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

# P-log: Refinement 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 and slightly modify the formal semantics to better match the intuitive meaning of the language constructs. We also define a new class of coherent (i.e., logically and probabilistically consistent) P-log programs which facilitates 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) [18] and its extensions and probabilistic reasoning with Causal Bayesian Networks [38]. To facilitate reading of the paper we briefly describe the basics of the *original* language. For more details the reader should refer to [8]. An accurate description of the syntax and semantics of the variant of P-log proposed in this paper will be given in Section 3.

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

The basic building blocks of original P-log are atoms formed by arithmetic functions and relations, and atoms of the form $f(\bar{x}) = y$, where $f$ is a user defined function called an *attribute*[1] and $\bar{x}$ denotes the list $(x_1, \ldots, x_n)$ of its parameters; $f(\bar{x})$ is called an *attribute term*. Literals are expressions of the form $f(\bar{x}) = y$ and $f(\bar{x}) \neq y$. Literals preceded by default negation *not* are referred to as extended literals or simply e-literals. There are several types of P-log rules:

- *Regular rules* – usual non-disjunctive ASP rules built from these atoms,
- *Random selection rules* of the form

$$[\ r\ ]\ random(f(\bar{x}) : \{X : p(X)\}) \leftarrow body.$$

  Rule $r$ states that, given *body*, the value $y$ of attribute term $f(\bar{x})$ is selected at random among $f(\bar{x})$'s possible values satisfying property $p$; $p$ is often referred to as the *dynamic range* of $f(\bar{x})$. One can think of this selection as a result of a particular physical or mental experiment associated with the rule.
- *Activity records* – special facts of the form $obs(l)$, where $l$ is a literal, and $do(f(\bar{x}) = y)$. The former is used to record observations of the results of random selections. The latter describes a deliberate intervention which stops the process of random selection and simply assigns value $y$ to $f(\bar{x})$. The intuition behind the last construct is similar to that in [38]. As in Pearl's work it is used to reason about causality, counterfactuals, etc. in which the difference between actions and observations plays a critical role.
- *Causal probability statements* (also called *pr*-atoms) which are of the form $pr_r(f(\bar{x}) = y) = v$. A statement says that if the value of $f(\bar{x})$ was fixed by experiment $r$, then $y$ is selected as the value of $f(\bar{x})$ with probability $v$. P-log also allows causal probability statements of the form $pr(f(\bar{x}) = y|B) = v$. (Here $|$ stands for the causal stroke which in the original paper is denoted by $|_c$. The precise description of this is given in Section 3.) In the absence of causal probability statements, the probability of a selection is computed using *indifference principle*[2] which says that "possible outcomes of a random experiment are assumed to be equally probable if we have no reason to prefer one of them to any other."

Semantically, P-log program $\Pi$ defines a collection of possible worlds corresponding to beliefs of a rational agent associated with it, and probability measure defined on sets of these worlds. Probability function $P_\Pi(f(\bar{x}) = y)$ returns the degree of reasoner's belief in the value of $f(\bar{x})$ being $y$ which is defined as the (normalized) probability measure of all possible worlds in which $f(\bar{x})$ takes value $y$. Here is an example of a simple P-log program $\Pi_0$. (For the convenience of copying and running programs, we will use ASCII symbols `:-` and `-` instead of $\leftarrow$ and classical (strong) negation $\neg$ respectively).

```
a, b : boolean.
[r] random(a) :- -b.
-b :- not b.
a :- b.
```

[1] If $f$ is a boolean function we use $f(\bar{x})$ and -$f(\bar{x})$ as shorthands for $f(\bar{x}) = true$ and $f(\bar{x}) = false$ respectively.

[2] For a deeper discussion of this principle and its logical justification see, for instance, [25].

The first statement is a declaration of 0-arity boolean attributes $a$ and $b$. The second states that if $b$ is false, then the value of $a$ is generated at random. The *not* in the third rule is default negation of ASP. The rule states that if there is no reason to believe $b$, then $b$ should be false. The last rule says that a reasoner believing $b$ must also believe $a$. The program has two possible worlds: $W_1 = \{b = false, a = true\}$ and $W_2 = \{b = false, a = false\}$ corresponding to two possible assignment of values to $a$. By the indifference principle, $P_{\Pi_0}(a = true) = 0.5$. Since no possible world contains $b$, $P_{\Pi_0}(b = true) = 0$. Note, that program $\Pi_1 = \Pi_0 \cup \{b\}$ has a different possible world, $\{a = true, b = true\}$ and that $P_{\Pi_1}(a = true) = P_{\Pi_1}(b = true) = 1$. Not surprisingly, P-log, built on ASP, is logically non-monotonic. Learning $b$ forces the reasoner to withdraw its previous conclusion.

Another distinctive feature of P-log is its 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 [20] P-log was expanded to allow abductive reasoning in the style of CR-Prolog [6] and infinite possible worlds. In addition to being logically non-monotonic, this extended language is also “probabilistically non-monotonic” with respect to observations — an addition of new observations can add new possible worlds and substantially change the original probabilistic model. In the original P-log new observations can only eliminate possible worlds. A P-log prototype query answering system (based on ASP solvers) is described in Zhu’s dissertation [49]. 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 [49], development and automation of mathematical models of machine ethics [39], methods for combining probabilistic and logical reasoning in robotics [48], etc.

Despite these achievements, P-log is still lacking scalable reasoning algorithms and systems. This drawback was partially addressed in recent dissertation of the first author [1]. This work forced us to have a second look at the original version of P-log and to discover a number of small, but subtle and important issues with its original syntax and semantics. To address these issues, we

– Clarified and made explicit the treatment of P-log’s partial functions. Among other things, we explicitly defined the truth of $f(\bar{x}) \neq y$ to mean that $f(\bar{x})$ has the value and this value is different from $y$ and prohibited occurrences of statements of the form $f(\bar{x}) \neq y$ in the heads of P-log rules.
– Clarified syntax and semantics of observations and actions of P-log by making their formal semantics better reflect their informal meaning. The change led to reifying names of random selection rules of the program and using these names as parameters of *do* and *random*.

In addition, our work on P-log reasoning algorithms allowed us to realize the need for better sufficient condition for coherency of P-log programs.

Intuitively, program $\Pi$ with no activity records is coherent if (a) it is logically consistent, i.e. has at least one possible world, and (b) probabilistically consistent, i.e. defines probabilistic distribution $P_\Pi$ compatible with causal probabilities specified by pr-atoms of the program. Program $\Pi$ containing activity records is coherent if it is logically consistent and the program $\Pi'$, obtained from $\Pi$ by removing the activity records, is coherent.

Let us illustrate this by an example (for the precise definition, see [8]). Let $\Pi$ be the program

```
a : {0,1,2}.
random(a).
pr(a=0)=1/2.
pr(a=1)=1/2.
pr(a=2)=1/2.
```

Clearly, $\Pi$ has three possible worlds, $\{a = 0\}$, $\{a = 1\}$, and $\{a = 2\}$, corresponding to possible values of $a$. As expected, the definition of $P_\Pi$ gives $P_\Pi(a = 0) = P_\Pi(a = 1) = P_\Pi(a = 2) = 1/3$, which is different from causal probabilities defined by *pr*-atoms. The program is incoherent.

To be useful, P-log programs written by a programmer or a knowledge engineer should be coherent. To facilitate construction of such programs and proving their coherency, [8] defines a class of causally ordered, unitary programs and shows that such programs are coherent. Let us call this class $\mathcal{A}$. The definition of class $\mathcal{A}$ serves as a guideline for the design of coherent P-log programs. The corresponding theorem guarantees coherency of the final product. Unfortunately, we discovered that there are interesting and important coherent programs which do not belong to this class. In this paper we define a new class, $\mathcal{B}$, of *dynamically* causally ordered unitary programs, and show that such programs are coherent. Even though $\mathcal{B}$ is broad and contains all examples of "real" P-log programs we designed or found in the literature, it is not a generalization of $\mathcal{A}$. However, known examples of programs belonging to $\mathcal{A}$ and not belonging to $\mathcal{B}$ are highly artificial and, in our opinion unlikely to be found in practice. While it is possible to define a class of coherent programs containing both $\mathcal{A}$ and $\mathcal{B}$, the definition will be too complex and will be difficult to use as a guideline for constructing coherent programs. In addition to facilitating construction of coherent programs, these new results play an important role in the development of a new P-log reasoning algorithm [1]. Unfortunately the size of this material prevents us from including it in this paper, but we plan to soon publish it separately. Finally, we expanded the notion of coherency, class $\mathcal{B}$ and the coherency theorem to the CR-Prolog based version of P-log. We often refer to this language as *P-log with cr-rules* where "cr" stands for "consistency restoring".

The rest of the paper is organized as follows. Section 2 discusses the reasons for the proposed changes in P-log syntax and semantics. Section 3 defines the new syntax and semantics precisely. Section 4 contains the background necessary for the definition of the new class, $\mathcal{B}$, of coherent programs. Section 5 defines class $\mathcal{B}$ and states the main coherency theorem. Section 6 gives a number of examples of programs from this class which were not covered by previous results. Section 7 expands the notion of coherency to P-log with cr-rules, defines an extension, $\mathcal{B}^+$, of

$\mathcal{B}$ to this language and proves coherency of its programs. Section 8 discusses related work and Section 9 concludes the paper. The appendix given in this paper contains accurate definitions of the probability in the new version of P-log. Appendix B from the full version of the paper[3] contains the proof of coherency of programs from $\mathcal{B}$. The paper contains substantial amount of material not presented in [3]. It has more detailed and accurate description of the language, more examples, definition of class $\mathcal{B}$ and the corresponding coherency theorem. Moreover, all examples of programs given in the paper, not including those with cr-rules from CR-Prolog, were run on prototype implementation[4].

## 2 Discussion of the Proposed Changes

In this section we discuss the changes we decided to make in the informal and formal descriptions of P-log and give several examples relevant to these changes. The examples are meant to illustrate the changes and give an additional evidence of the power of P-log as a tool for knowledge representation and reasoning.

**Partial Functions**: The designers of the original P-log did not pay sufficient attention to the treatment of partial functions in the language. They were neither explicitly prohibited nor explicitly allowed. This oversight in the design of the original P-log led to 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 $\Pi$ consisting of the only rule:

$$a \neq false$$

where $a$ is declared as $a : boolean$.

There is no explicit definition of the meaning of the statement $f(\bar{x}) \neq y$ in the original P-log, but it is assumed to mean that $f(\bar{x})$ is defined, i.e. has a value, and this value is different from $y$. The intuition agrees with some other extensions of ASP with function symbols, e.g. [5,10,11]. Together with declaration $a : boolean$, this statement should imply that the value of $a$ is $true$. However, the program $\Pi$ has one possible world consisting of literal $a \neq false$ and hence $\Pi$ does not entail $a = true$. 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.

[3] https://github.com/iensen/plog2.0/raw/master/papers/plog_ref_dco_full.pdf

[4] 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. The implementation is available from https://github.com/iensen/plog2.0/wiki.

**P-log observations**: Another problem with the original P-log is related to the intuitive meaning of P-log observations. According to the original P-log [8], such observations are used "to record the outcomes of random events". However, axiom (12) from [8] does not faithfully reflect this intuition. The axiom 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 observation simply eliminates some of the possible worlds of the program, which reflects the understanding of observations in classical probability theory. This view is also compatible with the treatment of observations in action languages. So, our version of P-log allows observations of values of arbitrary attributes. This eliminates the necessity to precisely define random events, which may include "conditionally" random attributes – the attributes whose value is generated randomly in one possible world of a program and is deterministic in another one. Capturing this intuition is a non-trivial task. We were not able to find any adverse consequences of this decision.

The following example contains observations of values of attributes which are generated by random experiments as well as observations of values of attributes which are not generated in this manner. The example uses the version of P-log with cr-rules and, we believe, has some independent interest from the knowledge representation perspective.

*Example 1* [Indirect exceptions to defaults, observations, and "explaining away" phenomena] Consider a default *"Normally, healthy people do not have a slow heart rate"* represented by defeasible rule

$$\neg slow\_rate(P) \leftarrow healthy(P), not\ ab(P) \tag{1}$$

where $ab(P)$ (read as"abnormal" $P$) holds when the default is not applicable to the person in question. This is one of several ways to represent defaults discussed in, say, [16] where this rule is used together with a cr-rule

$$ab(P) \overset{+}{\leftarrow} \tag{2}$$

encoding possible *indirect exceptions* to this default. According to the semantics of CR-Prolog, rule (2) says that default (1) may not be applicable to $P$, but this is very rare and can be ignored unless needed for restoring consistency of the program. (More information on the semantics of CR-Prolog and on indirect exceptions to defaults can be found in [16]). If used together with

$$healthy(bob) \tag{3}$$

which states that Bob is healthy, the program will ignore cr-rule (2) and entail $\neg slow\_rate(bob)$.

Let us assume, in addition, that

– there are two possible causes for slow heart rate in healthy people: a person is physically active or is maintaining a special diet to lower blood pressure,
– according to some reliable statistics, eight out of ten healthy people with slow heart rate are physically active and two follow the special diet.

For simplicity, we assume that our causes are deterministic and represent the first statement by rules:

$$slow\_rate(P) \leftarrow active(P) \tag{4}$$

$$slow\_rate(P) \leftarrow diet(P) \tag{5}$$

(Non-deterministic causes can be represented by defaults or random selection rules.) To better understand the reasoning of CR-Prolog, it will be instructive to expand the program by an observation

$$obs(slow\_rate(bob)) \tag{6}$$

The observation is understood as a constraint eliminating the possibility of Bob not having a slow heart rate and hence contradicts the conclusion of the default. Rule 2, however, may prevent the default's application and hence eliminate this source of inconsistency. Notice, however, that program still can not derive that Bob's heart rate is slow. This can only be done by rules (4) and (5), but to use them the reasoner should look at possible causes of Bob's slow rate. There are, however, no rules allowing the reasoners to do that. Hence, the program written so far does not satisfy (6) and, therefore, is inconsistent. To remedy the problem, we should expand the program by information which would allow the reasoner to select possible causes of our observation. This can be naturally done in P-log by two random selection rules:

$$[\ ra(P)\ ]\ random(active(P)) \leftarrow ab(P) \tag{7}$$

$$[\ rd(P)\ ]\ random(diet(P)) \leftarrow ab(P) \tag{8}$$

The above mentioned statistics allowing the reasoner to assign probabilities to these causes is given by the corresponding causal probability statements:

$$pr_{ra(P)}(active(P)) = 0.8 \tag{9}$$

referring to rule (7) and

$$pr_{rd(P)}(diet(P)) = 0.3 \tag{10}$$

referring to (8). The program consisting of rules (1)–(10) has possible worlds:

$W_1 = \{ab, healthy, slow\_rate, active, \neg diet\}$
$W_2 = \{ab, healthy, slow\_rate, diet, \neg active\}$
$W_3 = \{ab, healthy, slow\_rate, active, diet\}$

(For readability we are omitting parameter "bob" from the literals). The probability measures of the possible worlds are approximately 0.65, 0.07 and 0.28, respectively.

