the literals of } \Sigma \text{ formed by } f(\bar{x}) \text{ belong to } U \tag{47}$$

  From (47) we have that

  $$f(\bar{x}) \text{ belongs to the signature of } b_U(\Pi) \tag{48}$$

  Therefore, by definition of $\tau(\Pi)$, we have $r \in \tau(b_U(\Pi)$. Therefore, (38) holds.

From (37) and (38) we have (36).

3. We show that for every possible world X of $b_U(\Pi)$:

$$e_{\tau(U)}(\tau(\Pi) \setminus b_{\tau(U)}(\tau(\Pi)), \tau(X)) = \tau(e_U(\Pi \setminus b_U(\Pi), X)) \tag{49}$$

We prove (49) in two directions:

$$e_{\tau(U)}(\tau(\Pi) \setminus b_{\tau(U)}(\tau(\Pi)), \tau(X)) \subseteq \tau(e_U(\Pi \setminus b_U(\Pi), X)) \tag{50}$$

$$\tau(e_U(\Pi \setminus b_U(\Pi), X)) \subseteq e_{\tau(U)}(\tau(\Pi) \setminus b_{\tau(U)}(\tau(\Pi)), \tau(X)) \tag{51}$$

We start from (50). Let r be a rule s.t.

$$r \in e_{\tau(U)}(\tau(\Pi) \setminus b_{\tau(U)}(\tau(\Pi)), \tau(X)) \tag{52}$$

By construction of $e_{\tau(U)}(\tau(\Pi) \setminus b_{\tau(U)}(\tau(\Pi)), \tau(X))$ we have that $r = R_{\tau(U)}(r')$, where

$$r' \in \tau(\Pi) \tag{53}$$

$$r' \notin b_{\tau(U)}(\tau(\Pi)) \tag{54}$$

$$pos(r') \cap \tau(U) \subseteq \tau(X) \tag{55}$$

$$neg(r') \cap \tau(U) \cap \tau(X) = \emptyset \tag{56}$$

From (54) we have:

$$r' \text{ contains a literal which is not from } \tau(U) \tag{57}$$

There are only two possibilities:

– $r' = \tau(r'')$ for some rule r'' from Π. From (55) and (c) by lemma 1 we have:

$$pos(r'') \cap U \text{ is satisfied by } X \tag{58}$$

From (56) and (c) by lemma 1 we have:

$$\text{every literal in } \{not\ l \mid l \in neg(r'') \cap U\} \text{ is not satisfied by } X \tag{59}$$

From (57) we have:

$$lit(r'') \text{ contains a literal which is not in } U \tag{60}$$

Therefore,

$$r'' \notin b_U(\Pi) \tag{61}$$

From (58),(59) and (61) we have:

$$R_U(r'') \in e_U(\Pi \setminus b_U(\Pi), X) \tag{62}$$

Therefore,

$$\tau(R_U(r'')) \in \tau(e_U(\Pi \setminus b_U(\Pi), X)) \tag{63}$$

Since $r' = \tau(r'')$, by construction of R and by Lemma 1 we have:

$$\tau(R_U(r'')) = R_{\tau(U)}(r') \tag{64}$$

From (63) and (64) we have:

$$R_{\tau(U)}(r') \in \tau(e_U(\Pi \setminus b_U(\Pi), X)) \tag{65}$$

Therefore, since $r = R_{\tau(U)}(r')$, we have:

$$r \in \tau(e_U(\Pi \setminus b_U(\Pi), X)) \tag{66}$$

Therefore, in this case, (50) holds.

– r' is of the form

$$\neg f(\bar{x}, y_1) \leftarrow f(\bar{x}, y_2)$$

where $y_1 \neq y_2$ and

$$f(\bar{x}) \text{ is an attribute term of } \Sigma \tag{67}$$

From (57) by clause 2 of the definition of the splitting set we have:

$$\text{no literal formed by } f(\bar{x}) \text{ belongs to U} \tag{68}$$

From (67) and (68) by Definition 10 we have:

$$f(\bar{x}) \text{ belongs to the signature of } e_U(\Pi \setminus b_U(\Pi), X) \tag{69}$$

From (68) we have:

$$R_{\tau(U)}(r') = r' = r \tag{70}$$

From (69) we have:

$$r' \in \tau(e_U(\Pi \setminus b_U(\Pi), X)) \tag{71}$$

From (71) and (70) we have:

$$r \in \tau(e_U(\Pi \setminus b_U(\Pi), X)) \tag{72}$$

Therefore, in this case, (50) holds.

We next prove (51).

Let r be a rule s.t.

$$r \in \tau(e_U(\Pi \setminus b_U(\Pi), X)) \tag{73}$$

There are only two possibilities:

– $r = \tau(r')$ for some $r' \in e_U(\Pi \setminus b_U(\Pi), X)$. By construction of $e_U(\Pi \setminus b_U(\Pi), X)$, we have that there is r'' such that:

$$r'' \in \Pi \tag{74}$$

$$r' = R_U(r'') \tag{75}$$

and:

$$r'' \notin b_U(\Pi) \tag{76}$$

$$pos(r'') \cap U \text{ is satisfied by } X \tag{77}$$

$$\text{every literal from } neg(r'') \cap U \text{ is not satisfied by } X \tag{78}$$

From (76) we have:

$$lit(r'') \text{ contains a literal not from } U \tag{79}$$

From (79), clause 2 of the splitting set definition and the construction of τ we have:

$$lit(\tau(r'')) \text{ contains a literal not from } \tau(U) \tag{80}$$

Therefore,

$$\tau(r'') \notin b_{\tau(U)}(\tau(\Pi)) \tag{81}$$

From (77) and Lemma 1 we have:

$$pos(\tau(r'')) \cap \tau(U) \text{ is satisfied by } \tau(X) \tag{82}$$

From (78) and Lemma 1 we have:

$$\text{every literal from } neg(\tau(r'')) \cap \tau(U) \text{ is not satisfied by } \tau(X) \tag{83}$$

From (74) we have:

$$\tau(r'') \in \tau(\Pi) \tag{84}$$

From (81), (82), (83) and (84) we have:

$$R_{\tau(U)}(\tau(r'')) \in e_{\tau(U)}(\tau(\Pi) \setminus b_{\tau(U)}(\tau(\Pi)), \tau(X)) \tag{85}$$

By construction of R and by Lemma 1 we have:

$$\tau(R_U(r'')) = R_{\tau(U)}(\tau(r'')) \tag{86}$$

From (85) and (86) we have:

$$\tau(R_U(r'')) \in e_{\tau(U)}(\tau(\Pi) \setminus b_{\tau(U)}(\tau(\Pi)), \tau(X)) \tag{87}$$

From (75) and (87) we have:

$$\tau(r') \in e_{\tau(U)}(\tau(\Pi) \setminus b_{\tau(U)}(\tau(\Pi)), \tau(X)) \tag{88}$$

From the fact that $r = \tau(r')$ and (88) we have:

$$r \in e_{\tau(U)}(\tau(\Pi) \setminus b_{\tau(U)}(\tau(\Pi)), \tau(X)) \tag{89}$$

Therefore, in this case, (51) holds.

– r is of the form

$$\neg f(\bar{x}, y_1) \leftarrow f(\bar{x}, y_2)$$

for some $y_1 \neq y_2$.
From (73) by construction of $\tau(e_U(\Pi \setminus b_U(\Pi), X))$ we have:

$$f(\bar{x}) = y_1 \text{ and } f(\bar{x}) = y_2 \text{ are atoms of the signature of } e_U(\Pi \setminus b_U(\Pi), X) \tag{90}$$

Therefore, by clause 2 of Definition 10, we have:

$$f(\bar{x}) = y_1 \text{ and } f(\bar{x}) = y_2 \text{ are atoms of } \Sigma \tag{91}$$

and

$$\text{no literals of } U \text{ are formed by } f(\bar{x}) \tag{92}$$

Therefore,

$$\tau(U) \text{ contains no literals of the forms } f(\bar{x}, y) \text{ or } \neg f(\bar{x}, y) \tag{93}$$

Since $neg(r) = \{\}$, we have:

$$\text{every literal in } \{not\ l \mid l \in neg(r) \cap \tau(U)\} \text{ is not satisfied by } \tau(X) \tag{94}$$

From (93) we have $pos(r) \cap \tau(U) = \emptyset$, therefore:

$$\text{every literal in } pos(r) \cap \tau(U) \text{ is satisfied by } \tau(X) \tag{95}$$

From (91) we have:

$$r \in \tau(\Pi) \tag{96}$$

From (93) we have:

$$r \notin b_{\tau(U)}(\tau(\Pi)) \tag{97}$$

From (96), (97), (95), (94) we have:

$$R_{\tau(U)}(r) \in e_{\tau(U)}(\tau(\Pi) \setminus b_{\tau(U)}(\tau(\Pi)), \tau(X)) \tag{98}$$

From (93) we have:

$$R_{\tau(U)}(r) = r \tag{99}$$

From (99) and (98) we have:

$$r \in e_{\tau(U)}(\tau(\Pi) \setminus b_{\tau(U)}(\tau(\Pi)), \tau(X)) \tag{100}$$

Therefore, in this case, (51) holds.

From (50) and (51) we have (49).

4. $\rightarrow$ we show that if A a possible world of Π, then $A = X \cup Y$ for some solution $\langle X, Y\rangle$ to Π with respect to U.
Let A be a possible world of Π. By lemma 2 we have that $\tau(A)$ is an answer set of $\tau(\Pi)$. By splitting set theorem from [19] we have that $\tau(A) = X' \cup Y'$ for some solution $\langle X', Y'\rangle$ to $\tau(\Pi)$ with respect to $\tau(U)$. Let Σ be the signature of Π. We will construct two sets of atoms of Σ, X and Y, such that
   - $X' = \tau(X)$
   - $Y' = \tau(Y)$
   - $\langle X, Y\rangle$ is a solution to Π with respect to U
   - $X \cup Y = A$.

   In (a) we construct X. In (b) we construct Y. In (c) we show $X' = \tau(X)$. In (d) we show $Y' = \tau(Y)$. In (e) we show $\langle X, Y\rangle$ is a solution to Π with respect to U. In (f) we show $X \cup Y = A$.
   (a) We construct X. First, from construction of $\tau(\Pi)$ if follows that

$$\text{if } f(\bar{x}, y) \text{ occurs as an atom in } \tau(\Pi) \text{ then } f(\bar{x}) = y \text{ is an atom of } \Sigma \tag{101}$$

   Since X' is an answer set of the program $b_{\tau(U)}(\tau(\Pi))$, it can only contain literals which occur in the head of the rules of the program $b_{\tau(U)}(\tau(\Pi))$, and, by definition of $b_{\tau(U)}(\tau(\Pi))$,

$$\text{all literals in } X' \text{ occur in } \tau(\Pi) \tag{102}$$

   Let X be the set defined as follows:

$$X = \{f(\bar{x}) = y \mid f(\bar{x}, y) \in X'\} \tag{103}$$

   From (101) and (102) we have that X is a set of atoms of Σ.
   (b) We construct Y. Using arguments similar to the ones in (a), we can show that if $f(x, y) \in Y'$, then $f(x) = y$ is an atom for Σ. Then the set Y, defined as follows

$$Y = \{f(\bar{x}) = y \mid f(\bar{x}, y) \in Y'\}, \tag{104}$$

   is a set of atoms of Σ.
   (c) We show that $X' = \tau(X)$. Since X' is an answer set of the program $b_{\tau(U)}(\tau(\Pi))$, X' is consistent. Therefore, it is sufficient to show that for every atom $f(\bar{x}) = y$ such that

$$f(\bar{x}) = y \in X \tag{105}$$

   in X,

$$\{\neg f(\bar{x}, y_1) \mid y_1 \neq y\} \subseteq X' \tag{106}$$

   By construction of $\tau(\Pi)$, for every literal $f(\bar{x}) \neq y_1$ where $y_1 \neq y$, $\tau(\Pi)$ contains a rule r: $\neg f(\bar{x}, y_1) \leftarrow f(\bar{x}, y)$

   We prove that

$$r \text{ belongs to } b_{\tau(U)}(\tau(\Pi)) \tag{107}$$

   Since X' is an answer set of $b_{\tau(U)}\tau(\Pi)$, and X' contains $f(\bar{x}, y)$, $f(\bar{x}, y) \in \tau(U)$. By definition of $\tau(U)$, $\tau(U)$ should also include $\neg f(\bar{x}, y_1)$. Thus,

$\tau(U)$ contains both $\tau(f(\bar{x}) = y)$ and $\tau(f(\bar{x}) \neq y_1)$ and, by construction of $b_{\tau(U)}(\tau(\Pi))$, we have (107)

Since X' is an answer set of $b_{\tau(U)}\tau(\Pi)$, from (107) we have:

$$X' \text{ satisfies } r \tag{108}$$

From (105) and (103) we have that X' satisfies the body of r. Therefore, from (108) we have X' also satisfies the head of r, which is $\neg f(\bar{x}, y_1)$. Therefore, (106) holds.