Let us denote the resulting program by $\Pi_{obs}$. Then,

$$P_{\Pi_{obs}}(active) \approx 0.65 + 0.28 = 0.93 \tag{11}$$

and

$$P_{\Pi_{obs}}(diet) \approx 0.07 + 0.28 = 0.35. \tag{12}$$

As expected, *active* is the more likely cause of the observed slow heart rate of Bob. Let us now consider a case where some direct or indirect observations established that our patient is on the special diet. This is represented by a program $\Pi^2_{obs} = \Pi_{obs} \cup \{obs(diet(bob))\}$ which has two possible worlds: $W_2$ and $W_3$.

$$P_{\Pi^2_{obs}}(active) = \frac{0.8 \times 0.3}{0.3 \times 0.8 + 0.3 \times 0.2} = 0.8 \tag{13}$$

(a decrease from 0.93) and, clearly,

$$P_{\Pi^2_{obs}}(diet) = 1. \tag{14}$$

This is an example of so called *explaining away* phenomena (see, for instance, [47]): *if there are competing possible causes for some event, and the chances of one of those causes increases, the chances of the other cause must decline since it is being "explained away" by the first explanation.*

The example shows how the representation of a normal behavior of a system together with possible causes of system's deviation from this behavior can be used to do predictions based on normality assumptions as well as explanations of unexpected observations. We believe that the methodology it illustrates may be useful in various domains dealing with causal reasoning.

**P-log intervening actions**: Now we discuss a modification we made to informal and formal semantics of intervening actions. The original P-log represents such an intervention by statement $do(f(\bar{x}) = y)$ which indicates that "$f(\bar{x}) = y$ is made true as a result of a deliberate (non-random) action". Note that this formulation does not require $f(\bar{x})$ to be in any way connected to randomness. It does not have to be declared as random or depend on another random attribute term. This was, most likely, deliberate. The original intervening action *do* of Pearl, which served as a model of intervention in P-log, only applies to random variables (no other types are available in Bayesian Nets) and seems to be understood as intervention into a random process. One would expect the original P-log follow this model. This is, however, not entirely trivial. Unlike Bayesian nets, attributes of P-log may not be random. Moreover, the notion of randomness in the original P-log is conditional: the statement

$$[\ r\ ]\ random(f(\bar{x})) \leftarrow body$$

declares $f(\bar{x})$ to be random only when the body of the rule is satisfied. So it was not immediately clear how to express the restriction limiting application of *do* to random attributes only. Instead, *do* was made applicable to arbitrary attributes. This seemed harmless since, according to the original semantics, for non-random $f(\bar{x})$, $do(f(\bar{x}) = y)$ is (modulo *do*) equivalent to $f(\bar{x}) = y$. We now believe that this original impression was wrong. First, the decision allowed the same thought to be expressed by two different language constructs which 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 exactly one language construct.* Moreover, applying *do* to interfere with a random experiment in which the value of random $f(\bar{x})$ is selected from the collection of values satisfying some property $p$ causes an ambiguity of an intuitive interpretation: should the value deliberately assigned to $f(\bar{x})$ belong to its dynamic range or any value from the range of $f(\bar{x})$

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. These considerations led to the decision of limiting applications of *do* to random attributes. To express the corresponding constraint we slightly modified the notion of possible world by allowing possible worlds contain special atoms formed by *random* and did necessary changes in the set of general axioms of P-log as described in Section 3. At the first glance, this new understanding runs into a conflict with later work by J. Pearl in which intervention is done in the context of structural models (which are different from Bayesian nets). Such a model describes an autonomous mechanism which determines a value of one particular non-deterministic (endogenous) variable in terms of its immediate causes and random “disturbances” coming from outside the system. In this approach performing interventions on non-random variables becomes essential for answering a variety of important queries such as “The probability that event $B$ occurred because of event $A$” or “The probability that event $B$ would have been different if it were not for event $A$” (see, for instance, [23,24])

The following firing squad example, originally introduced in [37] to demonstrate reasoning with causal models, shows how this can be done in the new version of P-log.

*Example 2* Consider the following story: “*A captain may give an order to shoot the prisoner. The order causes guards A and B to shoot, and shooting causes death of a prisoner. The guard A is new to the job and can pull the trigger (and kill the prisoner) by accident.*” It is naturally represented in P-log by program $F$ with declarations:

$$guards = \{a, b\}$$

$$order, dead : boolean$$

$$pull, shoot : guards \rightarrow boolean$$

two “random selection rules” which tell that captain may or may not give an order and guard $A$ may or may not pull the trigger and that these events happen according to some random mechanisms:

$$[\ r_{order}\ ]\ random(order)$$

$$[\ r_a\ ]\ random(pull(a))$$

causal relations of the story represented by defaults:

$$shoot(G) \leftarrow order, not\ \neg shoot(G)$$

$$shoot(G) \leftarrow pull(G), not\ \neg shoot(G)$$

$$dead \leftarrow shoot(G), not\ \neg dead$$

read as “Normally, order causes guards to shoot, pulling the trigger causes shooting, and shooting causes death of the prisoner”,
and implicit assumptions about completeness of our causes which are present in the story implicitly:

$$\neg shoot(G) \leftarrow \neg order, \neg pull(G)$$
$$\neg dead \leftarrow \neg shoot(a), \neg shoot(b)$$

which say that "guards do not shoot except by order or by accidentally pulling the trigger and that prisoner does not die from natural causes". Finally, the story seems to assume that guard $B$ is experienced enough not to pull the trigger by accident, so we add

$$\neg pull(b)$$

This completes the construction of program $F$.

Note, that the representation of causal relations by defaults (instead of strict rules used, for instance, in the original translation of structural models into P-log [9]) is justified by the fact that usually such relations are not totally deterministic. After all, a guard can refuse the order or deliberately miss, and we may want to ask what would happen and with what probability if this were the case. It is exactly this representation which allows P-log to deal with complex conditional and counterfactual queries.

Let us use $F$ to show how this works by answering a query "*What is the probability that the prisoner is dead given that A is prevented*[5] *from shooting?*". To answer we simply need to compute probability, $P_{F \cup \{\neg shoot(a)\}}(dead)$, of *dead* with respect to program $F \cup \{\neg shoot(a)\}$. It is easy to check that $P_F(dead) = 0.75$, $P_{F \cup \{\neg shoot(a)\}}(dead) = 0.5$, and, therefore, $P_{F \cup \{\neg shoot(a)\}}(dead) < P_F(dead)$. Program $F$ can also be used to answer counterfactual queries using the technique developed in [9].

Notice that, unlike defaults, strict rules of P-log do not correspond to Pearl's mechanisms which can be modified by interventions. This is intentional. Rules, say,

$$alive(X) \leftarrow \neg dead(X)$$
$$\neg alive(X) \leftarrow dead(X)$$

can be viewed as a definition of *alive* in terms of *dead* and definitions, we believe, shall not be interfered with.

We hope that this example demonstrates how the richness of P-log can be used to give an accurate formal representation of an informal story, to capture nuances which will be difficult to express otherwise, and to answer complex queries. It allows us to prohibit application of *do* to non-random attribute terms and to stay as close as possible to the intuition of *do* as an intervention into (a physical or mental) random experiment.

## 3 Syntax and Semantics of the New Version 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 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.

[5] By removing bullets from his gun or by some other means.

## 3.1 Syntax and Informal Semantics

A *sorted signature* $\Sigma$ 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 function 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 sort $s$ and write $o \in s$. 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.
  We assume that $S$ contains universal sort $\mathbf{U}$ consisting of all object constants from $O$, sort $\mathbf{R}$ whose elements are called *rule names*, sort **boolean**, and sorts $\mathbf{N}$ and $\mathbf{Q}$ of standard representations of natural and rational numbers respectively.
- Every function symbol from $F$ has sort names assigned to its parameters and its range. In what follows we use standard mathematical notation
  $$f : s_1, \dots, s_n \to s$$
  to describe these assignments. We assume that for each sort name $s$, $F$ has function symbol $instance_s : \mathbf{U} \to \mathbf{boolean}$ (where $instance_s(o) = true$ iff $o \in s$).
- Set $F$ is partitioned into two parts: *attributes* and *arithmetic* functions such as $+$, $-$, etc., defined on natural or rational numbers. (The precise collection of arithmetic functions available to the users will depend on the language implementation).

Now we define terms and atoms of P-log signature $\Sigma$. Every such signature has three types of variable-free terms:

- Standard *arithmetic terms.*
- *Regular terms* - expressions of the form $f(\bar{x})$ where $f : s_1, \dots, s_n \to s$ is an attribute and $\bar{x}$ is a list of $n$ constants such that for every $i$, $x_i \in s_i$. We say that $s$ is the *range* of term $f(\bar{x})$. Sometimes we write it as $range(f(\bar{x}))$.
- *Special terms* listed below. Note that $f(\bar{x})$ denotes a regular term, $r$ is a rule name, $y$ is a constant from the range of $f(\bar{x})$, and $p$ is a unary boolean attribute.
  - $do(r, f(\bar{x}), y)$, which reads as "a random experiment $r$ assigning value to $f(\bar{x})$ is deliberately interfered with and $f(\bar{x})$ is assigned the value $y$".
  - $obs(f(\bar{x}), y, true)$, 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 $y$".[6]
  - $random(r, f(\bar{x}), p)$, which says that "$f(\bar{x})$ may take the value from the set $\{X : p(X)\} \cap range(f(\bar{x}))$ as the result of a random experiment $r$ which is either genuine or deliberately interfered with." For the sake of compatibility with original P-log, we will sometimes write $random(r, f(\bar{x}) : \{X : p(X)\})$ instead of $random(r, f(\bar{x}), p)$. In the former case, all occurrences of variable $X$ in $\{X : p(X)\}$ are *bound*. The parameter, $p$ of

[6] 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 $obs(f(\bar{x}))$.

*random* can be omitted, in which case $random(r, f(\bar{x}))$ is understood as $random(r, f(\bar{x}), instance_{range(f(\bar{x}))})$.

– $truly\_random(r, f(\bar{x}))$, which says that "$f(\bar{x})$ takes value as the result of the genuine random experiment $r$ (i.e., the one without any outside interference)".

The range of all special terms is **boolean**.

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

An *atom* (or *positive literal*) of signature $\Sigma$ 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*

A *negative literal* is an expression $\neg(t = y)$ (also written as $t \neq y$), which says that "$t$ has a value and it is different from $y$". If $t$ is boolean then $t = true$ and $t = false$ will often be written as $t$ and $\neg t$[7].

An atom of the form $t = y$ is called *special* if $t$ is a special term, otherwise it is called *regular*.

If $t$ is of one of the forms $obs(f(\bar{x}), z, true)$ or $obs(f(\bar{x}), z, false)$, the atom is called an *observation*. If $t$ is of the form $do(r, f(\bar{x}), z)$, the atom is called an *action* (or an *intervention*).

A *literal* of $\Sigma$ is either a positive literal of $\Sigma$ or a negative literal of $\Sigma$. A literal, possibly preceded by the default negation *not*, is called an *extended literal* or simply an *e-literal* of $\Sigma$. 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"). Sometimes we use atoms($\Sigma$) and e-lits($\Sigma$) to denote the sets of atoms and e-literals over signature $\Sigma$ respectively.

A *P-log rule*, $r$ over signature $\Sigma$ is of the form:

$$head(r) \leftarrow body(r) \tag{15}$$

where $head(r)$ is an atom of $\Sigma$ and $body(r)$ is a collection of e-literals of $\Sigma$. As discussed in the introduction, the syntax defined here does not allow negative literals in heads. Moreover, $head(r)$ cannot be of the form $ST = false$ where $ST$ is a special term, and cannot be an arithmetic atom. The head of the rule can be omitted. Such rules are referred to as *constraints*. If $head(r)$ is an observation or an action, we require $body(r)$ to be empty and the rule is called *an activity record*. If $head(r)$ is of the form $random(rn, a, p)$, the rule is called *a random selection rule* with name $rn$, $a$ is referred to as *random attribute term* of the program containing such a rule. A rule which is not an activity record or a random selection rule is called *a regular rule*.

*For readability, we will sometimes omit the first parameter of random, pr, do and truly_random, if the program $\Pi$ from the context satisfies at least one of the following conditions:*

[7] Note that this can sometimes lead to ambiguities: $t$ can be viewed as attribute, attribute term, and an atom. However, such ambiguities can always be resolved from the context.

*1. $\Pi$ contains only one selection rule for attribute term $a$,*
*2. $\Pi$ contains no do statements for $a$.*

Similar to regular ASP, P-log allows variables (denoted by identifiers starting with an upper case letter) in a rule wherever a constant may be used. So one can write, say, a rule $r$ of the form $f(X) \leftarrow q(X)$ where for some sorts $s_1$ and $s_2$, $f : s_1 \rightarrow$ **boolean** and $g : s_2 \rightarrow$ **boolean**. As usual, this is understood as a shorthand for the set of all *ground instances* of this rule, i.e. rules obtained from $r$ by replacing its variables by properly sorted object constants of the language and evaluating the arithmetic terms. In our case $X$ will be replaced by object constants from $s_1 \cap s_2$. (Note that if $s_1$ and $s_2$ are disjoint the set of such instances is empty.) The collection of all ground instances of $r$ is referred to as $r$'s *grounding*.

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

$$pr(r, f(\bar{x}) = y \mid B) = v \tag{16}$$

where $f(\bar{x})$ is a regular term such that $y \in range(f(\bar{x}))$, $B$ is a set of e-literals of $\Sigma$ and $v \in [0, 1]$ is a rational number. The statement says that "if the value of $f(\bar{x})$ is generated randomly by experiment $r$ 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 to* $f(\bar{x}) = y$ *by the pr-atom.*

Unless otherwise stated, we will assume that a P-log program contains the following rules, called *general P-log axioms.*

**General P-log axioms**:

– For every regular term $f(\bar{x})$ of $\Sigma$ the rules

$$\leftarrow not\ f(\bar{x}) = Y,\ obs(f(\bar{x}), Y, true) \tag{17}$$

$$\leftarrow not\ f(\bar{x}) \neq Y,\ obs(f(\bar{x}), Y, false). \tag{18}$$

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.

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

$$f(\bar{x}) = y_1\ \textbf{or}\ \ \ldots\ \textbf{or}\ \ f(\bar{x}) = y_k\ \leftarrow random(r, f(\bar{x}), p)^{8} \tag{19}$$

$$\begin{aligned} truly\_random(r, f(\bar{x})) \leftarrow\ & random(r, f(\bar{x}), p), \\ & not\ do(r, f(\bar{x}), y_1), \ldots, not\ do(r, f(\bar{x}), y_k) \end{aligned} \tag{20}$$

$$\leftarrow f(\bar{x}) = Y,\ not\ p(Y), random(r, f(\bar{x}), p) \tag{21}$$

[8] Disjunction here is a so called shifted disjunction [14] and hence, can be eliminated.

$$\begin{aligned}\leftarrow\ & random(r, f(\bar{x}), p),\\ & not\ f(\bar{X}) = Y,\\ & do(r, f(\bar{X}), Y).\end{aligned} \tag{22}$$

Intuitively, the rules (19) and (21) guarantee that if $random(r, f(\bar{x}), p)$ is true, then $f(\bar{x})$ is assigned the value satisfying condition $p$ by experiment $r$, rule (20) makes sure that $truly\_random(r, f(\bar{x}))$ is true iff the value of $f(\bar{x})$ is assigned as the result of a truly random experiment $r$, i.e., an experiment without any intervention, and rule (22) guarantees that the atoms made true by interventions are indeed true if $random(r, f(\bar{x}), p)$ is true.

In addition, for every rule $r$ which is not a general axiom, we disallow literals formed by *obs*, *do*, *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 free occurrence of any variable and no names of arithmetic functions. The *grounding of a program* with variables is the union of the grounding of every rule of the program. A ground program is *valid* if no two random selection rules have the same name. A program with variables is *valid* if its grounding is valid. *From now on, we only consider valid programs.*

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

a) Partial attributes are explicitly allowed in the language.
b) 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].

## 3.2 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 $\Pi$ is given by the collection $\mathcal{W}(\Pi)$ of possible worlds of $\Pi$ and the probability function $P_{\Pi}$[9], 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.

**Possible worlds**
The definition of $\mathcal{W}(\Pi)$ is very similar to the definition of answer sets of logic programs. An *interpretation* over signature $\Sigma$ is a (possibly partial) mapping $I$, defined on variable-free terms of $\Sigma$, which

– maps attribute terms of $\Sigma$ into values from their corresponding ranges,
– maps arithmetic terms into their standard intended values,
– maps every attribute term $instance_s(x)$, where $s$ is a sort name, to *true* if $x$ is an element of $s$ and to *false* otherwise.

[9] When $\Pi$ is clear from the context we may simply write $P$ instead of $P_{\Pi}$.

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 relation, we will use the notation for elements of $\Sigma$: $a$ 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 $I \not\models B$ or $I \models l$, and
6. $I \models \leftarrow B$ if $I \not\models B$.

We say that an atom $A$ of $\Sigma$ 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 $\Pi^I$ denote the standard *reduct* [17] of a program $\Pi$ with respect to interpretation $I$ – a program obtained from $\Pi$ by:

a) removing all rules containing $not\ l$ such that $I \models l$;
b) removing all other rules containing $not$.

Then possible worlds are defined as follows:

**Definition 1 (Possible World)**
Interpretation $I$ is a *possible world* of $\Pi$ if

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

The following examples illustrate the definition. Recall that, in addition to the explicitly stated rules, every program below contains general P-log axioms (17)–(22). To simplify the specification of programs' signatures, we use standard mathematical declarations of functions, $f : s_1, \dots, s_n \to s$ and sorts $s = \{t_1, \dots, t_m\}$. If $n = 0$, we simply write $f : s$. We follow the input language of our implementation and start the sort names with #.

*Example 3* 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 4* 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 (17).

If we add the rule

$$random(a)$$

to $P_2$, the new program 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

$$\neg a$$

the program would become inconsistent again, because any interpretation that satisfies both rules $\neg a$ and $obs(a, true)$ will violate reality check axiom (17).

## Probability Function

With one small exception which we will mention shortly, our definition of probability function $P_\Pi$ of program $\Pi$ is the same as that in [8], so we do not present it here. Instead, we will illustrate the construction of $P_\Pi$ by a simple example. The complete definition can be found in the appendix of this paper.

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 term $a$ s.t. $truly\_random(a) \in W$ 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 term 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 measure of $W$ divided by the sum of the unnormalized measures of all possible worlds of $\Pi$. Suppose now that $\Pi$ 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 $\Pi$ is the sum of the measures of the possible worlds from $E$. The *probability* with respect to program $\Pi$ of a literal $l$ of $\Pi$, $P_\Pi(l)$, is the sum of the measures of the possible worlds of $\Pi$ that satisfy $l$. For example, $P_{P_3}(a = 1)$ is $1/2$ because $W_1$ and $W_3 = \{a = 1, b = false\}$ are the only possible worlds of $P_3$ that contain $a = 1$ and $\mu(W_1) + \mu(W_3) = 1/2$.

As discussed in [8], the corresponding probability functions are only defined for programs which satisfy three conditions on possible worlds of a program $\Pi$. 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)** For any possible world $W$ of $\Pi$ and any random selection rule:

$$random(rn_1, a, p) \leftarrow body$$

such that $W$ satisfies *body*, there is no other rule whose head is of the form $a = y$ or $random(rn_2, a, q)$ and whose body is satisfied by $W$.

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 other two conditions remain unchanged (except the addition of rule names to *random* and *pr*). We restate them here for completeness. The second condition says:

**Condition 2 (Unique probability assignment)**
If $\Pi$ contains a random selection rule

$$random(rn, a, p) \leftarrow B \tag{23}$$

along with two different probability atoms

$$pr(rn, a \ = y \mid B_1) = v_1 \text{ and } pr(rn, a = y \mid B_2) = v_2 \tag{24}$$

then no possible world of $\Pi$ 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 $\Pi$, then we would have two different assignments to probability of $a = y$ which would be either contradictory (if $v_1 \neq v_2$) or repeat the same information (if $v_1 = v_2$). Both situations are undesirable.

Finally, the last condition is:

**Condition 3 (No probabilities assigned outside of dynamic range)**
If $\Pi$ contains a random selection rule

$$random(r, a, p) \leftarrow B_1 \tag{25}$$

along with probability atom

$$pr(r, a = y \mid B_2) = v \tag{26}$$

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

The condition guarantees that *pr*-atoms of the program do not assign probabilities to impossible values of attributes.

## 4 Background: Coherency and Causally Ordered Programs

In this section we introduce class $\mathcal{B}$ of P-log programs and show that every program from $\mathcal{B}$ is coherent, i.e., defines a collection of possible worlds corresponding to possible sets of beliefs of a rational agent associated with it, and a 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 3 may be incoherent. This can be caused by a well known logical inconsistency of the underlying ASP program, by allowing non-determinism of attribute terms not declared as random, and by some other factors.

Before defining coherency formally, we need some notation. Recall that the syntax of the language only allows observations of the form $obs(l)$, where $l$ is a literal. For a set of extended literals $B$, we introduce shorthand $obs(B)$ denoting the set of two rules, $l' \leftarrow B$ and $obs(l')$, where $l'$ is a fresh atom. The notion of coherency, introduced in [8], is given by the following definition.

**Definition 2 (Program Coherency)**
A P-log program $\Pi$ is called *coherent* if

1. $\Pi$ is *logically consistent*, i.e., has at least one possible world, and
2. $\Pi'$, which is obtained from $\Pi$ by removing all activity records, is *probabilistically consistent*, i.e.,
   (a) probability function $P_{\Pi'}$ is defined, and
   (b) for every selection rule

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

   and every probability atom

   $$pr(r, f(\bar{x}) = y \mid B) = v$$

   of $\Pi$, 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 $\Pi$, agree with corresponding conditional probabilities defined by its a priori part $\Pi'$.

*Example 5 (Incoherent Programs)*
Clearly, a program containing facts $p$ and $\neg p$ is incoherent logically, and has no possible worlds. Probabilistic incoherency is illustrated by program $P_4$:

```
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 unnormalized probabilistic measures 0.3, 0.3, and 0.7, and hence the probability $P_{P_4}(a) = 6/13$ which is different from the causal probability 0.3 given by the pr-atom. Intuitively, the program is incoherent because a non-deterministic attribute term $b$ 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 2 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 $\Pi$ allows a total ordering, $a_1, \ldots, a_k$ of attribute terms of $\Pi$ based on a simple dependency relation, defined as follows: (a) Attribute term $a_2$ "immediately depends" on $a_1$ if $a_2$ occurs in the head of a rule or pr-atom whose body contains an occurrence of $a_1$. (Note, that $B$ is the body of a pr-atom $pr(r, f = y \mid B)$ 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 attribute terms, i.e., for any two distinct random attribute terms, $a_1$ and $a_2$, if $(a_1, a_2)$ belongs to the ordering, then $(a_2, a_1)$ doesn't.

If such an ordering is given, then the possible worlds of $\Pi$ can be constructed gradually from possible worlds of programs $\Pi_0, \Pi_1, \ldots, \Pi_n$ such that $\Pi_n = \Pi$, $\Pi_i \subseteq \Pi_{i+1}$. Each $\Pi_i$ 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 term of $\Pi$, $L_1$ 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 $\Pi$ as the *signature* of $\Pi$.) To ensure that non-determinism is only possible for random attribute terms, the non-random base, $\Pi_0$, of $\Pi$ is required to have exactly one possible world.

The construction of possible worlds of $\Pi$ starts with the possible world $W_0$ of $\Pi_0$ and proceeds recursively by considering a possible world $W$ of $\Pi_i$ with a random selection rule for $a_{i+1}$ whose body is satisfied in $W$, and the following steps.

1. If there is a random selection rule for $a_{i+1}$ whose body is satisfied by $W$, building new possible worlds of $\Pi_{i+1}$ corresponding to possible values of $a_{i+1}$. In this case, for a program to be causally ordered, the possible outcomes of that selection shall not be constrained by logical rules or other random selections.

2. If the bodies of random selection rule for $a_{i+1}$ are all not satisfied by W, building a possible world of $\Pi_{i+1}$ where $a_{i+1}$ is undefined, or has an assignment resulting from non-random selection rules. In this case, for a program to be causally ordered, such a possible world must be unique.

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 6 (Causally Ordered Programs)*
Consider program, $P_5$:

```
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 attribute terms of $P_5$ 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 $\Pi_1$ which is equal to $P_5$, and random selection rule,

$$random(a) \leftarrow not\ b, f$$

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 $P_5$ were expanded by

$$e \leftarrow a, not\ e$$

then outcome $a = true$ would become impossible, and the program would not be causally ordered. It is also easy to see that program $P_4$ 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 $P_5$, 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 the addition of another $pr$-atom,

$$pr(\neg a) = 0.2$$

would make the program non-unitary.

Since $P_5$ 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 in Section 6).