(d) We show that $Y' = \tau(Y)$. Since Y' is an answer set of $e_{\tau(U)}(\tau(\Pi) \setminus b_{\tau(U)}(\tau(\Pi)), X')$, Y' is consistent. Therefore, it is is sufficient to show that for every atom $f(\bar{x}) = y$ such that

$$f(\bar{x}) = y \in Y \tag{109}$$

we have

$$\{\neg f(\bar{x}, y_1) \mid y_1 \neq y\} \subseteq Y' \tag{110}$$

By construction of $\tau(\Pi)$, for every literal $f(\bar{x}) \neq y_1$ of Σ where $y_1 \neq y$, there is a rule r

$$\neg f(\bar{x}, y_1) \leftarrow f(\bar{x}, y)$$

such that

$$r \in \tau(\Pi) \tag{111}$$

We prove that

$$r \in e_{\tau(U)}(\tau(\Pi) \setminus b_{\tau(U)}(\tau(\Pi)), X') \tag{112}$$

From the results on page 5 of [19], we have:

$$Y' \cap \tau(U) = \emptyset \tag{113}$$

From (113), (109) and (104) we have:

$$f(\bar{x}, y) \notin \tau(U) \tag{114}$$

From (114) by construction of $b_{\tau(U)}(\tau(\Pi))$ we have:

$$r \notin b_{\tau(U)}(\tau(\Pi)) \tag{115}$$

From (115) and (111) we have:

$$r \in \tau(\Pi) \setminus b_{\tau(U)}(\tau(\Pi)) \tag{116}$$

From (116) and (114) by definition of top we have (112).

Since Y' is an answer set of $e_{\tau(U)}(\tau(\Pi) \setminus b_{\tau(U)}(\tau(\Pi)), X')$, from (112) we have:

$$Y' \text{ satisfies } r \tag{117}$$

From (109) and (104) we have that Y' satisfies the body of r. Therefore, from (117) we have Y' also satisfies the head of the rule, which is $\neg f(\bar{x}, y_1)$. Therefore, (110) holds.

(e) We show that $\langle X, Y \rangle$ is a solution to Π with respect to U. We prove the clauses 1-3 of Definition 11 in i - iii respectively.

i. We show that X is a possible world of $b_U(\Pi)$. By construction,

$$X' \text{ is an answer set of } b_{\tau(U)}(\tau(\Pi)) \tag{118}$$

From (c), the fact that X' is an answer set of $b_{\tau(U)}(\tau(\Pi))$, 36 by lemma 2 we have that X is a possible world of $b_U(\Pi)$.

ii. We show that Y is a possible world of $e_U(\Pi \setminus b_U(\Pi), X)$.
Recall that:

$$Y' \text{ is an answer set of } e_{\tau(U)}(\tau(\Pi) \setminus b_{\tau(U)}(\tau(\Pi)), X') \tag{119}$$

From (119), (d), and (49) by Lemma 2 we have that Y is a possible world of $e_U(\Pi \setminus b_U(\Pi), X)$.

iii. We prove $X \cup Y$ is consistent. Recall that $\tau(A) = X' \cup Y'$ for a possible world A of Π. From (c) and (d) we have that $X \cup Y$ consists of all positive literals which are members of $X' \cup Y'$, which is precisely A. Since A is a possible world, A is consistent, as well as $X \cup Y$.

(f) In 2.e).iii we have already shown that $A = X \cup Y$.

By theorem 2 and the fact that if Y contains an atom $f(x, y)$, it also contains atoms $f(x, y_1)$ for every $y_1 \neq y$, there exists Y' such that $Y = \tau(Y')$ and Y' is a possible world of $e_U(\Pi \setminus b_U(P), X)$.

5. $\leftarrow$ we show that if $A = X \cup Y$ for some solution $\langle X, Y \rangle$ to Π with respect to U, then A is a possible world of Π.

By 1, $\tau(U)$ is a splitting set of $\tau(\Pi)$. We prove that $\langle \tau(X), \tau(Y) \rangle$ is a solution to $\tau(\Pi)$ with respect to $\tau(U)$. In (a) we show that $\tau(X)$ is a possible world of $b_{\tau(U)}(\tau(\Pi)$. In (b) we show $\tau(Y)$ is a possible world of $e_{\tau(U)}(\tau(\Pi) \setminus b_{\tau(U)}(\tau(\Pi)), X')$. In (c) we show $\tau(X) \cup \tau(Y)$ is consistent.

(a) We show that

$$\tau(X) \text{ is a possible world of } b_{\tau(U)}(\tau(\Pi)) \tag{120}$$

Since $\langle X, Y \rangle$ is a solution of Π w.r.t. U, we have

$$X \text{ is a possible world of } b_U(\Pi) \tag{121}$$

Therefore by Lemma 2 we have:

$$\tau(X) \text{ is a possible world of } \tau(b_U(\Pi)) \tag{122}$$

From (36) and (122) we have (120)

(b) We show that

$$\tau(Y) \text{ is a possible world of } e_{\tau(U)}(\tau(\Pi) \setminus b_{\tau(U)}(\tau(\Pi)), \tau(X)) \tag{123}$$

Since $\langle X, Y \rangle$ is a solution of Π w.r.t. U, we have:

$$Y \text{ is a possible world of } e_U(\tau(\Pi) \setminus b_U(\tau(\Pi)), X) \tag{124}$$

Therefore by Lemma 2 we have:

$$\tau(Y) \text{ is a possible world of } \tau(e_U(\tau(\Pi) \setminus b_U(\tau(\Pi)), X)) \tag{125}$$

From (49) and (125) we have (123)

(c) Since $\langle X, Y\rangle$ is a solution of Π w.r.t. U, $X \cup Y$ is consistent. Therefore, by construction, $\tau(X) \cup \tau(Y)$ is consistent (it consists of atoms from X and Y and all negative literals of the form $f = y$ for every atom $f = y_1$ where $y \neq y_1$).

The original paper also contains an analogy of the following claim that we will use in the proof of Theorem 1.

Lemma 3 Let U be a splitting set for a program Π. If A is a possible world of Π such that $A = X \cup Y$ for some solution $\langle X, Y\rangle$ to Π with respect to U, then $Y \cap U = \emptyset$.

□

Proof Since Y is a possible world of $e_U(\Pi \setminus b_U(\Pi), X)$, and, by clause 2 of Definition 10, the signature of $e_U(\Pi \setminus b_U(\Pi), X)$ does not contain literals from U, Y does not contain literals from U. Therefore, $Y \cap U = \emptyset$.

A.3 Proof of Theorem 1

In this section we will prove the theorem:
Theorem 1.
Every program from $\mathcal{B}$ is coherent.

□

The outline of the proof is the same as that of Theorem 1 from [8], which says that a program from a different class introduced there is coherent. First, [8] introduces the notion of a tableau representing a program and shows that programs considered there can be represented by such tableaux. The second part of the proof consists of the theorem which states that every program which can be represented by a tableau is coherent. The definition of a tableaux and the corresponding theorem about coherency in our proof is very close to that in [8]. The only changes are those related to our refinement of the semantics of the original P-log. However, proof of the first part requires a substantial amount of work and new insights, given in lemmas 6 - 25 below.

Definition 13 (Unitary Tree)
Let T be a tree in which every arc is labeled with a real number in [0,1]. We say T is *unitary* if the labels of the arcs leaving each node add up to 1.

□

Figure 1 gives an example of a unitary tree.

Definition 14 ($p_T(n)$)

Let T be a tree with labeled nodes and n be a node of T. By $p_T(n)$ we denote the set of labels of nodes lying on the path from the root of T to n, including the label of n and the label of the root.

□

Example 14 Consider the tree T from Figure 1. If n is the node labeled (13), then $p_T(n) = \{1, 3, 8, 13\}$.

□

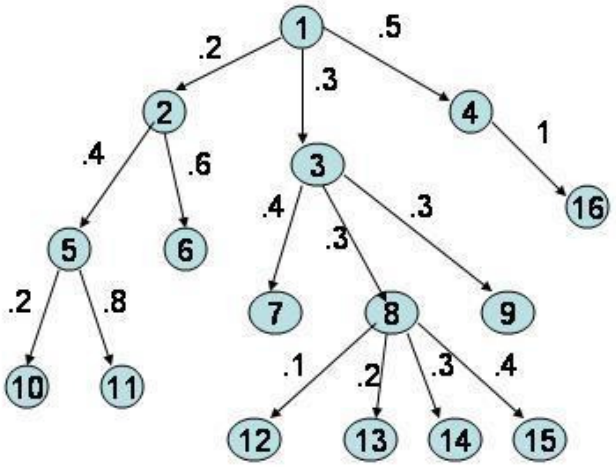


Fig. 1: Unitary tree T

Definition 15 (Path value)
Let T be a tree in which every arc is labeled with a number in [0,1]. The *path value* of a node n of T, denoted by $pv_T(n)$, is defined as the product of the labels of the arcs in the path to n from the root. (Note that the path value of the root of T is 1.)

□

When the tree T is obvious from the context we will simply write $pv(n)$.

Example 15 Consider the tree T from Figure 1. If n is the node labeled with 8, then $pv(n) = 0.3 \times 0.3 = 0.09$.

□

Lemma 4 *[Property of Unitary Trees]*
Let T be a unitary tree and n be a node of T. Then the sum of the path values of all the leaf nodes descended from n (including n if n is a leaf) is the path value of n.

□

The proof of Lemma 4 can be found in [8].

Definition 16 (A set of literals compatible with an e-literal)
A set S of literals of Π is *Π-compatible* with an e-literal l of Π if there exists a possible world of Π satisfying $S \cup \{l\}$. Otherwise S is *Π-incompatible* with l. S is *Π-compatible* with a set B of e-literals of Σ if there exists a possible world of Π satisfying $S \cup B$; otherwise S is *Π-incompatible* with B. □

Definition 17 (A set of literals guaranteeing an e-literal)
A set S of literals is said to *Π-guarantee* an e-literal l if S and l are Π-compatible and every possible world of Π satisfying S also satisfies l; S *Π-guarantees* a set B of e-literals if S Π-guarantees every member of B. □

Definition 18 (Ready to branch)
Let T be a tree whose nodes are labeled with atoms of Σ and r be a rule of Π of the form

$$random(a(\bar{t}) : \{X : p(X)\}) \leftarrow K.$$

where K can be empty. A node n of T is *ready to branch on $a(\bar{t})$ via r relative to Π* if

1. $p_T(n)$ contains no literal of the form $a(\bar{t}) = y$ for any y,
2. $p_T(n)$ Π-guarantees K,
3. for every pr-atom of the form $pr(a(\bar{t}) = y \mid B) = v$ in Π, either $p_T(n)$ Π-guarantees B or is Π-incompatible with B, and
4. for every $y \in range(a)$, $p_T(n)$ either Π-guarantees $p(y)$ or is Π-incompatible with $p(y)$ and moreover there is at least one $y \in range(a)$ such that $p_T(n)$ Π-guarantees $p(y)$.

If Π is obvious from the context we may simply say that n is ready to branch on $a(\bar{t})$ via r.

□

Proposition 2 Suppose n is ready to branch on $a(\bar{t})$ via some rule r of Π, and $a(\bar{t}) = y$ is Π-compatible with $p_T(n)$; and let W_1 and W_2 be possible worlds of Π-compatible with $p_T(n)$. Then $P(W_1, a(\bar{t}) = y) = P(W_2, a(\bar{t}) = y)$.