*Example 7 (Coherent program which is not causally ordered)*
Consider program: $P_6$

```
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 $P_5$ by adding the rule

$$random(b) \leftarrow a, not\ f \tag{$*$}$$

Since, according to the original definition of dependency, $a$ and $b$ depend on each other, there is no ordering of random attribute terms which would allow to view $P_6$ as the result of gradual construction of programs $\Pi_0$, $\Pi_1$, ..., corresponding to this ordering. Hence, $P_6$ is not causally ordered. However, it is easy to check that $P_6$ is coherent. It has possible worlds $W_0 = \{f, a, c\}$[10] and $W_1 = \{f, \neg a, d\}$ and probability $P_{P_6}(a) = 0.3$ which coincides with that of $P_6$. This, of course, can be immediately seen from the fact that, since $f \in P_6$, 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 $P_6$. 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 $P_6$, the uselessness of rule $(*)$ becomes apparent and the rule can be removed.

## 5 Dynamically Causally Ordered Programs and the Main Theorem

We start this section from defining the class $\mathcal{B}$ of dynamically causally ordered unitary programs which includes program $P_6$ from Example 7. The main idea is to replace causal ordering of attribute terms of the program by an ordering based on the dependency relation between attribute terms 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 3 only considers $B$ consisting of e-literals formed by attribute terms from $L$ (symbolically, $B \subseteq$ e-lits$(L)$). In what follows we allow $B$ containing e-literals over some other signature.

**Definition 3 (Extended Satisfiability Relation)**

1. Interpretation $I$ of signature $L$ *satisfies* a set $B$ of e-literals (over an arbitrary signature) if $B \subseteq$ e-lits$(L)$ 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-lits$(L)$ and is not satisfied by $I$.

[10] To shorten the description, we will omit special terms and literals formed by them from program signatures, interpretations, possible worlds, etc. in the remainder of this paper.

As an example, consider a set $B = \{a, not\ b\}$ and interpretation $I_0 = \{a\}$ of $L_0 = \{a\}$. Since $(not\ b) \notin$ e-lits$(L_0)$, $I_0$ 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 will also need some notation. By $L_{base}(\Pi)$ we denote the set of attribute terms of program $\Pi$ which do not depend on any of its random attribute terms; $R_{base}(\Pi)$ denotes the collection of rules of $\Pi$ 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 $\Pi$.

To eliminate useless rule of $\Pi$, we introduce the following notion:

**Definition 4 (Reduct)**

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

– $red(\Pi)$ is defined iff the base of $\Pi$ has exactly one possible world (denoted by $W_{base}$), and
– $red(\Pi)$ is the program obtained from $\Pi$ by removing all pr-atoms and rules whose bodies are falsified by $W_{base}$.

*Example 8 (Reduct)*
Consider program $P_6$ from Example 7. It is easy to see that $L_{base}(P_6) = \{f\}$ and $R_{base}(P_6) = \{f.\}$. The base has exactly one possible world $W_{base} = \{f\}$, and thus, $red(P_6)$ is defined.

Clearly, $red$ eliminates the useless third rule of $P_6$. The body $\{a, not\ f\}$ is falsified by $W_{base}$ (which is, of course, an interpretation of $L_{base}$). Hence, $red(P_6)$ includes every rule of $P_6$ except the third one. (One may note that, as expected, $red(P_6) = P_5$.)

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 $red$[11], is defined in several steps. We start with a strict ordering $\alpha$ of random attribute terms of $\Pi$ (often referred to as *probabilistic leveling*), and use the dependency relation to extend it to (not necessarily strict) ordering of all

[11] 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.

attributes of $\Pi$ (referred to as *total leveling*). (Note that the original probabilistic ordering of the program does not necessarily reflect the dependency relation between random attribute terms.) In a manner similar to that in the definition of causally ordered programs, this total leveling defines layers $\Pi_0, \dots, \Pi_n$ of $\Pi$, referred to as *dynamic structure of $\Pi$ induced by $\alpha$*. If $\alpha$ is such that possible worlds of $\Pi$ can be gradually constructed from the layers of the dynamic structure induced by $\alpha$, then $\Pi$ is dynamically causally ordered via $\alpha$. This intuition is captured by the following definitions.

**Definition 5 (Total Leveling)**

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

1. If $a$ is of the form $random(rn, b, p)$, then $|random(rn, b, p)| = |b|$.
2. Otherwise:
   (a) $|a| = 0$ iff $a$ does not depend on any random attribute term of $\Pi$ in $red(\Pi)$.
   (b) $|a| = i$ iff $i$ is the level of the random attribute term $b$ of $\Pi$ such that
      i. $a$ depends on $b$ in $red(\Pi)$ and
      ii. there is no random attribute term $c$ 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 $\alpha$.
Total levelings are extended to the e-literals of $\Sigma$ as follows:

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

*Example 9 (Total Leveling)*
Consider a probabilistic leveling $\alpha$ of the random attribute terms of $P_6$ such that $\alpha(b) = 1, \alpha(a) = 2$. In Example 8 we showed that $red(P_6)$ is defined. $\alpha$ can be expanded to the total leveling $|\ |$ as follows. Since $f$ does not depend on $a$ or $b$, its level is 0, i.e., $|f| = 0$. $random(a)$ has the same level as $a$, i.e., $|random(a)| = 2$. Similarly, $|random(b)| = 1$. Since $c, d$ depend on both $b$ and $a$ in $red(P_6)$, 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 $\Pi$ defined by an ordering of random attribute terms.

**Definition 6 (Dynamic Structure)**

Let $\Pi$ be a program such that $red(\Pi)$ is defined and $|\ |$ be the total ordering determined by a probabilistic leveling $a_1, \dots, a_k$ of the random attribute terms of $\Pi$.

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

1. $L_i$ consists of all attribute terms of $\Pi$ whose levels are $i$ or less in $|\ |$,
2. for every rule or pr-atom $r$ of $\Pi$, $\Pi_i$ 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 $\Pi_0, \dots, \Pi_k$.

*Example 10 (Dynamic Structure)*
Consider program $P_6$ and probabilistic leveling $\alpha(b) = 1$ and $\alpha(a) = 2$ from Example 9. We have already shown that $red(P_6)$ 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 $P_6$ induced by $\alpha$

| | |
|---|---|
| $L_0 = \{f\}$ | $\Pi_0$ is |
| | `f.` |
| $L_1 = \{f, b, random(b)\}$ | $\Pi_1$ is |
| | `f.` |
| $L_2 = \{f, b, random(b)$ $a, random(a), c, d\}$ | $\Pi_2$ is |
| | `f.`<br>`random(a) :- not b, f.`<br>`random(b) :- a, not f.`<br>`pr(a) = 0.3.`<br>`c :- a.`<br>`d :- -a.` |

Now we are ready to give the definition of a dynamically causally ordered (DCO) program. The definition will consist of three parts. First, we consider programs not containing activity records, and define what it means for such programs to be (a) DCO via a given probabilistic leveling, (b) DCO. Finally, we will define arbitrary DCO programs.

**Definition 7 (DCO via Probabilistic Leveling)**

Let $\Pi$ be a program not containing activity records.

$\Pi$ is *dynamically causally ordered (DCO) via a probabilistic leveling* $a_1, \dots, a_k$ of the random attribute terms of $\Pi$ if

1. $red(\Pi)$ is defined,
2. the dynamic structure $\langle L_0, \Pi_0 \rangle, \dots, \langle L_k, \Pi_k \rangle$ induced by $a_1, \dots, a_k$ 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 $\Pi$ with $a_i$ in the head, and $r$ is not a ground instance of a general axiom of $\Pi$ then the body of $r$ is either falsified or satisfied by $W_{i-1}$,
      ii. if $r$ is a random selection rule, $random(rn, a_i, p) \leftarrow B$, and $W_{i-1}$ satisfies $B$ then
         – there is $y \in range(a_i)$ such that $p(y) \in W_{i-1}$,
         – for every $y \in range(a_i)$, $p(y)$ belongs to 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 the bodies of all random selection rules with $a_i$ in the head then $W_{i-1} \cup \Pi_i$ has exactly one possible world.

If a program $\Pi$ is DCO via a probabilistic leveling $\alpha$, then $\alpha$ is called a *dynamic causal probabilistic leveling of* $\Pi$.

*Example 11 (DCO Program via a Probabilistic Leveling)*
Consider program $P_6$ and its dynamic structure induced by probabilistic leveling $\alpha(b) = 1, \alpha(a) = 2$ from Example 10. We will verify that $P_6$ is dynamically causally ordered via $\alpha$.

The first condition, the existence of $red(P_6)$, was established earlier. Consider the dynamic structure induced by $\alpha$ from Table 1.

The first layer $\langle L_0, \Pi_0 \rangle$, 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 term $a_1 = b$. The only rule with random attribute term $a_1 = b$ in the head mentioned in condition (i) of (2b) is the third rule of $P_6$. 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 term $a$ in the head. The only one such rule is the second rule of $P_6$. 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 $random(a)$ stands for $random(a, instance_{boolean})$. By definition of interpretation, every interpretation contains $instance_s(y)$ for each sort $s$ and $y \in s$. Hence, $instance_{boolean}(true) \in W_{i-1}$ 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, $P_6$ is dynamically causally ordered via the given probabilistic leveling. Hence, $\alpha$ is a dynamic causal probabilistic leveling of $P_6$.

**Definition 8 (Dynamically Causally Ordered Programs - I)**

Let $\Pi$ be a program not containing activity records. $\Pi$ is *dynamically causally ordered* if $\Pi$ is dynamically causally ordered via some probabilistic leveling of the random attribute terms of $\Pi$.

In the example 11, $P_6$ is dynamically causally ordered because it is so via the probabilistic leveling $\alpha(b) = 1, \alpha(a) = 2$.

**Definition 9 (Dynamically Causally Ordered Programs - II)**

Let $\Pi$ be an arbitrary program, and $\Pi'$ be the program obtained from $\Pi$ by removing all activity records. $\Pi$ is *dynamically causally ordered* if $\Pi'$ is dynamically causally ordered. A *dynamic causal probabilistic leveling of* $\Pi$ is a dynamic causal probabilistic leveling of $\Pi'$.

*The class of logically consistent dynamically causally ordered unitary programs is referred to as class $\mathcal{B}$.*

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

The proof of Theorem 1 is in Appendix B of the full version of this paper[12]. An important intermediate result of the proof is a formulation of splitting set theorem for P-log, originally introduced in [29] for logic programs, is given in Subsection B.2 of the proof.

We believe the theorem is useful for at least three reasons. Firstly, as we will show in Section 6, it can be used to establish coherency, and, therefore, consistency and correctness, of programs that do not belong to class $\mathcal{A}$ of causally ordered unitary programs, without computing all the possible worlds and their probabilistic measures. Secondly, the theorem was used in the dissertation of the first author to prove the correctness of a new efficient inference algorithm. We plan to publish these results in future. Finally, the authors of [8] suggest a methodology to construct coherent programs: (a) non-determinism should be resulted solely from random selection rules, and (b) the random choices not to be constrained by other rules. We believe the methodology is better captured by class $\mathcal{B}$ than by causally ordered programs. In Section 6, we will show examples of programs that follow the methodology, belong to class $\mathcal{B}$, but are not causally ordered. Also, as shown in the same section, there are causal ordered programs that do not belong to $\mathcal{B}$. However, all such programs we know are artificial. All meaningful causally ordered programs we are aware of belong to $\mathcal{B}$.

## 6 $\mathcal{B}$ versus $\mathcal{A}$: Examples

In this section, we present examples that show the difference between class $\mathcal{B}$ and class $\mathcal{A}$. The first three examples are rather natural P-log programs from class $\mathcal{B}$ which do not belong to $\mathcal{A}$. The last example is a program from $\mathcal{A}$ that does not belong to $\mathcal{B}$.

### 6.1 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:

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


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

[12] https://github.com/iensen/plog2.0/raw/master/papers/plog_ref_dco_full.pdf

```
% 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

$$random(throw(2)) \leftarrow throw(3) \neq 1, 2 = 3$$

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 $\alpha(throw(1)) = 1, \ldots, \alpha(throw(5)) = 5$. 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 7 are satisfied. The program is obviously unitary because it doesn't contain pr-atoms. Hence, the program belongs to class $\mathcal{B}$.

## 6.2 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:

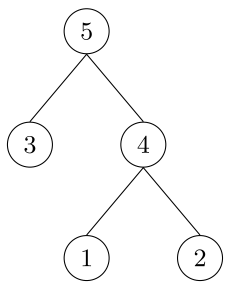
```
%% Random Tree Problem
% Sorts
#node = {1,2,3,4,5}.
#value = {1,2,3,4,5,6}.

% 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

$$random(value\_of(1) : \{X : possible\_value(X, 1)\}) \leftarrow \neg leaf(1)$$
$$possible\_value(3, 1) \leftarrow arc(4, 1), value\_of(4) = 3$$

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 a 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 attribute terms. Hence $value\_of(1)$, $value\_of(2)$ and $value\_of(3)$ can be ordered arbitrarily. Consider, for instance, an ordering $\alpha$ where $\alpha(value\_of(1)) = 1$, $\alpha(value\_of(2)) = 2$ and $\alpha(value\_of(3)) = 3$. Since $value\_of(4)$ "depends" only on $value\_of(1)$, and $value\_of(2)$, its level should be greater than $\alpha(value\_of(1))$ and $\alpha(value\_of(2))$. Since $\alpha(value\_of(3)) = 3$, we can have $\alpha(value\_of(4)) = 4$. Since $value\_of(5)$ "depends" on $value\_of(3)$ and $value\_of(4)$, we have $\alpha(value\_of(5)) = 5$.

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

$$\begin{aligned}
&arc(4, 5)\\
&arc(3, 5)\\
&arc(2, 4)\\
&arc(1, 4)\\
leaf(X) = false &\leftarrow arc(Y, X)\\
leaf(X) = true &\leftarrow not\ leaf(X) = false
\end{aligned}$$

which has the unique possible world

$$\begin{aligned}
W_{base} = \{&arc(4, 5), arc(3, 5), arc(2, 4), arc(1, 4),\\
&leaf(1), leaf(2), leaf(3), \neg leaf(4), \neg leaf(5)\}
\end{aligned}$$

Hence, $red(P^t)$ is defined.

Let $\langle L_0, \Pi_0\rangle, \dots, \langle L_5, \Pi_5\rangle$ be the dynamic structure induced by $\alpha$. 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 7 we start with level $i = 1$ and random attribute term $value\_of(1)$. The rules with $value\_of(1)$ in their heads are:

$$\begin{aligned}
random(value\_of(1)) &\leftarrow leaf(1)\\
random(value\_of(1) : \{X : possible\_value(X, 1)\}) &\leftarrow \neg leaf(1)\\
possible\_value(X, 1) &\leftarrow arc(N1, 1), value\_of(N1) = X
\end{aligned}$$

The first rule is satisfied by $W_0$, the second one is falsified by $W_0$, and every instance of the third rule for $X \in 1..6, N1 \in 1..5$ is falsified by $W_0$, because $arc(N1, 1)$ is not satisfied by $W_0$. Hence, condition (i) of Definition 7 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 7 holds. Condition (iii) of Definition 7 is true vacuously.

Similarly, we can show that for any level $i \in 2..5$, conditions (i) to (iii) of Definition 7 hold. Hence $P^t$ is dynamically causally ordered via probabilistic leveling $\alpha$. The program is obviously unitary because it doesn't contain pr-atoms. Hence, it belongs to class $\mathcal{B}$.

## 6.3 Blood Type Problem

This is a typical blood type problem frequently used in probability and statistics classes. This particular problem description is based on Section 4.1.3 from [49].

*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

| Father's gene / Mother'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), and 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 where Mary and Todd are the parents of 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 of the program are $genotype\_of(mary)$, $genotype\_of(todd)$ and $genotype\_of(john)$. $P^b$ contains the following rules

$$\begin{aligned} random(genotype\_of(mary) :&\{G : possible\_genotype(mary, G)\}) \leftarrow \\ & generation\_of(mary) = 2 \end{aligned}$$

and

$$\begin{aligned} & possible\_genotype(mary, g(g\_a, g\_b)) \leftarrow \\ &\qquad father(mary) = john, \\ &\qquad mother(mary) = todd, \\ &\qquad genotype\_of(john) = g(g\_a, g\_b), \\ &\qquad genotype\_of(todd) = g(g\_a, g\_b), \\ &\qquad possible\_combination(g(g\_a, g\_b), g(g\_a, g\_b), g(g\_a, g\_b)) \end{aligned}$$

Clearly, rules obtained from the two rules above by replacing $mary$ by $john$ and $john$ by $mary$ also belong to $P^b$. Hence, attribute terms $genotype\_of(mary)$ 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 $\alpha$:

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

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 $\alpha$. $\Pi_0$ consists of

$$\begin{gathered} generation(mary) = 1 \\ generation(todd) = 1 \\ generation(john) = 2 \\ mother\_of(john) = mary \\ father\_of(john) = todd \end{gathered}$$

$\Pi_0$ 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

$$\begin{gathered} random(genotype\_of(mary)) \leftarrow generation\_of(mary) = 1 \\ random(genotype\_of(mary) : \{G : possible\_genotype(mary, G)\}) \leftarrow \\ \qquad\qquad\qquad generation\_of(mary) = 2 \end{gathered}$$

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 7 are satisfied. Similarly, we can verify these conditions for levels 2 and 3. Hence, the program $P^b$ is dynamically causally ordered. The program is clearly unitary because the sum of pr-atoms over all possible values of $genotype\_of(X)$ is equal to 1. Hence, $P^b$ belongs to $\mathcal{B}$.

## 6.4 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 $\Pi_0$ 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

$$random(y : \{X : p(X)\}) \leftarrow x$$

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

$$a \leftarrow not\ b, x$$

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 selection 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. Since $P$ has no pr-atoms, it is unitary, and, therefore, belongs to $\mathcal{A}$.

Now we show that the program is not dynamically causally ordered. There are only two different probabilistic levelings:
$\alpha_1(y) = 1$ and $\alpha_1(x) = 2$,
$\alpha_2(y) = 2$ and $\alpha_2(x) = 1$.
In both cases $red(P)$ is defined, $\Pi_0$ is the same and has unique possible world $W_0$, and, therefore, conditions (1) and (2a) of Definition 7 are satisfied.

The program will not be dynamically causally ordered via $\alpha_1$. 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 7.

Now consider probabilistic leveling $\alpha_2$. Note that the total leveling $|\ |$, used for showing that the program is causally ordered, is not the total leveling determined by $\alpha_2$. 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 $h \in W_0$, the rule

$$a \leftarrow not\ h, y$$

is not in $red(P)$. Since $a$ and $b$ do not depend on $y$ in $red(P)$, their level cannot be 2 as in $|\ |$. 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 $\alpha_2$. 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 $\alpha_2$. We will show $P_1 = W_0 \cup \Pi_1^2 \cup \{\leftarrow not\ x = true\}$ does not have a unique possible world. $\Pi_1^2$ will contain

$$a \leftarrow not\ b, x$$
$$b \leftarrow not\ a, x$$

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 7. Hence, the program is not dynamically causally ordered via the probabilistic leveling $\alpha_2$.

In summary, the program is not dynamically causally ordered via any probabilistic leveling. Thus, it is not dynamically causally ordered and, therefore, it does not belong to $\mathcal{B}$.

## 7 Coherency for P-log with CR-Rules

The story of coherency told in this paper so far is limited to P-log programs without cr-rules. This is not surprising since the language and Definition 2 of coherency introduced in [8] does not allow cr-rules. After the original language was expanded in [20], no attempts were made to generalize the notion of coherency to P-log with such rules. In this section we give such a generalization, define a new class $B^+$ expanding $B$ by programs with cr-rules, and show that programs of $B^+$ are coherent according to the new definition. The class is rather broad. It contains all examples of programs of P-log with cr-rules written to formalize commonsense scenarios which we found in the literature.

We start with a short review. Recall that, syntactically, a cr-rule is an expression of the form

$$head \overset{+}{\leftarrow} body.$$

Intuitively, the rule says that if the reasoner associated with the program believes the body of the rule , then it may also believe its head; however, this possibility may be used only if there is no way to obtain consistent set of beliefs by using only regular rules of the program. A *P-log program with cr-rules* is a pair consisting of a sorted signature and a collection of P-log rules, pr-atoms and cr-rules over this signature. Let $\Pi$ be a P-log program with cr-rules. For a subset $R$ of cr-rules of $\Pi$, by $\alpha(R)$ we will denote a collection of rules obtained from rules of $R$ by replacing $\overset{+}{\leftarrow}$ with $\leftarrow$. By $\Pi_r$ (where $r$ stands for "regular") we will denote a program obtained from $\Pi$ by removing cr-rules. The definition of the semantics of cr-rules is based on the notion of abductive support. The notion is usually parameterized by preference relation on sets of cr-rules but, for simplicity, we compare such sets by their cardinality.

**Definition 10 (Abductive Support)**
A cardinality-minimal collection $R$ of cr-rules of $\Pi$ such that $\Pi_r \cup \alpha(R)$ is logically consistent (i.e. has a possible world) is called an *abductive support* of $\Pi$.

Since activity records simply activate some of the program's constraints, it can be shown that no abductive support can contain rules whose head is formed by *obs* and *do*. For simplicity, we will assume that such rules do not exist in the program.

**Definition 11 (Possible Worlds of P-log Programs with CR-Rules)** A set $W$ is called a *possible world* of $\Pi$ if it is a possible world of a regular P-log program $\Pi_r \cup \alpha(R)$ for some abductive support $R$ of $\Pi$.

The definition of probabilistic function defined by $\Pi$ remains unchanged.

To see what changes need to be made in the original definitions of coherency, let us recall that, intuitively, a program is coherent if it is logically consistent and defines probability distribution compatible with its probabilistic atoms. The addition of cr-rules does not change formalization of the first condition - we just require existence of a possible world of the program. The second condition though requires some changes. To see that, let us consider a program $T_0$:

```
a,b,c,d: #boolean.
random(a) :- b.
pr(a|b) = 0.5.
c :- a, not d.
d :- a, not c.
b+.
```

The original definition of coherency (Definition 2) can be applied to program with cr-rules. It is easy to check that according to such a definition $T_0$ is coherent because it is logically consistent and probabilistically consistent. The program assigns probability 0 to each of its atoms. If we consider

$$T_1 = T_0 \cup \{obs(b)\}$$

then, by the original definition applied to programs with cr-rules, $T_1$ is also coherent. It has a possible world $\{b, a, c\}$ and thus logically consistent. $T_1' = T_0$ and, hence, is probabilistically consistent. Therefore, $T_1$ is coherent. But this is unintuitive, since program $T_2$