□

Proof Suppose n is ready to branch on $a(\bar{t})$ via some rule r of Π, and $a(\bar{t}) = y$ is Π-compatible with $p_T(n)$; and let W_1 and W_2 be possible worlds of Π compatible with $p_T(n)$.

Case 1: Suppose $a(\bar{t}) = y$ has an assigned probability in W_1. Then there is a pr-atom $pr(a(\bar{t}) = y \mid B) = v$ of Π such that W_1 satisfies B. Since W_1 also satisfies $p_T(n)$, B is Π-compatible with $p_T(n)$. It follows from the definition of ready-to-branch that $p_T(n)$ Π-guarantees B. Since W_2 satisfies $p_T(n)$ it must also satisfy B and so $P(W_2, a(\bar{t}) = y) = v$.

Case 2: Suppose $a(\bar{t}) = y$ does not have an assigned probability in W_1. Case 1 shows that the assigned probabilities for values of $a(\bar{t})$ in W_1 and W_2 are precisely the same; so $a(\bar{t}) = y$ has a default probability in both worlds. We need only show that the possible values of $a(\bar{t})$ are the same in W_1 and W_2. Suppose then that for some z, $a(\bar{t}) = z$ is possible in W_1. Then W_1 satisfies $p(z)$. Hence since W_1 satisfies $p_T(n)$, we have that $p_T(n)$ is Π-compatible with $p(z)$. By definition of ready-to-branch, it follows that $p_T(n)$ Π-guarantees $p(z)$. Now since W_2 satisfies $p_T(n)$ it must also satisfy $p(z)$ and hence $a(\bar{t}) = z$ is possible in W_2. The other direction is the same.

Suppose n is ready to branch on $a(\bar{t})$ via some rule r of Π, and $a(\bar{t}) = y$ is Π-compatible with $p_T(n)$, and W is a possible world of Π compatible $p_T(n)$. We may refer to the $P(W, a(\bar{t}) = y)$ as $v(n, a(\bar{t}), y)$. Though the latter notation does not mention W, it is well defined by proposition 2.

Example 16 [Ready to branch]
Consider the following version of the dice example. Lets refer to it as Π_{11}

```
#dice = {d1,d2}.
#score = 1..6.
#person = {mike, john}.
roll: #dice -> #score.
owner: #dice -> #person.
owner(d1) = mike.
owner(d2) = john.
even(D) :- roll(D)= Y, Y mod 2 = 0.
-even(D) :- not even(D).
random(roll(D)).
pr(roll(D)=Y | owner(D) = john) = 1/6.
pr(roll(D)=6 | owner(D) = mike) = 1/4.
pr(roll(D)=Y | Y != 6, owner(D) = mike) = 3/20.
```

Now consider a tree T_2 of Figure 2 [8]. Let us refer to the root of this tree as n_1,

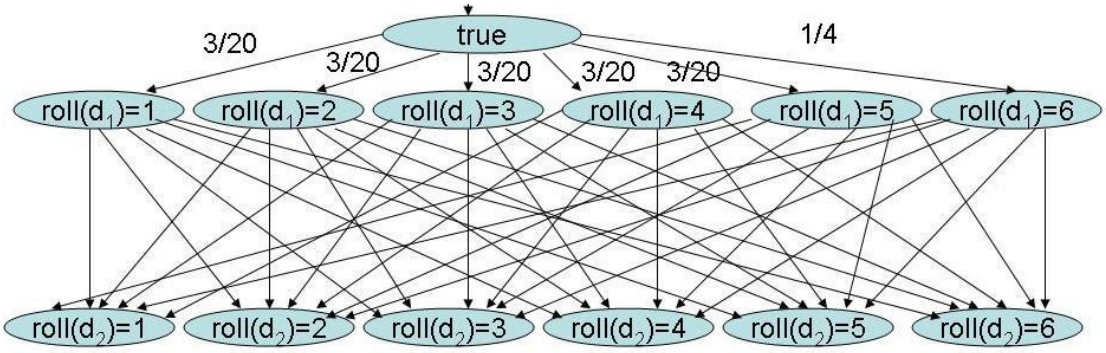


Fig. 2: T_2: The tree corresponding to the dice P-log program Π_{11}

the node $roll(d_1) = 1$ as n_2, and the node $roll(d_2) = 2$ connected to n_2 as n_3. Then $p_{T_2}(n_1) = \{true\}$, $p_{T_2}(n_2) = \{true, roll(d_1) = 1\}$, and $p_{T_2}(n_3) = \{true, roll(d_1) = 1, roll(d_2) = 2\}$. The set $\{true\}$ of literals Π_{11}-guarantees $\{owner(d_1) = mike, owner(d_2) = john\}$ and is Π_{11}-incompatible with $\{owner(d_1) = john, owner(d_2) = mike\}$. Hence n_1 and the attribute $roll(d_1)$ satisfy condition 3 of definition 18. Similarly for $roll(d_2)$. Other conditions of the definition hold vacuously and therefore n_1 is ready to branch on $roll(D)$ via $random(roll(D))$ relative to Π_{11} for $D \in \{d_1, d_2\}$. It is also easy to see that n_2 is ready to branch on $roll(d_2)$ via $random(roll(d_2))$, and that n_3 is not ready to branch on any attribute of Π_{11}.

[8] The root is labeled with *true*, which can be viewed, for instance, as a true arithmetic atom $1 = 1$

□

Definition 19 (Expanding a node)
In case n is ready to branch on $a(\bar{t})$ via some rule of Π, the *Π-expansion* of T at n by $a(\bar{t})$ is a tree obtained from T as follows: for each y such that $p_T(n)$ is Π-compatible with $a(\bar{t}) = y$, add an arc leaving n, labeled with $v(n, a(\bar{t}), y)$, and terminating in a node labeled with $a(\bar{t}) = y$. We say that n *branches on* $a(\bar{t})$.

□

Definition 20 (Expansions of a tree)
A *zero-step Π*-expansion of T is T. A *one-step Π-expansion of* T is an expansion of T at one of its leaves by some attribute term $a(\bar{t})$. For $n > 1$, an *n-step Π-expansion* of T is a one-step Π-expansion of an $(n-1)$-step Π-expansion of T. A *Π-expansion* of T is an n-step Π-expansion of T for some non-negative integer n.

□

For instance, the tree consisting of the top two layers of tree T_2 from Figure 2 is a Π_{11}-expansion of one node tree n_1 by $roll(d_1)$.

Definition 21 (Seed)
A *seed* is a tree with a single node labeled *true*.

□

Definition 22 (Tableau)
A *tableau* of Π is a Π-expansion of a seed which is maximal with respect to the subtree relation.

□

For instance, a tree T_2 of Figure 2 is a tableau of Π_{11}.

Definition 23 (Node representing a possible world)
Suppose T is a tableau of Π. A possible world W of Π is *represented* by a leaf node n of T if W is the set of atoms Π-guaranteed by $p_T(n)$.

□

For instance, a node n_3 of T_2 represents a possible world $\{owner(d_1, mike), owner(d_2, john), roll(d_1, 1), roll(d_2, 2), \neg even(d_1), even(d_2)\}$.

Definition 24 (Tree representing a program)
If every possible world of Π is represented by exactly one leaf node of T, and every leaf node of T represents exactly one possible world of Π, then we say T *represents* Π. □

It is easy to check that the tree T_2 represents Π_2.

Definition 25 (Probabilistic soundness)
Suppose Π is a P-log program and T is a tableau representing Π, such that R is a mapping from the possible worlds of Π to the leaf nodes of T which represent them. If for every possible world W of Π we have

$$pv_T(R(W)) = \mu(W)$$

i.e. the path value in T of $R(W)$ is equal to the normalized measure of W, then we say that the representation of Π by T is *probabilistically sound*. □

The following lemma gives conditions sufficient for the coherency of P-log programs. It will later be shown that all unitary, dynamically causally ordered programs satisfy the hypothesis of this theorem, establishing Theorem 1.

Lemma 5 (Coherency Condition)
Let Π be a program, and Π' be a program obtained from Π by removing activity records. If there exists a unitary tableau T representing Π', and this representation is probabilistically sound, then $P_{\Pi'}$ is defined, and for every pair of rules

$$random(a(\bar{t}) : \{X : p(X)\}) \leftarrow K. \tag{126}$$

and

$$pr(a(\bar{t}) = y \mid B) = v. \tag{127}$$

of Π' such that $P_{\Pi'}(B \cup K) > 0$ we have

$$P_{\Pi' \cup o(B) \cup o(K)}(a(\bar{t}) = y) = v$$

Hence Π is coherent.

□

Proof Since there exists a unitary tableau representing Π', by lemma 4 and Definition 25 we have that there exists at least one possible world with a non-zero measure. Therefore, $P_{\Pi'}$ is defined.

For any set S of literals, let $lgar(S)$ (pronounced "L-gar" for "leaves guaranteeing") be the set of leaves n of T such that $p_T(n)$ Π'-guarantees S.

Let μ denote the normalized measure on possible worlds induced by Π'.

Let Ω be the set of possible worlds of $\Pi' \cup o(B) \cup o(K)$. Since $P_{\Pi'}(B \cup K) > 0$ we have

$$P_{\Pi' \cup o(B) \cup o(K)}(a(\bar{t}) = y) = \frac{\sum_{\{W \,:\, W \in \Omega \,\wedge\, a(\bar{t})=y \,\in\, W\}} \mu(W)}{\sum_{\{W \,:\, W \in \Omega\}} \mu(W)} \tag{128}$$

Now, let

$$\alpha = \sum_{n \in lgar(B \cup K \cup \{a(\bar{t})=y)\}} pv(n)$$

$$\beta = \sum_{n \in lgar(B \cup K)} pv(n)$$

Since T is a probabilistically sound representation of Π', the right-hand side of (128) can be written as α/β. So we will be done if we can show that $\alpha/\beta = v$.

We first claim

$$\text{Every } n \in lgar(B \cup K) \text{ has a unique ancestor } ga(n) \text{ which branches on } a(\bar{t}) \text{ via } \textit{rule } (126). \tag{129}$$

If existence failed for some leaf n then n would be ready to branch on $a(\bar{t})$ which contradicts maximality of the tree. Uniqueness follows from Condition 1 of Definition 18.

Next, we claim the following:

$$\text{For every } n \in lgar(B \cup K),\ p_T(ga(n))\ \Pi'\text{-guarantees } B \cup K. \tag{130}$$

Let $n \in lgar(B \cup K)$. Since $ga(n)$ branches on $a(\bar{t})$, $ga(n)$ must be ready to branch on $a(\bar{t})$ via a rule of Π'. So by clause 3 of Definition 18, $ga(n)$ either Π'-guarantees B or is Π'-incompatible with B. But $p_T(ga(n)) \subset p_T(n)$, and $p_T(n)$ Π'-guarantees B, so $p_T(ga(n))$ cannot be Π'-incompatible with B. Hence $p_T(ga(n))$ Π'-guarantees B. It also follows from clause 2 of Definition 18 that $p_T(ga(n))$ Π'-guarantees K.

From (130), it follows easily that

$$\text{If } n \in lgar(B \cup K),\ \text{every leaf descended from of } ga(n) \text{ belongs to } lgar(B \cup K). \tag{131}$$

Let

$$A = \{ga(n) : n \in lgar(B \cup K)\}$$

In light of (129) and (131), we have

$$lgar(B \cup K) \text{ is precisely the set of leaves descended from nodes in } A. \tag{132}$$

Therefore,

$$\beta = \sum_{n \text{ is a leaf descended from some } a \in A} pv(n)$$

Moreover, by construction of T, no leaf may have more than one ancestor in A, and hence

$$\beta = \sum_{a \in A} \sum_{n \text{ is a leaf descended from } a} pv(n)$$

Now, by Lemma 4 on unitary trees, since T is unitary,

$$\beta = \sum_{a \in A} pv(a)$$

This way of writing β will help us complete the proof. Now for α. Recall the definition of α:

$$\alpha = \sum_{n \in lgar(B \cup K \cup \{a(\bar{t}) = y\})} pv(n)$$

Denote the index set of this sum by $lgar(B, K, y)$. Let

$$A_y = \{n : parent(n) \in A,\ \text{the label of } n \text{ is } a(\bar{t}) = y\}$$

Since $lgar(B, K, y)$ is a subset of $lgar(B \cup K)$, (132) implies that $lgar(B, K, y)$ is precisely the set of nodes descended from nodes in A_y. Hence

$$\alpha = \sum_{n' \text{ is a leaf descended from some } n \in A_y} pv(n')$$

Again, no leaf may descend from more than one node of A_y, and so by the lemma on unitary trees,

$$\alpha = \sum_{n \in A_y} \sum_{n' \text{ is a leaf descended from } n} pv(n') = \sum_{n \in A_y} pv(n) \tag{133}$$

Finally, we claim that every node n in A has a unique child in A_y, which we will label $ychild(n)$. The existence and uniqueness follow from (130), along with Condition 3 of Section 2, and the fact that every node in A branches on $a(\bar{t})$ via rule 126. Thus from (133) we obtain

$$\alpha = \sum_{n \in A} pv(ychild(n))$$

Note that if $n \in A$, the arc from n to $ychild(n)$ is labeled with v. Now we have:

$$\begin{aligned} & P_{\Pi' \cup obs(B) \cup obs(K)}(a(\bar{t}) = y) \\ & = \alpha/\beta \\ & = \sum_{n \in A} pv(ychild(n))/ \sum_{n \in A} pv(n) \\ & = \sum_{n \in A} pv(n) * v/ \sum_{n \in A} pv(n) \\ & = v. \end{aligned}$$

Lemma 6 *[Tableau for programs from $\mathcal{B}$]*
Suppose Π is a program from $\mathcal{B}$ and U is the set of activity records in Π; then there exists a tableau T of $\Pi \setminus U$ which represents $\Pi \setminus U$ such that the representation of $\Pi \setminus U$ by T is probabilistically sound.

□

Proof Let $\alpha = a_1 \ldots, , a_k$ be a probabilistic leveling of Π s.t. Π is dynamically causally ordered via α and $\Pi_0, \ldots, \Pi_k$ be the dynamic structure of $\Pi \setminus U$ induced by α. Let W_0 be the possible world of Π_0.

Consider a sequence $T_0, \ldots, T_k$ of trees where T_0 is a tree with one node, n_0, labeled by *true*, and T_i is obtained from T_{i-1} by expanding every leaf of T_{i-1} which is ready to branch on $a_i(\bar{t}_i)$ via any rule relative to Π_i by this term. Let $T = T_k$. We will show that T_m is a tableau of Π which represents Π and the representation is probabilistically sound.

Our proof will unfold as a sequence of lemmas: Let Σ_i be the signature of Π_i for every $i \in \{0..k\}$, and L_i be the set of all e-literals that can be formed by attribute terms from the signature of Π_i.

Lemma 7 Let Π be a P-log program with signature Σ, A be a set of attribute terms of Σ, L be the set of all e-literals of Σ formed by attribute terms from A. Suppose there exists a set of atoms $W_L \subseteq L$ such that every possible world W of Π, $W_L \subseteq W$ and $(W \setminus W_L) \cap L = \emptyset$. Let $R \subseteq \Pi$ be a subset of rules of Π such that for every $r \in R$, the body of r contains an e-literal from L which is not satisfied by W_L. We have:

$$\Omega_\Pi = \Omega_{\Pi \setminus R} \tag{134}$$

□

Proof In 1 we will prove $\Omega_\Pi \subseteq \Omega_{\Pi \setminus R}$. In 2 we will prove $\Omega_{\Pi \setminus R} \subseteq \Omega_\Pi$. (134) follows immediately from 1 and 2.

1. We prove
$$\Omega_{\Pi} \subseteq \Omega_{\Pi \setminus R} \tag{135}$$
Let $W \in \Omega_{\Pi}$ be a possible world of Π. We will show
$$W \in \Omega_{\Pi \setminus R} \tag{136}$$
Consider the reduct $\Pi' = (\Pi \setminus R)^W$. To show (136), in 1.1 we will show W satisfies the rules of $(\Pi \setminus R)^W$. In 1.2 we will prove W is minimal such set.
   1.1 We show W satisfies the rules of $(\Pi \setminus R)^W$. Since $W \in \Omega_{\Pi}$, W satisfies Π^W. Since $(\Pi \setminus R)^W \subseteq \Pi^W$, W satisfied the rules of $(\Pi \setminus R)^W$.
   1.2 For the sake of contradiction, suppose there exists $W' \subsetneq W$ such that W' satisfies $(\Pi \setminus R)^W$. We will show that
$$W' \text{ satisfies } \Pi^W \tag{137}$$
W' satisfies the subset $(\Pi \setminus R)^W$ of Π^W. Now suppose $r \in R^W$. Let r^* be the rule of Π which produced r in Π^W. By construction of R, the body of r^* contains an e-literal l formed by an attribute term from A not satisfied by W_L. Since $W_L \subseteq W$ and $(W \setminus W_L) \cap L = \emptyset$, L is the set of all e-literals formed by attribute terms from A, the body of r^* contains l which is not satisfied by W. l cannot have default negation (or else, the rule r shouldn't belong to the reduct R^W). Therefore, l belongs to the body of r. Since $W' \subsetneq W$, and all the literals in the body of r do not contain default negation, l is not satisfied by W'. Therefore, W' satisfies r.
2. We prove
$$\Omega_{\Pi \setminus R} \subseteq \Omega_{\Pi} \tag{138}$$
Let $W \in \Omega_{\Pi \setminus R}$ be a possible world of $\Pi \setminus R$. We will show
$$W \in \Omega_{\Pi} \tag{139}$$
Consider the reduct $\Pi' = (\Pi)^W$. To show (139), in 2.1 we will show W satisfies the rules of $(\Pi)^W$. In 2.2 we will prove W is minimal such set.
   2.1 Since $W \in \Omega_{\Pi \setminus R}$, it satisfies the rules of $(\Pi \setminus R)^W$. The further reasoning is similar to 1.2.
   2.2 For the sake of contradiction suppose there exists $W' \subsetneq W$ which satisfies $(\Pi)^W$. Since $(\Pi \setminus R)^W \subseteq (\Pi)^W$, W' also satisfies $(\Pi \setminus R)^W$, which is a contradiction to the fact that W is a possible world of $(\Pi \setminus R)$.

Lemma 8 Let Π be a program with signature Σ such that the base of Π has a unique possible world. We have $\Omega_{red(\Pi)} = \Omega_{\Pi}$. □

Proof Let L'_0 be the set of literals of Σ each of which does not depend on an a random attribute term of Π. L'_0 is a splitting set of Π. therefore, by splitting set theorem, for every possible world W of Π we have $W'_0 \subseteq W$ and $(W \setminus W'_0) \cap L'_0 = \emptyset$. By construction of $red(\Pi)$, $\Pi = red(\Pi) \cup R$, where the body of every rule in R contains a e-literal from L'_0 not satisfied by W'_0. Then the lemma follows trivially from lemma 7.

Lemma 9 Let $0 \leq i \leq k$ be an integer. $\Omega_{red(\Pi_i)} = \Omega_{\Pi_i}$. □

Proof Since $b_{L_0'}(\Pi_i) = b_{L_0'}(\Pi)$, $b_{L_0'}(\Pi_i)$ is the base of Π which has a unique possible world W_0'. Then the lemma follows immediately from Lemma 8.

Lemma 10 Let $0 \leq i \leq k$ be an integer. Let W_i be a possible world of Π_i. We have:

1. $W_0 \subseteq W_i$
2. $(W_i \setminus W_0) \cap L_0 = \emptyset$

Proof Let L_0' be the set of literals from Π's base signature, and W_0' be the answer set of Π's base. By Lemma 9

$$\Omega_{red(\Pi_i)} = \Omega_{\Pi_i} \tag{140}$$

Therefore, $W_i \in \Omega_{red(\Pi)}$. The lemma follows from the fact that L_0 is a splitting set of $red(\Pi_i)$, and $red(\Pi_0) = b_{L_0}(red(\Pi_i))$, and Lemma 9.

Lemma 11 Consider integers n, m such that $0 \leq n \leq m \leq k$. If W_m is a possible world of Π_m, then there exists a unique possible world W_n of Π_n such that $W_n \subseteq W_m$, and $(W_m \setminus W_n) \cap L_n = \emptyset$.

□

Proof In the first part of the proof we show the existence of W_n. We start from two special cases.

- *Case 1.* $n = m$. The claim clearly holds, we can have $W_n = W_m$.
- *Case 2.* $n = 0$. That is, we prove that if W_m is a possible world of the program Π_m, then there exists a unique possible world W_0 of the program Π_0 such that $(W_m \setminus W_0) \cap L_0 = \emptyset$.
  From the definition of a dynamically causally ordered program, Π_0 has a unique possible world W_0. Therefore, from Lemma (10) every possible world A_m of the program Π_m can be written as $W_0 \cup Y$, for some Y such that $Y \cap L_0 = \emptyset$.

  The proof is by double induction on n, m.

1. (*Base case* $n = m = 0$) The case follows immediately from *Case* 1.
2. (*Inductive Hypothesis*) Let h and j be two integers in the range $\{0..k\}$ such that $h \geq j > 0$. Let d and g be a pair of integers such that

   $$d \leq j,$$
   $$g \leq h,$$
   $$d \leq g$$

   and

   $$d + g < h + j.$$

   For every possible world W_d of the program Π_d there exists a possible world W_g of the program Π_g such that $W_d = W_g \cup U_g$, where $U_g \cap L_g = \emptyset$
3. (*Inductive Step*) We prove that for every possible world W_h of the program Π_h there exists a possible world W_j of the program Π_j such that $W_h = W_j \cup U_j$ where $U_j \cap L_j = \emptyset$. Let W_j be the set $W_h|_{L_j}$ we prove that W_j is a possible world of Π_j. In a) we show that W_j satisfies the rules of $\Pi_j^{W_j}$ and in b) we show that W_j is minimal.

(a) We show that W_j satisfies the rules of $\Pi_j^{W_j}$. Let r be a rule of $\Pi_j^{W_j}$ such that the body of r is satisfied by W_j. We prove that the head of r is satisfied by W_j. Let r' be the rule of Π_j from which r was obtained during the computation of $\Pi_j^{W_j}$. Since $W_h \setminus W_j$ does not contain literals from L_j, and the rules of the program Π_j is a subset of the rules of the program Π_h, r' belongs to the rules of Π_h, and the reduct Π^{W_h} will contain the rule r. Since $W_j \subset W_h$, W_h satisfies the body of r. Therefore, Since W_h is an answer set of Π_h, the head of r is included into W_h. Since r belongs to $\Pi_j^{W_j}$, its head must belong to L_j. Since $W_j = W_h|_{L_j}$, the head of r also belongs to W_j. This means that W_j satisfies the head of r.