```
random(a) :- b.
pr(a|b) = 0.5.
c :- a, not d.
d :- a, not c.
b.
```

which is logically equivalent to $T_1$ and has the same probability atoms is not probabilistically coherent. This happens, because of probabilistic non-monotonicity of observations. In the presence of cr-rules, addition of observations may create new possible worlds and substantially change probabilistic distribution defined by the program. This is what happened in $T_1$. Even though the distribution defined by $T_1' = T_0$ is vacuously compatible with pr-atom $pr(a \mid b) = 0.5$ which is useless in $T_0$, it is not enough to render $T_1$ probabilistically consistent. For that the pr-atom should be compatible with the new distribution defined by $T_1$, which is, of course, not the case.

The new definition of probabilistic consistency formalizes this observation:

**Definition 12** [Probabilistic consistency for P-log programs with cr-rules]
Let $\Pi$ be a P-log program without activity records.

- $\Pi$ is *probabilistically consistent with respect to a collection of activity records* $O$ if $\Pi \cup O$ has unique abductive support $R$[13] and regular program $\Pi_r \cup \alpha(R)$ is probabilistically consistent.
- $\Pi$ is *probabilistically consistent* if it is probabilistically consistent with respect to every collection of activity records $O$ *compatible* with $\Pi$, i.e., such that $\Pi \cup O$ is logically consistent.

**Definition 13** [Coherency]
P-log program $\Pi$ is called *coherent* if it is logically consistent and program $\Pi'$, obtained from $\Pi$ by dropping activity records, is probabilistically consistent.

It is not difficult to verify that for programs without cr-rules Definitions 2 and 13 are equivalent, i.e., a program $\Pi$ without cr-rules is coherent according to Definition 2 iff it is coherent according to Definition 13.

We illustrate our new definition of coherency by considering programs $T_0$ and $T_1$ described above. The cr-rule $b \overset{+}{\leftarrow}$ is not activated in $T_0$, the program's possible world is empty, $P_{T_0}(b) = 0$; $T_0$ is both logically and probabilistically consistent and, hence, coherent. Program $T_1$, however, is not probabilistically consistent and hence is not coherent. To see that, let us notice that activity record $O = obs(b)$ is compatible with $T_1'$ because $T_1' \cup O$ is logically consistent. $T_1' \cup O$ has a unique abductive support which is $\{b\}$. $(T_1')_r \cup \{b\}$ is program $T_2$ which is clearly not probabilistically consistent. Hence, $T_1'$ is not probabilistically consistent with respect to $O$ and, therefore, $T_1$ is not coherent.

[13] It may be possible to relax the uniqueness requirement and allow some programs with multiple abductive supports. This option, however, needs a more careful study and thus will be left for future work.

We now define a new class of programs, $\mathcal{B}^+$, expanding $\mathcal{B}$ by programs with cr-rules and generalize our coherency result for this class.

**Definition 14** $[\mathcal{B}^+]$
Let $\Pi$ be a program possibly containing cr-rules. Let $\Pi'$ be the program obtained from $\Pi$ by removing activity records. $\Pi$ belongs to $\mathcal{B}^+$ if and only if the following two conditions hold.

1. $\Pi$ is logically consistent.
2. For each set $O$ of activity records compatible with $\Pi'$, $\Pi' \cup O$ has a unique abductive support $R$, and $\Pi'_r \cup \alpha(R)$ belongs to $\mathcal{B}$.

**Theorem 2** *Every program from $\mathcal{B}^+$ is coherent.*

*Proof* Let $\Pi$ be a program that belongs to $\mathcal{B}^+$. By clause 1 of Definition 14, we have that $\Pi$ is logically consistent. Therefore, it is sufficient to show that

$$\Pi' \text{ is probabilistically consistent.} \tag{27}$$

Let $O$ be an arbitrary collection of activity records $O$ compatible with $\Pi'$. By Definition 12, to prove (27), it is sufficient to show that:

$$\Pi' \text{ is probabilistically consistent with respect to } O \tag{28}$$

By clause 2 of Definition 14, we have:

$$\Pi' \cup O \text{ has a unique abductive support } R \text{ and } \Pi'_r \cup \alpha(R) \text{ belongs to } \mathcal{B}. \tag{29}$$

Since cr-rules in the programs we consider do not have *do* and *obs* in the heads, the program $\Pi'_r \cup \alpha(R)$ doesn't have activity records. Therefore, from (29), Theorem 1 and Definition 2 we have:

$$\Pi'_r \cup \alpha(R) \text{ is probabilistically consistent.} \tag{30}$$

From (30) and Definition 14, we have (28). □

Now we will use this result to show that program $\Pi_{obs}$ from Example 1 is coherent. By Theorem 2, we just need to show that

$$\Pi_{obs} \in \mathcal{B}^+ \tag{31}$$

The program has only one cr-rule $ab \overset{+}{\leftarrow}$, in what follows denoted by $r$. As shown in Section 2, $\Pi_{obs}$ is consistent, so it is sufficient to show that both programs $(\Pi'_{obs})_r$ and $(\Pi'_{obs})_r \cup \{ab\}$ belong to $\mathcal{B}$. Clearly, both programs are unitary because the sum of values of pr-atoms for each attribute term does not exceed 1. The first program has a unique possible world containing atoms $\neg slow\_rate(bob)$ and $healthy(bob)$, and the bodies of all probability atoms and random selection rules are falsified by the absence of $ab$. So, it's fairly straightforward to see that conditions of dynamically causally ordered programs are satisfied for an arbitrary leveling. It also can be checked that the second program, $(\Pi'_{obs})_r \cup \{ab\}$, is dynamically causally ordered via the leveling assigning levels 1 and 2 to attributes *active* and *diet* respectively. Therefore, $\Pi_{obs}$ belongs to $\mathcal{B}^+$ and, by Theorem 2, coherent.

## 8 Related Work

Our paper belongs to the collection of works aimed at the development of declarative languages and knowledge representation methodology combining logic programming and probabilistic reasoning. This direction of research goes back to Probabilistic Horn Abduction of Poole [40] and Distribution Semantics of Sato [43]. While the original attempts were limited to definite programs they were quickly extended to languages with non-monotonic negation. Our work continues the investigation of one such attempt: knowledge representation language P-log. The logical foundation of the first version of P-log, introduced in [7,8], was Answer Set Prolog. In [6] this was extended to CR-Prolog which allows an important type of abductive reasoning. As it was described in the paper, our work is also closely connected with that by Weijun Zhu [49].

There is a substantial amount of work on languages combining non-monotonic logical reasoning with reasoning about probability which is different from P-log. The original P-log paper [8] presents the comparison with some of this work, including L-PLP [30], PRISM [44], NS-PLP [34] , SLP [33], PKB [35] , BLP [26], LPAD [46], ICL [41], PHA [41] and others. Since these languages contain neither partial functions nor other features of P-log which are refined in our paper, this comparison is still valid. Recently, several other languages expanding ASP with probabilistic information whose goals overlap with that of P-log, were introduced by various authors. In what follows we discuss the relationship between P-log and one of such languages – PDLP [31] (also known as Credal Logic [12]). A detailed study of the relationship between another relevant formalism $\mathrm{LP}^{\mathrm{MLN}}$[27] and P-log can be found in [28] and [2].

A program of Credal Logic is a pair consisting of an ASP program $\Pi$ and a collection of "probabilistic facts" - logical facts preceded by their probability. In what follows we use program $C$

$$\begin{gathered} 0.3 :: insomnia. \\ work \leftarrow not\ sleep. \\ sleep \leftarrow not\ work, not\ insomnia. \end{gathered}$$

from [12] as the running example. The first line of $C$ is a probabilistic fact which says that *insomnia* holds with probability 0.3. Intuitively, $C$ can be viewed as a generator of two ASP programs:

$$\begin{gathered} insomnia. \\ work \leftarrow not\ sleep. \\ sleep \leftarrow not\ work, not\ insomnia. \end{gathered}$$

which is generated with probability 0.3, and

$$\begin{gathered} work \leftarrow not\ sleep. \\ sleep \leftarrow not\ work, not\ insomnia. \end{gathered}$$

generated with probability 0.7. The first program has one stable model[14],

$$S_1 = \{insomnia, work\}$$

[14] As usual for the representation of stable models, atoms not listed in the models are false.

while the second has two stable models,

$$S_2 = \{sleep\}$$

and

$$S_3 = \{work\}.$$

The semantics of the program is given by its credal set - the collection of probability measures of the form $P(S_1) = 0.3$, $P(S_2) = \gamma * (1 - 0.3)$, and $P(S_3) = (1 - \gamma) * (1 - 0.3)$ for every $\gamma \in [0, 1]$. If we want to reason about a particular distribution, we can fix $\gamma$. Otherwise, we can compute maximum and minimum probability of a given event over all elements of the credal set.

To illustrate the difference between P-log and Credal Logic, we will first write a P-log program defining the probability model $P_\gamma$ of $C$ with a particular, fixed $\gamma$. Next, we discuss how P-log can be used to specify the credal set defined by $C$.

Since P-log is a sorted language, we will start with the declarations. The story formalized in $C$ seems to be about different types of person's activity, so we define a sort

$$\#activity = \{work, sleep\},$$

a non-boolean attribute

$$act : \#activity$$

and boolean attributes

$$insomnia : \#boolean$$

$$possible : \#activity \rightarrow \#boolean.$$

Both, *insomnia* and *act*, are random attributes defined by the following random selection rules:

$$random(insomnia).$$
$$random(act : \{Y : possible(Y)\}).$$

The last rule defines *act* as a function with values randomly selected from its "dynamic domain" - the set $\{Y : possible(Y)\}$.

Definition of *possible* is based on the default *"Normally an agent described by the program can perform every activity"*:

$$possible(X) \leftarrow not\ \neg possible(X).$$

But there could be exceptions, e.g:

$$\neg possible(sleep) \leftarrow insomnia.$$

The program has the following possible worlds:

$$W_1 = \{insomnia, possible(work), \neg possible(sleep), act = work\}$$

$$W_2 = \{\neg insomnia, possible(work), possible(sleep), act = work\}$$

$$W_3 = \{\neg insomnia, possible(work), possible(sleep), act = sleep\}.$$

Modulo the new term *possible*, the program is logically equivalent to $C$. To ensure that our program defines probability distribution $P_\gamma$, we simply add

$$pr(insomnia) = 0.3.$$
$$pr(act = sleep \mid \neg insomnia) = \gamma.$$

It is easy to check that the resulting program, $\Pi_\gamma$, defines $P_\gamma$, belongs to class $\mathcal{B}$ and is, therefore, coherent.

The program $\Pi_\gamma$ illustrates several features of P-log not available in Credal Logic. This includes the use of *random*, which allows to immediately see the source of randomness, which is somewhat hidden in $C$, the use of non-boolean functions, the ability to specify conditional probabilities and random attributes with dynamic domains, and the sort system which makes the program more readable and helps to capture a number of type related errors. Some other features of P-log like the *do* operator allowing intervention, and availability of cr-rules greatly facilitating abduction and diagnostic reasoning are also not available in Credal Logic.

The latter, however, has an interesting feature not available in P-log - the ability to specify credal set, i.e. the whole collection of distributions compatible with the program. If the collection has more than one element, it, of course, cannot be specified by a P-log program. To better understand the reason let us consider the program $\Pi_C$ obtained from $\Pi_\gamma$ by dropping information not present in $C$, namely, the pr-atom $pr(act = sleep \mid \neg insomnia) = \gamma$. Even though $\Pi_C$ and $C$ are equivalent logically, the probabilistic semantics of the two programs are different. Instead of defining the credal set of $C$, the P-log program $\Pi_C$ defines one probabilistic model determined by the probability measure, which is 0.3 on $S_1$ and $0.5 \cdot (1 - 0.3)$ on $S_2$ and on $S_3$. This is, of course, a consequence of the indifference principle embedded in the semantics of P-log. While the principle is very convenient for a programmer, it precludes a P-log program from specifying two or more probability distributions.

To define credal sets similar to those of Credal Logic, we should probably go from a P-log program to a *P-log module*.

For the purpose of our discussion, by a P-log module, $M$, we mean a P-log program $\Pi$ together with a collection of pr-atoms $IN$ referred to as possible inputs of $\Pi$. A subset $I$ of $IN$ is called a *valid input* of $\Pi$ if $\Pi \cup I$ is a coherent program. A *probability distribution defined by module* $M$ is the probability distribution defined by P-log program $\Pi \cup I$ for some valid input $I$[15].

To illustrate the definition let us go back to our example.

Consider a module $M$ consisting of P-log program $\Pi_C$ and the set $IN$ of pr-atoms of the form

$$pr(act = sleep \mid \neg insomnia) = \gamma$$

where $\gamma \in [0, 1]$. It is easy to see that module $M$ defines the same credal set as $C$. It is interesting to check if the notion of P-log module can be used to capture credal sets of arbitrary programs of Credal Logic but this is a subject for future work.

[15] This definition is similar to that of ASP module (see, for instance, [36]).

There is another language, Probabilistic Answer Set Programming (PASP) [32] which superficially may look similar to P-log. However, according to the authors of PASP, the languages are actually very different: "Even though both PASP and P-log are both methods of extending ASP with probabilities, their focus are very different and they should be used for different tasks."

There is a number of other interesting languages which have much in common with P-log. Logic of Causal Probability (CP-logic) introduced in [45] is a logical language for representing probabilistic causal laws. It is based on the well-founded semantics of logic programs. Here is how the authors of the language comment on the relationship between CP-logic and P-log: "In summary, the scope of P-log is significantly broader than that of CP-logic and it is a more full-blown knowledge representation language than CP-logic, which is only aimed at expressing a specific kind of probabilistic causal laws. However, when it comes to representing just this kind of knowledge, CP-logic offers the same advantages over P-log that it does over Bayesian networks."

There are also some similarities between P-log and another popular language combining logic programming and probability - Problog [13,15]. The underlying semantics of the original Problog [42] is close to that of PRISM [44]. Currently, the semantics is defined for programs with unique stable model which coincides with the program's two valued well-founded model. While the relationship between P-log and PRISM had been studied earlier, no such comparison between P-log and Problog is known to the authors. There are some features of P-log like sorts, non-boolean attributes, dynamic domains, cr-rules, *do* operator, clear separation between logical and probabilistic constructs of the language, etc., which, in our opinion, make it more powerful knowledge representation language. On another hand, unlike P-log, Problog has a well developed reasoning system (which, among other things, is capable of parametric learning) and good on-line documentation. We hope that a more detailed and accurate comparison between the languages will be made in the future.

## 9 Conclusion and Future Work

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. The new features are illustrated by several examples which may also be of interest from the standpoint of P-log programming methodology.
- 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 $\mathcal{A}$ 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 programs from $\mathcal{A}$ which do not belong to $\mathcal{B}$. We were not able, however, to find realistic examples of such programs.

- Defined the notion of coherency for programs of P-log with cr-rules, expanded the class $\mathcal{B}$ to allow such rules, and proved that programs from the new class $B^+$ are coherent according to the new definition.
- Gave an informal discussion comparing P-log and several other recently introduced languages combining ASP and probabilistic reasoning including Credal Logic.

We believe that this work made P-log a clearer and more expressive knowledge representation language, and added some new insights into its features and 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 [22,21]. We also plan to expand our implementation to cover programs with cr-rules. 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 needs 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 $\mathrm{LP}^{\mathrm{MLN}}$[27]. A detailed study of the relationship between this formalism and P-log can be found in [28] and [2]. Among other things, the results from these papers would allow us to use P-log inference for $\mathrm{LP}^{\mathrm{MLN}}$, and vice versa. Additional comparison of P-log with other formalisms combining logic and probability deserves further studies. We plan to continue work on P-log modules and use it to establish relationship between them and credal programs. It also can be interesting to establish formal relationship between P-log and Problog, and investigate if algorithms implemented in inference engines of Problog and other similar languages can be adopted for reasoning in P-log. As mentioned above we are also planning to use the results from this paper to continue our work on the new P-log reasoning system.

**Acknowledgements** We would like to thank Leroy Mason and Nelson Rushton for useful discussions on subjects related to this paper. We also grateful to the anonymous reviewers for their comments.

## 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. Bartholomew, M., Lee, J.: Stable models of multi-valued formulas: partial versus total functions. In: Principles of Knowledge Representation and Reasoning: Proceedings of the Fourteenth International Conference, KR 2014, Vienna, Austria, July 20-24, 2014 (2014). URL http://www.aaai.org/ocs/index.php/KR/KR14/paper/view/8020
11. Cabalar, P.: Partial functions and equality in answer set programming. In: Logic Programming, 24th International Conference, ICLP 2008, Udine, Italy, December 9-13 2008, Proceedings, pp. 392–406 (2008). DOI 10.1007/978-3-540-89982-2_36. URL https://doi.org/10.1007/978-3-540-89982-2_36
12. Cozman, F.G., Mauá, D.D.: On the semantics and complexity of probabilistic logic programs. Journal of Artificial Intelligence Research **60**, 221–262 (2017)
13. De Raedt, L., Kimmig, A.: Probabilistic (logic) programming concepts. Machine Learning **100**(1), 5–47 (2015)
14. 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
15. Fierens, D., Van den Broeck, G., Renkens, J., Shterionov, D., Gutmann, B., Thon, I., Janssens, G., De Raedt, L.: Inference and learning in probabilistic logic programs using weighted boolean formulas. Theory and Practice of Logic Programming **15**(3), 358–401 (2015)
16. Gelfond, M., Kahl, Y.: Knowledge representation, reasoning, and the design of intelligent agents: The answer-set programming approach. Cambridge University Press (2014)
17. 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)
18. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New Generation Computing **9**(3/4), 365–386 (1991)
19. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New generation computing **9**(3-4), 365–385 (1991)
20. 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)
21. 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
22. 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)
23. Halpern, J.Y.: A modification of the Halpern-Pearl definition of causality. In: Proceedings of the 24th International Conference on Artificial Intelligence, IJCAI'15, pp. 3022–3033. AAAI Press (2015). URL http://dl.acm.org/citation.cfm?id=2832581.2832671

24. Halpern, J.Y., Pearl, J.: Causes and explanations: a structural-model approach. Part I: Causes. The British journal for the philosophy of science **56**(4), 843–887 (2005)
25. Jaynes, E.T.: Probability theory: The logic of science. Cambridge university press (2003)
26. Kersting, K., Raedt, L.D.: Bayesian logic programs. In: Inductive Logic Programming, 10th International Conference, ILP 2000, Work-in-progress reports, London, UK, July 2000, Proceedings (2000). URL http://ceur-ws.org/Vol-35/07-KerstingDeRaedt.ps
27. 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
28. Lee, J., Yang, Z.: LP$^{\text{MLN}}$, 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)
29. 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)
30. Lukasiewicz, T.: Probabilistic logic programming. In: ECAI, pp. 388–392 (1998)
31. Lukasiewicz, T.: Probabilistic description logic programs. International Journal of Approximate Reasoning **45**(2), 288–307 (2007)
32. De Morais, E.M., Finger, M.: Probabilistic answer set programming. In: Brazilian Conference on Intelligent Systems, BRACIS 2013, Fortaleza, CE, Brazil, 19-24 October, 2013, pp. 150–156 (2013). DOI 10.1109/BRACIS.2013.33. URL https://doi.org/10.1109/BRACIS.2013.33
33. Muggleton, S., et al.: Stochastic logic programs. Advances in inductive logic programming **32**, 254–264 (1996)
34. Ng, R., Subrahmanian, V.S.: Probabilistic logic programming. Information and computation **101**(2), 150–201 (1992)
35. Ngo, L., Haddawy, P.: Answering queries from context-sensitive probabilistic knowledge bases. Theoretical Computer Science **171**(1-2), 147–177 (1997)
36. Oikarinen, E., Janhunen, T.: Modular equivalence for normal logic programs. In: ECAI, vol. 6, pp. 412–416 (2006)
37. Pearl, J.: Reasoning with cause and effect. In: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, IJCAI 99, Stockholm, Sweden, July 31 - August 6, 1999. 2 Volumes, 1450 pages, pp. 1437–1449 (1999)
38. Pearl, J.: Causality: Models, Reasoning, and Inference. Cambridge University Press (2000)
39. Pereira, L.M., Saptawijaya, A.: Programming Machine Ethics, 1st edn. Springer Publishing Company, Incorporated (2016)
40. Poole, D.: Probabilistic horn abduction and bayesian networks. Artificial Intelligence **64**, 81–129 (1993)
41. Poole, D.: The independent choice logic for modelling multiple agents under uncertainty. Artificial intelligence **94**(1-2), 7–56 (1997)
42. Raedt, L.D., Kimmig, A., Toivonen, H.: Problog: A probabilistic prolog and its application in link discovery. In: IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007, pp. 2462–2467 (2007). URL http://ijcai.org/Proceedings/07/Papers/396.pdf
43. Sato, T.: A statistical learning method for logic programs with distribution semantics. In: In Proceedings of the 12th International Conference on Logic Programming (ICLP'95), pp. 715–729. MIT Press (1995)
44. Sato, T., Kameya, Y.: PRISM: A language for symbolic-statistical modeling. In: Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence, IJCAI 97, Nagoya, Japan, August 23-29, 1997, 2 Volumes, pp. 1330–1339 (1997). URL http://ijcai.org/Proceedings/97-2/Papers/078.pdf
45. Vennekens, J., Denecker, M., Bruynooghe, M.: CP-logic: a language of causal probabilistic events and its relation to logic programming. Theory and practice of logic programming **9**(3), 245–308 (2009)
46. Vennekens, J., Verbaeten, S., Bruynooghe, M.: Logic programs with annotated disjunctions. In: Logic Programming, 20th International Conference, ICLP 2004, Saint-Malo, France, September 6-10, 2004, Proceedings, pp. 431–445 (2004). DOI 10.1007/978-3-540-27775-0_30. URL https://doi.org/10.1007/978-3-540-27775-0_30
47. Wellman, M.P., Henrion, M.: Explaining 'explaining away'. IEEE Transactions on Pattern Analysis and Machine Intelligence **15**(3), 287–292 (1993)

48. 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)
49. Zhu, W.: Plog: Its algorithms and applications. Ph.D. thesis, Texas Tech University (2012)

## Appendix A: Definition of Probability

In this appendix we formally define the probabilistic measure defined by a P-log program satisfying Conditions 1-3 from Section 3 . The definition is very similar to the one from [8], with a small difference related to the addition of rules names to special terms *random*, *truly_random*, *do* and probability atoms and the corresponding changes in axioms, introduced in this paper.

Let $\Pi$ be a P-log program with signature $\Sigma$, $W$ be an interpretation of $\Sigma$, $a$ be an attribute term of $\Sigma$. Consider random selection rule of the form

$$random(rn, a, p) \leftarrow B$$

such that $W$ satisfies $B$. Let $PO(W, rn, a)$ be the set of constants defined as follows:

$$PO(W, rn, a) = \{y \mid W \text{ satisfies } p(y) \text{ and } y \in range(a)\}.$$

We will refer to elements of the set $PO(W, rn, a)$ as *possible outcomes* of $a$ in $W$ via rule $rn$, and to every atom $a = y$ s.t. $y \in PO(W, rn, a)$ as a *possible atom* in $W$ via $rn$. Note that, by Condition 1, there can be at most one rule such that $a = y$ is possible in $W$ vie that rule, so we will sometimes say that $a = y$ is possible in $W$ if there is a rule $rn'$ such that $a = y$ is possible in $W$ via $rn'$.

Let $\Pi$ be a P-log program and $a$ be a random attribute term of the signature of $\Pi$. For every possible world $W$ of $\Pi$ and every possible atom $a = y$ in $W$ via some rule $random(r, a, p) \leftarrow B$, such that $W \models truly\_random(r, a)$, we will define the corresponding causal probability $P(W, a = y)$. Whenever possible, the probability of an atom $a = y$ will be directly assigned by pr-atoms of the program and denoted by $PA(W, a = y)$. To define probabilities of the remaining atoms we assume that by default, all values of a given attribute which are not assigned a probability by pr-atoms are equally likely. Their probabilities will be denoted by $PD(W, a = y)$. ($PA$ stands for *assigned probability* and $PD$ stands for *default probability*).

More precisely, for each atom $a = y$ possible in $W$ via some rule $r$ or $\Pi$:

1. Assigned probability:
   If $\Pi$ contains $pr(r, a = y \mid B') = v$, $W \models B'$, then
   $$PA(W, a = y) = v$$
   (note that Condition 2 implies that the probability is uniquely defined).
2. Default probability:
   Let
   $$A_a(W) = \{y \mid a = y \text{ is possible in } W \text{ and } PA(W, a = y) \text{ is defined}\},$$
   $$D_a(W) = \{y \mid a = y \text{ is possible in } W\} \setminus A_a(W)$$
   and $\alpha_a(W) = \sum_{y \in A_a(W)} PA(W, a = y)$.
   The default probability of $a = y$ in $W$ is defined as follows:
   $$PD(W, a = y) = \frac{1 - \alpha_a(W)}{|D_a(W)|}$$

3. Finally, the causal probability $P(W, a = y)$ of $a = y$ in $W$ is defined by:

$$P(W, a = y) = \begin{cases} PA(W, a = y) & \text{if } y \in A_a(W) \\ PD(W, a = y) & \text{otherwise.} \end{cases}$$

**Definition 15 (Measure)**

1. Let $W$ be an interpretation of $\Pi$. The *unnormalized probability*, $\hat{\mu}_\Pi(W)$, of $W$ *induced by* $\Pi$ is

$$\hat{\mu}_\Pi(W) = \prod_{W(a)=y} P(W, a = y)$$

where the product is taken over atoms for which $P(W, a = y)$ is defined.

2. Suppose $\Pi$ is a P-log program having at least one possible world with nonzero unnormalized probability. The *measure*, $\mu_\Pi(W)$, of a possible world $W$ *induced by* $\Pi$ is the unnormalized probability of $W$ divided by the sum of the unnormalized probabilities of all possible worlds of $\Pi$, i.e.,

$$\mu_\Pi(W) = \frac{\hat{\mu}_\Pi(W)}{\sum_{W_i \in \Omega(\Pi)} \hat{\mu}_\Pi(W_i)}$$

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

**Definition 16 (Probability)**
Suppose $\Pi$ 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 $\Pi$ is the sum of the measures of the possible worlds from $E$, i.e.

$$P_\Pi(E) = \sum_{W \in E} \mu_\Pi(W).$$

When $\Pi$ is clear from the context we may simply write $P$ instead of $P_\Pi$.

**Definition 17 (Probability of a literal)**
The *probability* with respect to program $\Pi$ of a literal $l$ of $\Pi$, $P_\Pi(l)$, is the sum of the measures of the possible worlds of $\Pi$ in which $l$ is true, i.e.

$$P_\Pi(l) = \sum_{W \models l} \mu_\Pi(W).$$

Note that, given that conditions 1-3 are satisfied, the function $P_\Pi$ is defined iff

$$\sum_{W_i \in \Omega(\Pi)} \hat{\mu}_\Pi(W_i) \neq 0$$

## Appendix B: Proof of the Main Result

We prove Theorem 1 in 3 steps. In section B.1 we describe a translation $\tau$ from P-log programs into ASP programs and show the relationship between the possible worlds of a given P-log program $\Pi$ and answer sets of its translation $\tau(\Pi)$. Then, in section B.2 we formulate splitting set theorem for P-log originally defined in [29] for Answer Set Prolog programs. Finally, in section B.3 we prove theorem 1 using the results from sections B.1 and B.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.

### B.1 Translation from P-log to ASP

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

1. if $f(\bar{x}) = y$ is a literal of $\Sigma$, $\tau(f(\bar{x}) = y)$ is $f(\bar{x}, y)$;
2. if $f(\bar{x}) \neq y$ is a literal of $\Sigma$, $\tau(f(\bar{x}) \neq y)$ is $\neg f(\bar{x}, y)$;
3. if $r$ is a rule of $\Pi$, $\tau(r)$ is an ASP rule obtained from $r$ by replacing all occurrences of literals in the rule with their translations;
4. if $\Pi$ is a P-log program with signature $\Sigma$, $\tau(\Pi)$ 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
   $$\neg f(\bar{x}, y_1) \leftarrow f(\bar{x}, y_2) \tag{32}$$
   for each two atoms $f(\bar{x}) = y_1$ and $f(\bar{x}) = y_2$ of $\Sigma$ such that $y_1 \neq y_2$;
5. if $A$ is a set of atoms of $\Sigma$, 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 $\Sigma$, 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 $\Sigma$, then $I$ satisfies a literal $l$ of $\Sigma$ 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 $\tau$, $\tau(I)$ 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 $\Pi$ be a P-log program not necessarily containing all general axioms. An interpretation $W$ of $\Pi$ is a possible world of $\Pi$ if and only if $\tau(W)$ is an answer set of $\tau(\Pi)$

*Proof*

$\Rightarrow$ Let $W$ be a possible world of $\Pi$. 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 $\tau$; 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 $\tau$ and the definition of $R_a$, we conclude that there is a rule $r'$ in $\Pi^W$ 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 $\Pi$, $W$ 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
$$A \text{ satisfies all the rules in } \tau(\Pi)^{\tau(W)} \tag{33}$$
and
$$A \text{ is a proper subset of } \tau(W) \tag{34}$$
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 $\Sigma$ 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 $\Pi^W$. 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 $\Pi^W$).

(a) We prove that $I$ satisfies the rules of $\Pi^W$. Let $r$ be a rule of $\Pi^W$ 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 $\Pi^W$, $r$ 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
   $$|\tau(W)^+| = |A^+| \tag{35}$$
   (and, even more precisely, $\tau(W)^+ = A^+$)
   iii. By definition of $\tau(W)$,
   $$|\tau(W)^-| = \sum_{f(\bar{x}) \in \{f(\bar{x}) \mid \exists y : W(f(\bar{x}))=y\}} (|range(f(\bar{x}))| - 1) \tag{36}$$
   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 (32) in $\tau(\Pi)$. Since $A$ satisfies the rules of $\tau(W)$, in particular, those of the form (32), from (35) and the construction of $\tau(W)$, it follows that the number of negative literals in A is bounded below as follows:
   $$|A^-| \geq \sum_{f(\bar{x}) \in \{f(\bar{x}) \mid \exists y : W(f(\bar{x}))=y\}} (|range(f(\bar{x}))| - 1) \tag{37}$$