(b) We show that W_j is minimal. That is, there does not exist an interpretation W_j' of Π_j such that

$$W_j' \subsetneq W_j \tag{141}$$

and W_j' satisfies the rules of $\Pi_j^{W_j}$. We prove by contradiction. Suppose such an interpretation exists. Let's define U_j and W_h' as follows:

$$U_j = W_h \setminus W_j \tag{142}$$

$$W_h' = W_j' \cup U_j \tag{143}$$

From (142) we have:

$$U_j \cap W_j = \emptyset \tag{144}$$

From (144) and (141) we have:

$$U_j \cap W_j' = \emptyset \tag{145}$$

From (141) - (145) we have:

$$W_h' \subsetneq W_h \tag{146}$$

We show that W_h' satisfies the rules of $\Pi_h^{W_h}$, thus, obtaining a contradiction to the fact that W_h is a possible world of Π_h. Let r be a rule of $\Pi_h^{W_h}$ such that W_h' satisfies the body of r. We prove that W_h' satisfies the head of r. Let r' be the rule of Π_h from which r was obtained during the computation of $\Pi_h^{W_h}$. We consider two possible cases.

i. r' is a rule of Π_j. In this case r must belong to $\Pi_j^{W_j}$. Since W_j' satisfies the rules of $\Pi_j^{W_j}$, and it satisfies the body of r, it must satisfy the head of r.

ii. r' is not a rule of Π_j. We show that W_h satisfies the body of r'. First, since r belongs to the reduct $\Pi_h^{W_h}$, $\{not\ l \mid l \in neg(r)\}$ must be satisfied by W_h. Since W_h' satisfies the body of r, which is precisely $pos(r)$, and $W_h' \subsetneq W_h$, W_h satisfies $pos(r)$ too. This means

$$W_h \text{ satisfies the body of } r' \tag{147}$$

Let us denote the head of r' by l_0. Since W_h is a possible world of Π_h, from (147) we have

$$W_h \text{ satisfies } l_0 \tag{148}$$

We consider two cases:

A. l_0 is a member of L_j. We first prove that l_0 must be of one of the forms $random(a_q, p)$ or $a_q = y$ for some random attribute term a_q, where $q \leq j$. We prove by contradiction. Suppose l_0 is either formed by a random attribute term a_r, where $r > j$, or it is formed by a non-random attribute term $random(a_r, p)$, where $r > j$, or it is formed by a non-random attribute term b whose attribute is not *random*. The first two cases are clearly impossible, because L_j contains only attribute terms $a_0, \ldots, a_j$ and $random(a_s, p)$ for $0 \leq s \leq r$, and l_0 belongs to L_j. Consider the latter case, where l_0 is formed by non-random attribute term b whose attribute is not *random*. We show that, in this case, the level of attribute term b in Π must exceed j, thus, obtaining a contradiction to the fact that l_0 is a member of L_j.
   - We show by contradiction that the rule r' belongs to $red(\Pi)$. Suppose that r' does not belong to $red(\Pi)$. This implies that there is an extended literal el in the body of r' formed by an atom in the signature of Π_0 such that W_0 does not satisfy el. By case #2, we get that the possible world W_h can be written as $W_0 \cup U'$, where $U' \cap L_0 = \emptyset$. Therefore, W_h does not satisfy the literal el, which contradicts 147.
   - We show that the level of $body(r')$ in Π must exceed j. Suppose the level of $body(r')$ does not exceed j. In this case, if both b and $body(r')$ have level $\leq j$, the rule r' must belong to Π_j, which contradicts our previous assumption. Thus, since r' belongs to $red(\Pi)$, b must have a level $> j$.

   Thus, we are left with the two cases when

$$l_0 \text{ is either formed by } a_q \text{ or is of the form } random(a_q, p) \quad (149)$$

for random attribute term $a_q \in \{a_1 \ldots, a_j\}$. By inductive hypothesis, there exists a possible world W_{q-1} of the program Π_{q-1} such that

$$W_h = W_{q-1} \cup U_{q-1}, \quad (150)$$

and

$$U_{q-1} \cap L_{q-1} = \emptyset \quad (151)$$

Since r' does not belong to Π_j,

$$r' \text{ contains at least one literal which does not belong to } L_j. \quad (152)$$

Since $q \leq j$,

$$L_q \subseteq L_j. \quad (153)$$

From (152) and (153) it follows that

$$r' \text{ contains at least one literal which does not belong to } L_q \quad (154)$$

Since the head of r' is formed by a_q, from (154) it follows that

$$\text{the body of } r' \text{ contains a literal which does not belong to } L_q \quad (155)$$

From (155) it follows that

$$W_{q-1} \text{ does not satisfy the body of } r'. \tag{156}$$

Therefore, by clause 1 of Definition 6 from (156) it follows that we have only of the two cases:

- W_{q-1} falsifies the body of r'
  which means that the body of r' contains an extended literal el_{q-1} from the signature of Π_{q-1} such that W_{q-1} does not satisfy it. From (150) and (151) it follows that W_h does not satisfy el_{q-1}, and, therefore, it does not satisfy the $body(r')$. Therefore, we have a contradiction to (147).
- r' is a general axiom, which is, given that it's head is either $random(a_q, p)$ or a_q, must be of the form:

$$a_q = y \leftarrow random(a_q, p), not\ a_q = y_1, \ldots, not\ a_q = y_k \tag{157}$$

  In this case, the level of all attribute terms in r' is $q \leq j$, and, therefore, the rule r' must belong to Π_j. Contradiction to the main assumption in ii.

B. l_0 is not a member of L_j. In this case, since, by (148), W_h satisfies l_0 and $W_h = W_j \cup U_j$, and all the literals in W_j belong to L_j, l_0 belongs to U_j. Since $W'_h = W'_j \cup U_j$, W'_h satisfies l_0.

In the second part of the proof we show the uniqueness of W_j. Suppose there exist two different possible worlds W_j^1 and W_j^2 of Π_j such that

$$W_h = W_j^1 \cup U_j^1 \tag{158}$$

$$W_h = W_j^2 \cup U_j^2 \tag{159}$$

$$U_j^1 \cap L_j = \emptyset \tag{160}$$

$$U_j^2 \cap L_j = \emptyset \tag{161}$$

Since L_j contains all the literals that can be constructed from the signature of Π_j, from equations (158) and (160) it follows that

$$W_j^1 = W_h|_{L_j} \tag{162}$$

Similarly, from equations (159) and (161) it follows that

$$W_j^2 = W_h|_{L_j} \tag{163}$$

From equations (162) and (163) it follows that $W_j^1 = W_j^2$. This contradicts our original assumption that W_j^1 and W_j^2 are two **different** possible world of Π_j.

Lemma 12 Let Π be a program with a possible world W. The program $\Pi \cup W$ has a unique possible world W.

Proof Since W is a possible world of Π, W satisfies Π^W. Therefore, W satisfies $(\Pi \cup W)^W = \Pi^W \cup W^W = \Pi^W \cup W$. W is minimal, because, every possible world of $\Pi^W \cup W$ must include W.

W is the only possible world of $\Pi \cup W$, because every possible world of $\Pi \cup W$ must include W, and, by Proposition 1, no possible world which includes W and is different from W can exist.

Lemma 13 Let $i \in \{0..k-1\}$ be an integer and V be a possible world of Π_i. Let Π' be a program from the set $\{\Pi_{i+1} \cup V \cup \{\leftarrow not\ a_{i+1} = y\}, \Pi_{i+1} \cup V \cup \{a_{i+1} = y\}, \Pi_{i+1} \cup V\}$. For every possible world W of Π', $W_0 \subseteq W$ and $W \setminus W_0 \cap L_0 = \emptyset$.

□

Proof Let L_0' be the set of literals from the base S of Π', and W_0' be the answer set of S. We first show

$$\Omega_{red(\Pi')} = \Omega_{\Pi'} \tag{164}$$

Since $b_{L_0'}(\Pi_i') = bot_{L_0'}(\Pi_i) \cup W_0'$, $b_{L_0'}(\Pi_i')$ has a unique possible world W_0'. Therefore, by splitting set theorem, for every possible world W of Π_i' we have $W_0' \subseteq W$ and $(W \setminus W_0') \cap L_0' = \emptyset$. By construction of $red(\Pi_i')$, $\Pi = red(\Pi_i') \cup R$, where the body of every rule in R contains a e-literal from L_0' not satisfied by W_0'. (164) follows immediately from lemma 7.

Clearly, L_0 is a splitting set of $red(\Pi_i')$, and $b_{L_0}(red(\Pi_i')) = red(\Pi_0) \cup W_0$, and Lemma 9. By lemma (9), W_0 is a possible world of $red(\Pi_0)$. By Lemma 12, W_0 is a possible world of $red(\Pi_0) \cup W_0$. Therefore, by splitting set theorem, for every possible W of $red(\Pi_i')$ we have $W_0 \subseteq W$ and $W \setminus W_0 \cap L_0$. Therefore, by (164), the lemma holds.

Lemma 14 Let $i \in \{0..k-1\}$ be an integer and V be a possible world of Π_i. Let Π' be a program from the set $\{\Pi_{i+1} \cup V \cup \{\leftarrow not\ a_{i+1} = y\}, \Pi_{i+1} \cup V \cup \{a_{i+1} = y\}, \Pi_{i+1} \cup V\}$. For every possible world W of Π', $W \setminus V$ does not contain literals from L_i.

□

Proof We define X as follows:

$$X = (W \setminus V)|_{L_i} \tag{165}$$

We show:

$$W \setminus X \text{ satisfies the rules of } \Pi_{i+1}^W \tag{166}$$

Let r be a rule of Π_{i+1}^W. We consider 2 cases:

1. Suppose the head of r is not a literal of X. If the body of r is satisfied by $W \setminus X$, it is also satisfied by W, thus, since W satisfies the rules of Π_{i+1}^W, it contains the head of r. Since the head of r does not belong to X, $W \setminus X$ satisfies head of r.
2. Suppose the head of r is formed by a literal from X. We need to show that, if the body of r is satisfied by $W \setminus X$, the head of r is also satisfied by $W \setminus X$. We consider two cases:

(a) the head of r is of the form $a_j = y$, or of the form $random(a_j, p)$, for a random attribute term a_j, where $j \leq i$. By Lemma 11, there must exists a possible world V_{j-1} of Π_{j-1} such that

$$V_{j-1} \subseteq V \tag{167}$$

and

$$(V \setminus V_{j-1}) \cap L_{j-1} = \emptyset \tag{168}$$

Let r' be the rule of Π_{i+1} from which r was obtained during the computation of the reduct Π_{i+1}^W By definition of dynamically causally ordered program, there are three cases:

i.
$$V_{j-1} \text{ satisfies the body of } r' \tag{169}$$
and
$$\text{all the literals occurring in } r' \text{ are in } L_{j-1} \tag{170}$$
Since $j \leq i$, $L_{j-1} \subseteq L_i$, we have
$$r' \text{ belongs to } \Pi_i \tag{171}$$
From (167), (168), (169) and (170) we have that V satisfies the body of r'.
Since V is a possible world of Π_i from (171) we have V contains the head of r' (and r, since the heads of r and r' are the same). Since $V \subseteq (W \setminus X)$, $(W \setminus X)$ also contains the head of r

ii. V_{j-1} falsifies the body of r'. Because $(V \setminus V_{j-1}) \cap L_{j-1} = \emptyset$, V falsifies the body of r'. That is, there exists an e-literal l belonging to the body of r' such that $l \in L_{j-1}$ and V does not satisfy l. Because $((W \setminus X) \setminus V) \cap L_i = \emptyset$, and $L_{j-1} \subseteq L_i$, we have that $((W \setminus X) \setminus V) \cap L_{j-1} = \emptyset$. Therefore, since V falsifies $body(r')$, $W \setminus X$ falsifies $body(r')$. If l does not contain default negation, this contradicts our original assumption that the body of r is satisfied by $W \setminus X$. Suppose now $l = not\ l'$, where $l' \in L_{j-1}$. Since V does not satisfy l, V satisfies l'. Since $V \subseteq W$ (in all 3 cases) , W satisfies l'. Therefore, W does not satisfy l. This contradicts the fact that r from the reduct Π_{i+1}^W is obtained from r'.

iii. r' is a general axiom of the form
$$a_j = y \leftarrow random(a_j, p), not\ a_j = y_1, \ldots, not\ a_j = y_k$$
and r is of the form
$$a_j = y \leftarrow random(a_j, p)$$
Since $W \setminus X$ satisfies the body of r, and all the literals of L_i from $W \setminus X$ are contained in V, we have
$$random(a_j, p) \in V \tag{172}$$
Since $V \subseteq W$, $random(a_j, p) \in W$. Since r belongs to the reduct Π_{i+1}^W, $\{a_j = y_1, \ldots, a_j = y_k\} \cap W = \emptyset$. Since $V \subseteq W$,
$$\{a_j = y_1, \ldots, a_j = y_k\} \cap V = \emptyset \tag{173}$$

Since $r' \in \Pi_i$ (all the literals are clearly in $L_j \subseteq L_i$), and V is a possible world of Π_i, from (172) and (173) we have $a_j = y \in V$. Since X does not contain literals from V, and $V \subseteq W$, $a_j = y \in W \setminus X$.

(b) the head of r is formed by a non-random attribute term nr, whose attribute is not *random*, and whose level in Π is $\leq i$. Let r' be the rule of Π^W_{i+1} from which r was obtained. We consider two cases:

i. r' does not belong to $red(\Pi)$. In this case the body of r' contains an e-literal $l \in L_0$ such that $|a| = 0$ and W_0 does not satisfy l.
By Lemma 13 we have:

$$W_0 \subseteq W \tag{174}$$

$$(W \setminus W_0) \cap L_0 = \emptyset \tag{175}$$

From (175) we have $((W \setminus X) \setminus W_0) \cap L_0 = \emptyset$. Therefore, since W_0 does not satisfy l and $l \in L_0$, $W \setminus X$ also does not satisfy l, which contradicts the fact that $W \setminus X$ satisfies the body of r.

ii. r' belongs to $red(\Pi)$ The body of r' must consist of literals in L_i (otherwise, the head of r will not belong to L_i, which contradicts the fact $l \in X$). Since $(W \setminus X) \setminus V$ does not contain literals from L_i, and $(W \setminus X)$ satisfies the body of r, V satisfies the body of r. Since r belongs to the reduct Π^W_{i+1}, W satisfies all extended literals of the body of r' preceded by default negation. Since $V \subseteq W$, V also satisfies all extended literals of the body of r' preceded by default negation. This means that V satisfies the body of r'. Since all the literals in r' are members of L_i, r' must belong to Π_i. Since V is a possible world of Π_i, it must satisfy r', thus, the head of r', which is the same as the head of r, must belong to V. Since $V \subseteq W \setminus X$, and V satisfies the head of r, $W \setminus X$ satisfies the head of r.

To conclude the proof, we consider the 3 possible values of Π' from the lemma separately and show $X = \emptyset$:

1. Suppose W is the possible world of $\Pi_{i+1} \cup V$. We need to show that $X = \emptyset$. For the sake of contradiction suppose $X \neq \emptyset$. We have previously shown that $W \setminus X$ satisfies the rules of Π^W_{i+1}. Since $V \subseteq W \setminus X$, $W \setminus X$ satisfies V. Therefore, $W \setminus X$ satisfies the rules of $\Pi^W_{i+1} \cup V^W$, which contradicts the fact that W is a possible world of $\Pi_{i+1} \cup V$.
2. Suppose W is the possible world of $\Pi_{i+1} \cup V \cup \{\leftarrow not\ a_{i+1} = y\}$. We show that $X = \emptyset$. For the sake of contradiction suppose $X \neq \emptyset$. We previously showed that $W \setminus X$ satisfies the rules of Π^W_{i+1}. Since $V \subseteq W \setminus X$, $W \setminus X$ satisfies V. Since W is a possible world of $\Pi_{i+1} \cup V \cup \{\leftarrow not\ a_{i+1} = y\}$, W contains $a_{i+1} = y$. Therefore, $(\Pi_{i+1} \cup V \cup \{\leftarrow not\ a_{i+1} = y\})^W = (\Pi_{i+1} \cup V)^W$.Therefore, $(W \setminus X) \subsetneq W$ satisfies all the rules of $(\Pi_{i+1} \cup V \cup \{\leftarrow not\ a_{i+1} = y\})^W$, which contradicts the fact that W is a possible world of $(\Pi_{i+1} \cup V \cup \{\leftarrow not\ a_{i+1} = y\})$.
3. Suppose W is the possible world of $\Pi_{i+1} \cup V \cup \{a_{i+1} = y\}$. We show that $X = \emptyset$. For the sake of contradiction suppose $X \neq \emptyset$. We previously showed that $W \setminus X$ satisfies the rules of Π^W_{i+1}. Since $V \subseteq W \setminus X$, $W \setminus X$ satisfies V. Since W is a possible world of $\Pi_{i+1} \cup V \cup \{a_{i+1} = y\}$, W satisfies $a_{i+1} = y$. Since $a_{i+1} \in L_{i+1} \setminus L_i$, and X consists of literals from L_i, $W \setminus X$ satisfies $a_{i+1} = y$. Therefore, $W \setminus X$ satisfies $(\Pi_{i+1} \cup V \cup \{a_{i+1} = y\})^W$, which contradicts the fact that W is a possible world of $(\Pi_{i+1} \cup V \cup \{a_{i+1} = y\})$.

Lemma 15 Let i be an integer in the range $\{0..k-1\}$ and V be a possible world of Π_i.

1. if no random selection rule with a_{i+1} is active in V, then every possible world of $\Pi_{i+1} \cup V$ is a possible world of Π_{i+1},
2. if there is a random selection rule of the form

$$random(a_{i+1}, p) \leftarrow B \tag{176}$$

active in V, and $p(y) \in V$, then every possible world of $\Pi_{i+1} \cup V \cup \{\leftarrow not\ a_{i+1} = y\}$, is a possible world of Π_{i+1}.

□

Proof Let Π_{ext} denote the program $\Pi_{i+1} \cup V$ in case 1 and and the program $\Pi_{i+1} \cup V \cup \{\leftarrow not\ a_{i+1} = y\}$ in case 2. Let W be a possible world of Π_{ext}. We show that W is a possible world of Π_{i+1}. We first show that W satisfies the rules of Π_{i+1}^W, and then we show that W is minimal.

1. We show that W satisfies the rules of Π_{i+1}^W. Let r be a rule of Π_{i+1}^W such that W satisfies the body of r. We need to show that W satisfies the head of r. Since W is a possible world of Π_{ext}, it satisfies the rules of Π_{ext}^W, which include the rules in Π_{i+1}^W.
2. We show that W is minimal. That is, there does not exist an interpretation W' such that W' satisfies the rules of Π_{i+1}^W and $W' \subsetneq W$. We prove by contradiction. Suppose such W' exists. We show that W' satisfies the rules of Π_{ext}^W, obtaining a contradiction to the fact that W is a possible world of Π_{ext}. By definition, W' satisfies the rules of Π_{i+1}^W. Therefore, since, in the second case of the Lemma $\{\leftarrow not\ a_{i+1} = y\}^W = \emptyset$, we just need to show that

$$W' \text{ satisfies } V^W \tag{177}$$

Since V is a collection of facts, we just need to show that $V \subset W'$. We prove by contradiction.
Suppose there is an atom $a = y$ of Π such that

$$a = y \in V \tag{178}$$

and

$$a = y \notin W' \tag{179}$$

Let us define V' to be:

$$V' = W'|_{L_i} \tag{180}$$

From (179) and (180) we have:

$$a = y \notin V' \tag{181}$$

By lemma 14 we have

$$W \setminus V \text{ does not contain literals from } L_i \tag{182}$$

Therefore, since V is a possible world of Π_i, we have:

$$V = W|_{L_i} \tag{183}$$

Since $W' \subsetneq W$, from (183) and (180) we have:

$$V' \subseteq V \tag{184}$$

From (184), (178) and (181) we have:

$$V' \subsetneq V \tag{185}$$

We next show

$$V' \text{ satisfies the rules of } \Pi^V \tag{186}$$

From (182) we have:

$$\Pi_i^V \subseteq \Pi_{i+1}^W \tag{187}$$

Let r be a member of Π_i^V such that V' satisfies the body of r. We show that V' satisfies the head of r. From (180) we have that the body of r is satisfied by W'.
Since W' satisfies the rules of Π_{i+1}^W, from (187) W' satisfies the head of r. Since r is a member of Π_i^V, $head(r) \in L_i$. Therefore, from (180), V' satisfies $head(r)$. Therefore, (186) holds, and, considering (184), we have a contradiction.

Lemma 16 For every $i \in \{0, \ldots, k\}$ and every leaf node n of T_i program Π_i has a unique possible world W satisfying $p_{T_i}(n)$. □

Proof We use induction on i. The case where $i = 0$ follows from Condition 2 (a) of Definition 6 of dynamically causally ordered program. Assume that the lemma holds for $i-1$ and consider a leaf node n of T_i. By construction of T, there exists a leaf node m of T_{i-1} which is either the parent of n or equal to n. By inductive hypothesis there is a unique possible world V of Π_{i-1} containing $p_{T_{i-1}}(m) \setminus \{true\}$.

(i) First we will show that every possible world W of Π_i containing $p_{T_{i-1}}(m)$ also contains V. By lemma 11, set $V' = W|_{L_{i-1}}$ is a possible world of Π_{i-1}. Obviously, $p_{T_{i-1}}(m) \setminus \{true\} \subseteq V'$. By inductive hypothesis, $V' = V$, and hence $V \subseteq W$.
Now let us consider two cases.

(ii) For every random selection r rule of Π of the form

$$random(a_i : \{X : p(X)\}) \leftarrow K \tag{188}$$

V falsifies K. We will show that in this case m is not ready to branch on a_i w.r.t Π_i. It is sufficient to show that for every random selection rule of the form (188), V is not Π_i-compatible with K. Since V falsifies K, there exists an e-literal $l \in K$ such that:

$$V \text{ does not satisfy } l \tag{189}$$

$$l \in L_i \tag{190}$$

Let us show by contradiction. Suppose there exists a possible world W of Π_i such that

$$W \text{ satisfies } V \cup K \tag{191}$$

By lemma 11 we have:

$$(W \setminus V) \cap L_i = \emptyset \tag{192}$$

From (189) , (190) and (192) we have:

$$W \text{ does not satisfy } l \tag{193}$$

Therefore, since $l \in K$, we have a contradiction to (191). Therefore, m is not ready to branch on a_i w.r.t Π_i, and, by construction of T_i, $m = n$. By condition 3 of Definition 6, we have $V \cup \Pi_{i-1}$ has exactly one possible world, W. By lemma 15, W is a possible world of Π_i. Obviously, W contains V and hence $p_{T_{i-1}}(m)$. Since $n = m$ this implies that W contains $p_{T_i}(n)$.

Uniqueness follows immediately from (i) and Condition (3) of Definition 6.

(iii) There is a random selection rule r of the form (188) active in V.

We will show that m is ready to branch on a_i via rule r relative to Π.

Condition (1) of the definition of "ready to branch" (Definition 18) follows immediately from construction of T_{i-1}.

To prove Condition (2) we need to show that $p_{T_{i-1}}(m)$ Π_i-guarantees K. Since r is active in V, by Condition 2 (b) of Definition 6 we have that there exists y_0 such that $p(y_0) \in V$ and $V \cup \Pi_{i+1}$ has a possible world containing $a = y_0$, say, W_0. Since r is active in V,

$$V \text{ satisfies } K \tag{194}$$

Therefore, by lemma 14, W_0 satisfies K. Since V contains $p_{T_{i-1}}(m)$, W_0 also contains $p_{T_{i-1}}(m)$. Therefore,

$$\text{V is } \Pi_i\text{-compatible with } K \tag{195}$$

Now consider a possible world W of Π_i which contains $p_{T_{i-1}}(m)$. By (i) we have that $V \subseteq W$. Since V satisfies K so does W (by lemma 11, $(W \setminus V) \cap L_{i-1} = \emptyset$). Condition (2) of the definition of ready to branch is satisfied.

To prove condition (3) consider $pr(a_i = y \mid B) = v$ from Π_i such that B is Π_i-compatible with $p_{T_{i-1}}(m)$. Π_i-compatibility implies that there is a possible world W_0 of Π_i which contains both, $p_{T_{i-1}}(m)$ and B. By (i) we have that $V \subseteq W_0$. By Lemma 11 we have that $(W \setminus V) \cap L_{i-1} = \emptyset$. From condition (1) of 6 it follows that either V satisfies B or V falsifies B. If V falsifies B, then W_0 does not satisfy B. Hence, V satisfies B. Since for every possible world W' of Π_i containing $p_{T_{i-1}}(m)$ we have that W' contains V and, by Lemma 11 $(W \setminus V) \cap L_{i-1} = \emptyset$, we have that W' satisfies B which proves condition (3) of the definition.

To prove Condition (4) we consider $y_0 \in range(a_i)$ such that $p(y_0) \in V$ (The existence of such y_0 is proven at the beginning of (iii)). We show that $p_{T_{i-1}}(m)$ Π_i-guarantees $p(y_0)$. Condition (2) of Definition 6 guarantees that Π_i has possible world, say W, containing V. By construction, $p(y_0) \in V$ and hence $p(y_0)$ and $p_{T_{i-1}}(m)$ are Π_i compatible. From (i) we have that $p_{T_{i-1}}(m)$ Π_i-guarantees $p(y_0)$. Similar argument shows that if $p_{T_{i-1}}(m)$ is Π_i-compatible with $p(y)$ then $p(y)$ is also Π_i-guaranteed by $p_{T_{i-1}}(m)$.

We can now conclude that m is ready to branch on a_i via rule r relative to Π_{i+1}. This implies that a leaf node n of T_i is obtained from m by expanding it by an atom $a_i = y$.

By Condition (2) of Definition 6, program $V \cup \Pi_i \cup \{\leftarrow not\ a_i = y_0\}$ has exactly one possible world, W. By lemma 15 we have that W is a possible world of Π_i. Clearly W contains $p_{T_i}(n)$. Uniqueness follows immediately from (i) and Condition (2) of Definition 6.

Lemma 17 For all $i \in \{0..k\}$, every possible world of Π_i satisfies $p_{T_i}(n)$ for some unique leaf node n of T_i. □

Proof We use induction on i. The case where $i = 0$ is immediate. Assume that the lemma holds for $i-1$, and consider a possible world W of Π_i. By Lemma 11, Π_{i-1} has a possible world V such that:

$$V \subseteq W \tag{196}$$

$$(W \setminus V) \cap L_{i-1} = \emptyset \tag{197}$$

By the inductive hypothesis there is a unique leaf node m of T_{i-1} such that V contains $p_{T_{i-1}}(m)$. Consider two cases.

(a) For every random selection rule

$$random(a_i : \{X : p(X)\}) \leftarrow K \tag{198}$$

K is falsified by V. In part ii of the proof of Lemma 16 we have shown that in this case m is not ready to branch on a_i. This means that m is a leaf of T_i and $p_{T_{i-1}}(m) = p_{T_i}(m)$. Let $n = m$. Since $V \subseteq W$ we have that $p_{T_i}(n) \subseteq W$. To show uniqueness suppose n' is a leaf node of T_i such that $p_{T_i}(n') \subseteq W$, and n' is not equal to n. By construction of T_i there is some j and some $y_1 \neq y_2$ such that $a_j = y_1 \in p_{T_i}(n')$ and $a_j = y_2 \in p_{T_i}(n)$. Since W is an interpretation, it is impossible.

(b) There is a random selection rule r of the form

$$random(a_i : \{X : p(X)\}) \leftarrow K \tag{199}$$

active in V. Since r is active in V, we have

$$V \text{ satisfies } K \tag{200}$$

and, therefore

$$\text{every literal from } K \text{ is in } L_{i-1} \tag{201}$$

From (200), (201), (196) and (197) we have:

$$W \text{ satisfies } K \tag{202}$$

From clause 2 of Definition 6 and the fact that $a_i \in L_i$ we have:

$$r \in \Pi_i \tag{203}$$

Since W is a possible world of Π_i, it must satisfies r together with a general axiom from Π_i:

$$a_i = y_1 \mid \ldots \mid a_i = y_m \leftarrow random(a_i : \{X : p(X)\}) \\ \leftarrow a_i = Y, not\ p(Y)$$

Therefore, since W satisfies the body of r, there exists $y \in range(a)$ such that

$$a_i = y \in W \tag{204}$$

and

$$p(y) \in W \tag{205}$$

Since r is active in V, by clause 2 of Definition 6 we must have $p(y) \in L_{i-1}$. From (205) and (197) we have:

$$p(y) \in V \tag{206}$$

Repeating the argument from part (iii) of the proof of Lemma 16 we can show that m is ready to branch on a_i via r relative to Π_i. Since $p_{T_{i-1}}(m) \subseteq V \subseteq W$, $p_{T_{i-1}}(m)$ is Π_i-compatible with $p(y)$. Thus, there is a leaf node n of T_i which is a son of m labeled with $a_i = y$. It is easy to see that W contains $p_{T_k}(n)$. The proof of uniqueness is similar to that used in (a).

Lemma 18 Let i, j be integers s.t. $0 < i \leq j \leq k$. For every leaf node n of T_{i-1}, every set B of extended literals of L_i, and we have $p_{T_{i-1}}(n)$ is Π_i-compatible with B iff $p_{T_{i-1}}(n)$ is Π_j-compatible with B.

□

Proof $\rightarrow$
Suppose that $p_{T_{i-1}}(n)$ is Π_i-compatible with B. This means that there is a possible world V of Π_i which satisfies $p_{T_{i-1}}(n)$ and B. By Lemma 17, there exists a unique leaf node n' of T_i such that $p_{T_i}(n') \setminus \{true\} \subseteq V$. Consider a leaf node m of T_j belonging to a path containing node n' of T_i. By Lemma 16, Π_j has a unique possible world W containing $p_{T_j}(m)$. By lemma 11 $W = V' \cup U$ where V' is a possible world of Π_i and $U \cap L_i = \emptyset$. This implies that V' contains $p_{T_i}(n')$, and hence, by Lemma 16 $V' = V$. Since V satisfies B and $U \cap L_i = \emptyset$ we have that W also satisfies B. Since $p_{T_{i-1}}(n) \subseteq V \subseteq W$, we have $p_{T_{i-1}}(n)$ is Π_j-compatible with B.

$\leftarrow$
Let W be a possible world of Π_j satisfying $p_{T_{i-1}}(n)$ and B. By Lemma 11, we have that $W = V \cup U$ where V is a possible world of Π_i and $U \cap L_i = \emptyset$. Since B and $p_{T_{i-1}}(n)$ belong to the language of L_i we have that B and $p_{T_{i-1}}(n)$ are satisfied by V and hence $p_{T_{i-1}}(n)$ is Π_i-compatible with B.

Lemma 19 Let i, j be integers such that $0 < i \leq j \leq k$. For every leaf node n of T_{i-1}, every set B of extended literals of L_i, we have $p_{T_{i-1}}(n)$ Π_i-guarantees B iff $p_{T_{i-1}}(n)$ Π_j-guarantees B. □

Proof $\rightarrow$
Let us assume that $p_{T_{i-1}}(n)$ Π_i-guarantees B. This implies that $p_{T_{i-1}}(n)$ is Π_i-compatible with B, and hence, by Lemma 18 $p_{T_{i-1}}(n)$ is Π_j-compatible with B. Now let W be a possible world of Π_j satisfying $p_{T_i}(n)$. By Lemma 11 $W = V \cup U$ where V is a possible world of Π_i and $U \cap L_i = \emptyset$. This implies that V satisfies $p_{T_{i-1}}(n)$. Since $p_{T_{i-1}}(n)$ Π_i-guarantees B we also have that V satisfies B. Finally, since $U \cap L_i = \emptyset$ we can conclude that W satisfies B.

$\leftarrow$
Suppose now that $p_{T_{i-1}}(n)$ Π_j-guarantees B. This implies that $p_{T_{i-1}}(n)$ is Π_i-compatible with B. Now let V be a possible world of Π_i containing $p_{T_{i-1}}(n)$. By Lemma 17, there exists a unique leaf node n' of T_i such that

$$V \text{ satisfies } p_{T_i}(n') \tag{207}$$

To show that V satisfies B let us consider a leaf node m of a path of T_j containing n'. By Lemma 16 Π_j has a unique possible world W satisfying $p_{T_j}(m)$. By construction,

$$W \text{ satisfies } p_{T_i}(n') \tag{208}$$

By Lemma 11, $W = V' \cup U$ where V' is a possible world of Π_i and $U \cap L_i = \emptyset$. Since $p_{T_i}(n')$ is in L_i, we have:

$$V' \text{ satisfies } p_{T_i}(n') \tag{209}$$

From (207) and (209) we by Lemma 16 we have:

$$V = V' \tag{210}$$

Since $p_{T_{i-1}}(n) \setminus \{true\} \subseteq V = V' \subseteq W$, we have W satisfies $p_{T_{i-1}}(n)$. Therefore, since $p_{T_{i-1}}(n)$ Π_j-guarantees B, W satisfies B. Since B belongs to the language of L_i it is satisfied by V'. Therefore, from $V' = V$ we have that V satisfies B and we conclude $p_{T_{i-1}}(n)$ Π_i-guarantees B.

Lemma 20 Let i, j be integers such that $0 < i \leq j \leq k$. Every leaf node n of T_{i-1}, n is ready to branch on term a_i relative to Π_i iff n is ready to branch on a_i relative to Π_j. □

Proof $\rightarrow$
Suppose n is ready to branch on a_i via rule r

$$random(a_i : \{X : p(X)\}) \leftarrow K \tag{211}$$

relative to Π_i. We show that n is ready to branch on a_i via r relative to Π_j. We prove the conditions 1-4 of the definition:

1. Condition 1 follows immediately from the fact that n is ready to branch on a_i relative to Π_i.

2. We prove condition 2:

$$p_{T_{i-1}}(n) \ \Pi_j\text{-guarantees } K \tag{212}$$

By lemma 16, there is a unique possible world W_{i-1} of Π_{i-1} such that

$$W_{i-1} \text{ satisfies } p_{T_{i-1}}(n) \tag{213}$$

We prove that

$$r \text{ is active in } W_{i-1} \tag{214}$$

We prove by contradiction. Suppose r is not active in W_{i-1}. By condition 1 of 6 we have W_{i-1} falsifies K. That is, there is a literal $l \in L_{i-1}$ such that W_{i-1} does not satisfy l. Then, by conditions 2-3 of 6 and Lemma 15 we have that Π_i has a possible world W_i containing W_{i-1}. By Lemma 11, $W_i \setminus W_{i-1} \cap L_{i-1} = \emptyset$. Therefore, W_i does not satisfy l, and, therefore, K. This, given that $p_{T_{i-1}}(n) \setminus \{true\} \subseteq W_{i-1} \subseteq W_i$ contradicts the fact n is ready to branch on a_i via r relative to Π_i. Therefore, (214) holds. Therefore, the literals occurring in the body of r are from L_{i-1}, and by lemma (19) we conclude (212).

3. We prove condition 3. Let $pr(a_i = y \mid B) = v$ be a pr-atom from Π_j. We show that

$$p_{T_{i-1}}(n) \ \text{ either } \Pi_j\text{-guarantees } B \text{ or is } \Pi_j\text{-incompatible with } B \tag{215}$$

Since n is ready to branch on a_i via rule r relative to Π_i, we have 3 cases:

(a) $pr(a_i = y \mid B) = v$ is a pr-atom from Π_i, and B is Π_i-guaranteed by $p_{T_{i-1}}(n)$. Using the arguments similar to the ones from 2, we can obtain $B \in L_{i-1}$, and conclude by lemma (19) that $p_{T_{i-1}}(n) \ \Pi_j$-guarantees B

(b) $pr(a_i = y \mid B) = v$ is a pr-atom from Π_i, and B is Π_i-incompatible with $p_{T_{i-1}}(n)$. That is,

$$\text{every possible world } W_i \text{ or } \Pi_i \text{ satisfying } p_{T_{i-1}}(n) \text{ does not satisfy } B \tag{216}$$

By lemma 16, there is a unique possible world W_{i-1} of Π_{i-1} such that

$$W_{i-1} \text{ satisfies } p_{T_{i-1}}(n) \tag{217}$$

We prove

$$W_{i-1} \text{ falsifies } B \tag{218}$$

For the sake of contradiction, suppose (218) is false. By condition 1 of definition (6) we have:

$$W_{i-1} \text{ satisfies } B \tag{219}$$

By 6 and lemma (15) we have that Π_i has a possible world W_i containing W_{i-1}. Since B is in L_i, by lemma (11) we have

$$W_i \text{ satisfies } B \tag{220}$$

Since $p_{T_{i-1}}(n) \setminus \{true\} \subseteq W_{i-1} \subseteq W_i$, we have a contradiction from (220) and (216).

Therefore, (218) holds. Now let W_j be a possible world of Π_j satisfying $p_{T_{i-1}}(n)$. By lemma (11), There is a possible W'_{i-1} of Π_{i-1} such that $W'_{i-1} \subseteq W_j$ and

$$(W'_{i-1} \cap W_j) \cap L_{i-1} = \emptyset \tag{221}$$

Since $p_{T_{i-1}}(n)$ is in L_{i-1}, we have $p_{T_{i-1}}(n) \subseteq W'_{i-1}$. Therefore, By lemma 16

$$W'_{i-1} = W_{i-1} \tag{222}$$

From (222), (218), (221) we have that W_j does not satisfy B. Therefore, $p_{T_{i-1}}(n)$ is Π_j-incompatible with B

(c) $pr(a_i = y \mid B) = v$ does not belong to Π_i. That is,
B contains an e-literal l

$$l \notin L_i \tag{223}$$

By lemma 16, there is a unique possible world W_{i-1} of Π_{i-1} such that

$$W_{i-1} \text{ satisfies } p_{T_{i-1}}(n) \tag{224}$$

Since B has l s.t. (223), W_{i-1} cannot satisfy B. Therefore by condition 1 of definition (6),

$$W_{i-1} \text{ falsifies } B \tag{225}$$

Similarly to 2, given (225), we can show that every possible world W_j satisfying $p_{T_{i-1}}(n)$ does not satisfy B, which implies $p_{T_{i-1}}(n)$ is Π_j-incompatible with B

4. We prove condition 4. By lemma 16, there is a unique possible world W_{i-1} of Π_{i-1} such that:

$$W_{i-1} \text{ satisfies } p_{T_{i-1}}(n) \tag{226}$$

As in 1, we can show that r is active in W_{i-1}. Therefore, by condition 2 of 6, we have that every atom $p(y)$ s.t. $y \in range(a_i)$ belongs to L_{i-1}. Therefore, condition 4 immediately follows from the fact n is ready to branch on a_i via rule r relative to Π_i and lemmas (18), (19).

$\leftarrow$

Now suppose n is ready to branch on a_i via rule r

$$random(a_i : \{X : p(X)\}) \leftarrow K \tag{227}$$

relative to Π_j. We show that n is ready to branch on a_i via r relative to Π_i. We prove the conditions 1-4 of the definition:

1. Condition 1 follows immediately from the fact that n is ready to branch on a_i relative to Π_i.

2. We prove Condition 2:

$$p_{T_{i-1}}(n) \ \Pi_i\text{-guarantees } K \tag{228}$$

By lemma 16, there is a unique possible world W_{i-1} of Π_{i-1} such that

$$W_{i-1} \text{ satisfies } p_{T_{i-1}}(n) \tag{229}$$

We prove that

$$r \text{ is active in } W_{i-1} \tag{230}$$

We prove by contradiction. Suppose r is not active in W_{i-1}. By condition 1 of 6 we have W_{i-1} falsifies K. That is, there is a literal $l \in L_{i-1}$ such that W_{i-1} does not satisfy l. Then, by conditions 2-3 of 6 and Lemma 15 we have that Π_j has a possible world W_j containing W_{i-1}. By Lemma 11, $W_j \setminus W_{i-1} \cap L_{i-1} = \emptyset$. Therefore, W_j does not satisfy l, and, therefore, K. This, given that $p_{T_{i-1}}(n) \setminus \{true\} \subseteq W_{i-1} \subseteq W_j$ contradicts the fact n is ready to branch on a_i via r relative to Π_j. Therefore, (230) holds. Therefore, the literals occurring in the body of r are from L_{i-1}, and by lemma (19) we conclude (228).

3. We prove condition 3. Let $pr(a_i = y \mid B) = v$ be a pr-atom from Π_i. We show that

$$p_{T_{i-1}}(n) \quad \text{either } \Pi_i\text{-guarantees } B \text{ or is } \Pi_i\text{-incompatible with } B \tag{231}$$

Since n is ready to branch on a_i via rule r relative to Π_j, we have 2 cases:

(a) B is Π_j-guaranteed by $p_{T_{i-1}}(n)$. Using the arguments similar to the ones from 2, we can obtain $B \in L_{i-1}$, and conclude by Lemma 19 that $p_{T_{i-1}}(n)$ Π_j-guarantees B

(b) B is Π_j-incompatible with $p_{T_{i-1}}(n)$. That is,

$$\text{every possible world } W_j \text{ or } \Pi_j \text{ satisfying } p_{T_{i-1}}(n) \text{ does not satisfy } B \tag{232}$$

By lemma 16, there is a unique possible world W_{i-1} of Π_{i-1} such that

$$W_{i-1} \text{ satisfies } p_{T_{i-1}}(n) \tag{233}$$

We prove

$$W_{i-1} \text{ falsifies } B \tag{234}$$

For the sake of contradiction, suppose (218) is false. By condition 1 of definition (6) we have:

$$W_{i-1} \text{ satisfies } B \tag{235}$$

By 6 and lemma (15) we have that Π_j has a possible world W_j containing W_{i-1}. Since B is in L_i, by lemma (11) we have

$$W_j \text{ satisfies } B \tag{236}$$

Since $p_{T_{i-1}}(n) \setminus \{true\} \subseteq W_{i-1} \subseteq W_i$, we have a contradiction from (236) and (232).

Therefore, (234) holds. Now let W_i be a possible world of Π_i satisfying $p_{T_{i-1}}(n)$. By lemma (11), There is a possible W'_{i-1} of Π_{i-1} such that $W'_{i-1} \subseteq W_j$ and

$$(W'_{i-1} \cap W_i) \cap L_{i-1} = \emptyset \tag{237}$$

Since $p_{T_{i-1}}(n)$ is in L_{i-1}, we have $p_{T_{i-1}}(n) \subseteq W'_{i-1}$. Therefore, By lemma 16

$$W'_{i-1} = W_{i-1} \tag{238}$$

From (238), (234), (2211) we have that W_i does not satisfy B. Therefore, $p_{T_{i-1}}(n)$ is Π_i-incompatible with B

4. As in 1, we can show that r is active in W_{i-1}. Therefore, by condition 2 of 6, we have that every atom $p(y)$ s.t. $y \in range(a_i)$ belongs to L_{i-1}. Therefore, condition 4 immediately follows from the fact n is ready to branch on a_i via rule r relative to Π_j and lemmas (18), (19).

Lemma 21 $T = T_k$ is a tableau for $\Pi \setminus U = \Pi_k$.

□

Proof Follows immediately from the construction of the T's and Π's, the definition of a tableau, and Lemmas 20 and 18. □

Lemma 22 $T = T_k$ represents $\Pi \setminus U = \Pi_k$.

□

Proof Let W be a possible world of Π. By Lemma 17 W contains $p_T(n)$ for some unique leaf node n of T. By Lemma 16, W is the set of literals Π-guaranteed by $p_T(n)$, and hence W is represented by n. Suppose now that n' is a node of T representing W. Then $p_T(n')$ Π-guarantees W which implies that W contains $p_{T_m}(n')$. By Lemma 17 this means that $n = n'$, and hence we proved that every answer set of Π is represented by exactly one leaf node of T.

Now let n be a leaf node of T. By Lemma 16 Π has a unique possible world W containing $p_T(n)$. It is easy to see that W is the set of literals represented by n. □

Lemma 23 Suppose T is a tableau representing Π. If n is a node of T which is ready to branch on $a(\bar{t})$ via r, then all possible worlds of Π compatible with $p_T(n)$ are probabilistically equivalent with respect to r.

□

Proof This is immediate from Conditions (3) and (4) of the definition of ready-to-branch.

Notation: If n is a node of T which is ready to branch on $a(\bar{t})$ via r, the Lemma 23 guarantees that there is a unique scenario for r containing all possible worlds compatible with $p_T(n)$. We will refer to this scenario as *the scenario determined by* n.

Lemma 24 $T = T_m$ is unitary

□

Proof We need to show that for every node n of T, the sum of the labels of the arcs leaving n is 1. Let n be a node and let s be the scenario determined by n. s satisfies (1) or (2) of the Definition of a unitary rule from [8]. In case (1) is satisfied, the definition of $v(n, a(\bar{t}), y)$, along with the construction of the labels of arcs of T, guarantee that the sum of the labels of the arcs leaving n is 1. In case (2) is satisfied, the conclusion follows from the same considerations, along with the definition of $PD(W, a(\bar{t}) = y)$.

Lemma 25 $T = T_m$ is a probabilistically sound representation of $\Pi \setminus U$.

Proof Let R be a mapping from the possible worlds of $\Pi \setminus U$ to the leaf nodes of T which represent them. We need to show that for every possible world W of $\Pi \setminus U$ we have

$$v_T(R(W)) = \mu(W). \tag{239}$$

By definition of μ, we have:

$$\mu(W) = \frac{\hat{\mu}(W)}{\sum_{W_i \in \Omega(\Pi \setminus U)} \hat{\mu}(W_i)} \tag{240}$$

where

$$\hat{\mu}(W) = \prod_{W(a)=y} P(W, a = y) \tag{241}$$

where the product is taken over atoms for which $P(W, a = y)$ is defined.

By lemma 24, T is a unitary tree. Therefore, by lemma 4 we have that the sum of path values of it's leaves is 1. Therefore, it is sufficient to show that for every possible world W of $\Pi\setminus$

$$v_T(R(W)) = \hat{\mu}(W). \tag{242}$$

To prove 242, it is sufficient to show that for every possible world W of $\Pi \setminus U$ (1) $p_T(R(W))$ contains an atom $a = y$ if and only if $a = y \in W$ and $P(W, a = y)$ is defined, (2) if n is a node in the path P of T from its root to $R(W)$ which branches on a, then the probability assigned to the arc which goes from n to its child in P, $v(n, a, y)$ is equal to $P(W, a = y)$.

1) $\Rightarrow$ We first show that if $p_T(R(W))$ contains an atom $a = y$, then $P(W, a = y)$ is defined and $W(a) = y$.
By definition of $P(W, a = y)$, it is defined if and only if there exists a rule of Π of the form

$$random(a : \{X : p(X)\}) \leftarrow K \tag{243}$$

such that W satisfies K, $truly\_random(a)$ and $p(y)$.
By definition of T, if $a = y$ belongs to $p_T(R(W))$, there must exist a node n in the path from the root of T to $R(W)$ such that n branches on a via some rule r of the form (243) of Π. This means that $p_T(n)$ $\Pi \setminus U$-guarantees K and $p(y)$. By construction $p_T(R(W))$ contains $p_T(N)$, thus, $p_T(R(W))$ $\Pi \setminus U$-guarantees the body of r and $p(y)$. Since $R(W)$ represents W, W must contain all positive literals in $p_T(R(W))$. Therefore, W satisfies both K and $p(y)$. From rule 243 it follows that W satisfies $random(a : \{X : p(X)\})$, and, since $\Pi \setminus U$ does not contain activity records, W satisfies $truly\_random(a)$.

By definition of a tableau representing a program, $p_T(R(W))$ $\Pi \setminus U$-guarantees W. By lemma 17 and minimality of possible worlds, W contains $p_T(R(W))$. This, $W(a) = y$.

$\Leftarrow$ We show that if $P(W, a = y)$ is defined, and $W(a) = y$, then $p_T(R(W))$ contains an atom $a = y$. We prove by contradiction. Suppose $p_T(R(W))$ does not contain an atom $a = y$. There are two possible cases:

(a) $p_T(R(W))$ contains an atom $a = y_1$, where $y_1 \neq y$. By definition of a tree representing a possible world, $p_T(R(W))$ $\Pi \setminus U$-guarantees W. By lemma 17 and minimality of possible worlds (proposition 1), we have that

$$W \text{ satisfies } p_T(R(W)) \tag{244}$$

Therefore, $W(a)$ contains both $a = y_1$ and $a = y$, which is impossible by definition of an interpretation.

(b) $p_T(R(W))$ contains no literal of the form $a = y_1$ for any y_1. In this case, using minimality of possible worlds, it is easy to see that $R(W)$ is ready to branch on a, which contradicts the definition of a tableau.

2) We show that if n is a node in the path P of T from its root to $R(W)$ which branches on a, then the probability assigned to the arc which goes from n to its child in P, $v(n, a, y)$, is equal to $P(W, a = y)$. By definition of $v(n, a, y)$, we only need to show that W is $\Pi \setminus U$-compatible with $p_T(n)$. Since W is a possible world of $\Pi \setminus U$, it is sufficient to show that W contains $p_T(n) \setminus \{true\}$. From 244 we have that W contains $p_T(R(W)) \setminus \{true\}$. Since n is a node on the path P from the root of T to $R(W)$, $p_T(R(W))$ contains $p_T(n) \setminus \{true\}$. Therefore, W contains $p_T(n) \setminus \{true\}$.

Therefore, as shown by Lemmas 22, 25, and 24, T is a unitary probabilistically sound representation of $\Pi \setminus U$, that concludes the proof of Lemma 6.

We are now ready to prove the main theorem.

Theorem 1
Every dynamically causally ordered, unitary program is coherent.

Proof Suppose Π is dynamically causally ordered and U be the set of activity records of Π. Proposition 6 tells us that $\Pi \setminus U$ is represented by some tableau T. Lemmas 24 and 25 tells us that the tree is unitary and that the representation is probabilistically sound correspondingly. Thus, by Lemma 5 Π is coherent.