   v. By combining equation (36) and inequality (37), we get
   $$|A^-| \geq |\tau(W)^-| \tag{38}$$

vi. From equation (35) and inequality (38) we have

$$\begin{aligned}|A| &= |A^+| + |A^-| \\ &= |\tau(W)^+| + |A^-| \\ &\geq |\tau(W)^+| + |\tau(W)^-| \\ &= |\tau(W)| \end{aligned} \tag{39}$$

that contradicts our original assumption (34) 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 $\Pi$ such that $A = \tau(I)$. We prove that $I$ is a possible world of $\Pi$. In 5) we show that $I$ satisfies the rules of $\Pi^I$ and in 6) we show the minimality of $I$.

5) We prove that $I$ satisfies the rules of $\Pi^I$. Let $r$ be a rule of $\Pi^I$ 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 $\Pi^I$ 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 $\tau$ and the definition of reducts in ASP and P-log it follows that $\Pi^I$ 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 $\Pi^I$, $I'$ 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 $I'$,

$$I' \subsetneq I \tag{40}$$

Thus, by definition of $\tau$,

$$\tau(I')^+ \text{ is a proper subset of } \tau(I)^+ \tag{41}$$

From (41) it follows immediately

$$|\tau(I')^+| < |\tau(I)^+| \tag{42}$$

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

$$\tau(I)^- = \bigcup_{f(\bar{x})=y \in I} \{\neg f(\bar{x}) = y_1) | y_1 \in range(f(\bar{x})) \wedge y_1 \neq y\} \tag{43}$$

$$\tau(I')^- = \bigcup_{f(\bar{x})=y \in I'} \{\neg f(\bar{x}) = y_1) | y_1 \in range(f(\bar{x})) \wedge y_1 \neq y\} \tag{44}$$

From (40), (43) and (44) it follows that

$$\tau(I')^- \subseteq \tau(I)^- \tag{45}$$

From (41) and (45) and the fact that $\tau(I) = \tau(I)^+ \cup \tau(I)^-$ and $\tau(I') = \tau(I')^+ \cup \tau(I')^-$ we get

$$\tau(I') \text{ is a proper subset of } \tau(I) \tag{46}$$

**Proposition 1** Let $\Pi$ be a P-log program and $W_1, W_2$ be two possible worlds of $\Pi$. 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 $\Pi$ such that

$$W_1 \subsetneq W_2 \tag{47}$$

By Lemma 2,

$$\tau(W_1) \text{ and } \tau(W_2) \text{ are answer sets of } \tau(\Pi). \tag{48}$$

By definition of $\tau$,

$$\tau(W_1) = W_1 \cup \bigcup_{f(\bar{x})=y \in W_1} \{f(\bar{x}) \neq y_1 | y_1 \in range(f(\bar{x})) \wedge y_1 \neq y\} \tag{49}$$

$$\tau(W_2) = W_2 \cup \bigcup_{f(\bar{x})=y \in W_2} \{f(\bar{x}) \neq y_1 | y_1 \in range(f(\bar{x})) \wedge y_1 \neq y\} \tag{50}$$

From (47), (49) and (50) it follows that

$$\tau(W_1) \subsetneq \tau(W_2) \tag{51}$$

(51) and (48) contradict the theorem about minimality of answer sets of ASP programs (see Lemma 1 in [19]).

## B.2 Splitting Set Theorem For P-log

In this section we present the P-log version of the original Splitting Set Theorem from [29]. The adoption requires change in the definition of splitting set (see Definition 21). Other definitions follow [29] and are presented for completeness.

Let $\Pi$ be a program with signature $\Sigma$, and $X$ and $U$ be sets of literals of $\Sigma$. As in [29], we will define the *bottom* and the *top* of a program $\Pi$ 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

$$l \leftarrow l_1, \ldots, l_k, not\ l_{k+1}, \ldots, not\ l_m \tag{52}$$

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 18 (Bottom w.r.t. $U$)**
The bottom of $\Pi$ 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 $\Sigma$ 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 19 (Top w.r.t. $X$ and $U$)**
The top of $\Pi$ 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
$$\{R_U(r) \mid r \in \Pi \text{ and } pos(r) \cap U \text{ is satisfied by } X \text{ and every e-literal} \\ \text{from } \{not\ l \mid l \in neg(r) \cap U\} \text{ is not satisfied by } X\}$$
2. the signature of $e_U(\Pi, X)$ consists of all attribute terms of $\Sigma$ which do not form a literal in $U$.

□

We borrow the definition of a solution to $\Pi$ w.r.t $U$:

**Definition 20 (Solution to $\Pi$ w.r.t $U$)**
Let $\Pi$ be a P-log program. A solution to $\Pi$ 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 21 (Splitting set)**
A *splitting set* for a P-log program $\Pi$ is any set U of literals of $\Pi$'s signature such that,

1. for every rule $r$ of $\Pi$ 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 $\Sigma$ formed by $f(\bar{x})$ belong to $U$.

□

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

**Theorem 3** *[Splitting Set Theorem]*
Let $U$ be a splitting set for a program $\Pi$. A set $A$ of literals is a possible world of $\Pi$ if and only if $A = X \cup Y$ for some solution $\langle X, Y\rangle$ to $\Pi$ with respect to $U$.

□

Note that the condition 2 from Definition 21, 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 $\Pi$. 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 $\Pi$ with respect to $U$.

*Proof (Proof for theorem 3)*
In 1 we show that $\tau(U)$ is a splitting set (as defined in [29]) 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 $\Pi$, then $A = X \cup Y$ for some solution $\langle X, Y\rangle$ to $\Pi$ 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 $\Pi$ with respect to $U$, then $A$ is a possible world of $\Pi$.

1. We show that $\tau(U)$ is a splitting set (as defined in [29]) for the program $\tau(\Pi)$. Let $r$ be a rule of $\tau(\Pi)$ such that:

$$\text{the head of } r \text{ is included into } \tau(U) \tag{53}$$

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 $\Pi$ such that $r = \tau(r')$. In this case, by construction of $\tau(r')$ and clause 1 of Definition 21 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 (53) we have that $f(x) \neq y_1 \in U$. Therefore, by clause 2) of definition 21 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:

$$\tau(b_U(\Pi)) = b_{\tau(U)}(\tau(\Pi)) \tag{54}$$

We prove (54) in two directions:

$$\tau(b_U(\Pi)) \subseteq b_{\tau(U)}(\tau(\Pi)) \tag{55}$$

$$b_{\tau(U)}(\tau(\Pi)) \subseteq \tau(b_U(\Pi)) \tag{56}$$

We start from (55). Suppose $r$ is a a rule such that

$$r \in \tau(b_U(\Pi)) \tag{57}$$

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 $\Pi$. Therefore,

$$r \text{ belongs to } \tau(\Pi) \tag{58}$$

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:

$$\text{every literal occurring in } r \text{ is from } \tau(U) \tag{59}$$

From (59) and (58) we have

$$r \in b_{\tau(U)}(\tau(\Pi)) \tag{60}$$

Therefore, (55) holds.

– $r$ is of the form $\neg f(\bar{x}, y_1) \leftarrow f(\bar{x}, y_2)$ In this case, by definition of $\tau$ from (57) 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 (55) holds.

We next show (56). Suppose $r$ is a a rule such that

$$r \in b_{\tau(U)}(\tau(\Pi)) \tag{61}$$

There are two possible cases:

– $r$ is a translation of some rule $r'$ of $\Pi$. In this case, from (61) 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, (56) 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:

$$\neg f(\bar{x}, y_1) \in \tau(U) \tag{62}$$

$$f(\bar{x}, y_2) \in \tau(U) \tag{63}$$

and

$$r \in \tau(\Pi) \tag{64}$$

From (62) and (63) we have:

$$\text{all the literals of } \Sigma \text{ formed by } f(\bar{x}) \text{ belong to } U \tag{65}$$

From (65) we have that

$$f(\bar{x}) \text{ belongs to the signature of } b_U(\Pi) \tag{66}$$

Therefore, by definition of $\tau(\Pi)$, we have $r \in \tau(b_U(\Pi)$. Therefore, (56) holds.

From (55) and (56) we have (54).

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

$$e_{\tau(U)}(\tau(\Pi) \setminus b_{\tau(U)}(\tau(\Pi)), \tau(X)) = \tau(e_U(\Pi \setminus b_U(\Pi), X)) \tag{67}$$

We prove (67) in two directions:

$$e_{\tau(U)}(\tau(\Pi) \setminus b_{\tau(U)}(\tau(\Pi)), \tau(X)) \subseteq \tau(e_U(\Pi \setminus b_U(\Pi), X)) \tag{68}$$

$$\tau(e_U(\Pi \setminus b_U(\Pi), X)) \subseteq e_{\tau(U)}(\tau(\Pi) \setminus b_{\tau(U)}(\tau(\Pi)), \tau(X)) \tag{69}$$

We start from (68). Let $r$ be a rule s.t.

$$r \in e_{\tau(U)}(\tau(\Pi) \setminus b_{\tau(U)}(\tau(\Pi)), \tau(X)) \tag{70}$$

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

$$r' \in \tau(\Pi) \tag{71}$$

$$r' \notin b_{\tau(U)}(\tau(\Pi)) \tag{72}$$

$$pos(r') \cap \tau(U) \subseteq \tau(X) \tag{73}$$

$$neg(r') \cap \tau(U) \cap \tau(X) = \emptyset \tag{74}$$

From (72) we have:

$$r' \text{ contains a literal which is not from } \tau(U) \tag{75}$$

There are only two possibilities:

– $r' = \tau(r'')$ for some rule $r''$ from $\Pi$. From (73) and (c) by lemma 1 we have:

$$pos(r'') \cap U \text{ is satisfied by } X \tag{76}$$

From (74) and (c) by lemma 1 we have:

$$\text{every literal in } \{not\ l \mid l \in neg(r'') \cap U\} \text{ is not satisfied by } X \tag{77}$$

From (75) we have:

$$lit(r'') \text{ contains a literal which is not in } U \tag{78}$$

Therefore,

$$r'' \notin b_U(\Pi) \tag{79}$$

From (76),(77) and (79) we have:

$$R_U(r'') \in e_U(\Pi \setminus b_U(\Pi), X) \tag{80}$$

Therefore,

$$\tau(R_U(r'')) \in \tau(e_U(\Pi \setminus b_U(\Pi), X)) \tag{81}$$

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

$$\tau(R_U(r'')) = R_{\tau(U)}(r') \tag{82}$$

From (81) and (82) we have:

$$R_{\tau(U)}(r') \in \tau(e_U(\Pi \setminus b_U(\Pi), X)) \tag{83}$$

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

$$r \in \tau(e_U(\Pi \setminus b_U(\Pi), X)) \tag{84}$$

Therefore, in this case, (68) 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

$$f(\bar{x}) \text{ is an attribute term of } \Sigma \tag{85}$$

From (75) by clause 2 of the definition of the splitting set we have:

$$\text{no literal formed by } f(\bar{x}) \text{ belongs to U} \tag{86}$$

From (85) and (86) by Definition 19 we have:

$$f(\bar{x}) \text{ belongs to the signature of } e_U(\Pi \setminus b_U(\Pi), X) \tag{87}$$

From (86) we have:

$$R_{\tau(U)}(r') = r' = r \tag{88}$$

From (87) we have:

$$r' \in \tau(e_U(\Pi \setminus b_U(\Pi), X)) \tag{89}$$

From (89) and (88) we have:

$$r \in \tau(e_U(\Pi \setminus b_U(\Pi), X)) \tag{90}$$

Therefore, in this case, (68) holds.

We next prove (69).
Let $r$ be a rule s.t.

$$r \in \tau(e_U(\Pi \setminus b_U(\Pi), X)) \tag{91}$$

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:

$$r'' \in \Pi \tag{92}$$

$$r' = R_U(r'') \tag{93}$$

and:

$$r'' \notin b_U(\Pi) \tag{94}$$

$$pos(r'') \cap U \text{ is satisfied by } X \tag{95}$$

$$\text{every literal from } neg(r'') \cap U \text{ is not satisfied by } X \tag{96}$$

From (94) we have:

$$lit(r'') \text{ contains a literal not from } U \tag{97}$$

From (97), clause 2 of the splitting set definition and the construction of $\tau$ we have:

$$lit(\tau(r'')) \text{ contains a literal not from } \tau(U) \tag{98}$$

Therefore,

$$\tau(r'') \notin b_{\tau(U)}(\tau(\Pi)) \tag{99}$$

From (95) and Lemma 1 we have:

$$pos(\tau(r'')) \cap \tau(U) \text{ is satisfied by } \tau(X) \tag{100}$$

From (96) and Lemma 1 we have:

$$\text{every literal from } neg(\tau(r'')) \cap \tau(U) \text{ is not satisfied by } \tau(X) \tag{101}$$

From (92) we have:

$$\tau(r'') \in \tau(\Pi) \tag{102}$$

From (99), (100), (101) and (102) we have:

$$R_{\tau(U)}(\tau(r'')) \in e_{\tau(U)}(\tau(\Pi) \setminus b_{\tau(U)}(\tau(\Pi)), \tau(X)) \tag{103}$$

By construction of $R$ and by Lemma 1 we have:

$$\tau(R_U(r'')) = R_{\tau(U)}(\tau(r'')) \tag{104}$$

From (103) and (104) we have:

$$\tau(R_U(r'')) \in e_{\tau(U)}(\tau(\Pi) \setminus b_{\tau(U)}(\tau(\Pi)), \tau(X)) \tag{105}$$

From (93) and (105) we have:

$$\tau(r') \in e_{\tau(U)}(\tau(\Pi) \setminus b_{\tau(U)}(\tau(\Pi)), \tau(X)) \tag{106}$$

From the fact that $r = \tau(r')$ and (106) we have:

$$r \in e_{\tau(U)}(\tau(\Pi) \setminus b_{\tau(U)}(\tau(\Pi)), \tau(X)) \tag{107}$$

Therefore, in this case, (69) 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 (91) by construction of $\tau(e_U(\Pi \setminus b_U(\Pi), X))$ we have:

$$f(\bar{x}) = y_1 \text{ and } f(\bar{x}) = y_2 \text{ are atoms of the signature of } e_U(\Pi \setminus b_U(\Pi), X) \tag{108}$$

Therefore, by clause 2 of Definition 19, we have:

$$f(\bar{x}) = y_1 \text{ and } f(\bar{x}) = y_2 \text{ are atoms of } \Sigma \tag{109}$$

and

$$\text{no literals of } U \text{ are formed by } f(\bar{x}) \tag{110}$$

Therefore,

$$\tau(U) \text{ contains no literals of the forms } f(\bar{x}, y) \text{ or } \neg f(\bar{x}, y) \tag{111}$$

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

$$\text{every literal in } \{not\ l \mid l \in neg(r) \cap \tau(U)\} \text{ is not satisfied by } \tau(X) \tag{112}$$

From (111) we have $pos(r) \cap \tau(U) = \emptyset$, therefore:

$$\text{every literal in } pos(r) \cap \tau(U) \text{ is satisfied by } \tau(X) \tag{113}$$

From (109) we have:

$$r \in \tau(\Pi) \tag{114}$$

From (111) we have:

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

From (114), (115), (113), (112) we have:

$$R_{\tau(U)}(r) \in e_{\tau(U)}(\tau(\Pi) \setminus b_{\tau(U)}(\tau(\Pi)), \tau(X)) \tag{116}$$

From (111) we have:

$$R_{\tau(U)}(r) = r \tag{117}$$

From (117) and (116) we have:

$$r \in e_{\tau(U)}(\tau(\Pi) \setminus b_{\tau(U)}(\tau(\Pi)), \tau(X)) \tag{118}$$

Therefore, in this case, (69) holds.

From (68) and (69) we have (67).

4. $\rightarrow$ we show that if $A$ a possible world of $\Pi$, then $A = X \cup Y$ for some solution $\langle X, Y\rangle$ to $\Pi$ with respect to $U$.

Let $A$ be a possible world of $\Pi$. By lemma 2 we have that $\tau(A)$ is an answer set of $\tau(\Pi)$. By splitting set theorem from [29] we have that $\tau(A) = X' \cup Y'$ for some solution $\langle X', Y'\rangle$ to $\tau(\Pi)$ with respect to $\tau(U)$. Let $\Sigma$ be the signature of $\Pi$. We will construct two sets of atoms of $\Sigma$, $X$ and $Y$, such that

- $X' = \tau(X)$
- $Y' = \tau(Y)$
- $\langle X, Y\rangle$ is a solution to $\Pi$ 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 $\Pi$ 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

$$\text{if } f(\bar{x}, y) \text{ occurs as an atom in } \tau(\Pi) \text{ then } f(\bar{x}) = y \text{ is an atom of } \Sigma \tag{119}$$

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 $b_{\tau(U)}(\tau(\Pi))$,

$$\text{all literals in } X' \text{ occur in } \tau(\Pi) \tag{120}$$

Let $X$ be the set defined as follows:

$$X = \{f(\bar{x}) = y \mid f(\bar{x}, y) \in X'\} \tag{121}$$

From (119) and (120) we have that $X$ is a set of atoms of $\Sigma$.

(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 $\Sigma$. Then the set $Y$, defined as follows

$$Y = \{f(\bar{x}) = y \mid f(\bar{x}, y) \in Y'\}, \tag{122}$$

is a set of atoms of $\Sigma$.

(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

$$f(\bar{x}) = y \in X \tag{123}$$

in $X$,

$$\{\neg f(\bar{x}, y_1) \mid y_1 \neq y\} \subseteq X' \tag{124}$$

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

$$r \text{ belongs to } b_{\tau(U)}(\tau(\Pi)) \tag{125}$$

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 (125)

Since $X'$ is an answer set of $b_{\tau(U)}\tau(\Pi)$, from (125) we have:

$$X' \text{ satisfies } r \tag{126}$$

From (123) and (121) we have that $X'$ satisfies the body of $r$. Therefore, from (126) we have $X'$ also satisfies the head of $r$, which is $\neg f(\bar{x}, y_1)$. Therefore, (124) 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

$$f(\bar{x}) = y \in Y \tag{127}$$

we have

$$\{\neg f(\bar{x}, y_1) \mid y_1 \neq y\} \subseteq Y' \tag{128}$$

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

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

such that

$$r \in \tau(\Pi) \tag{129}$$

We prove that

$$r \in e_{\tau(U)}(\tau(\Pi) \setminus b_{\tau(U)}(\tau(\Pi)), X') \tag{130}$$

From the results on page 5 of [29], we have:

$$Y' \cap \tau(U) = \emptyset \tag{131}$$

From (131), (127) and (122) we have:

$$f(\bar{x}, y) \notin \tau(U) \tag{132}$$

From (132) by construction of $b_{\tau(U)}(\tau(\Pi))$ we have:

$$r \notin b_{\tau(U)}(\tau(\Pi)) \tag{133}$$

From (133) and (129) we have:

$$r \in \tau(\Pi) \setminus b_{\tau(U)}(\tau(\Pi)) \tag{134}$$

From (134) and (132) by definition of top we have (130).

Since $Y'$ is an answer set of $e_{\tau(U)}(\tau(\Pi) \setminus b_{\tau(U)}(\tau(\Pi)), X')$, from (130) we have:

$$Y' \text{ satisfies } r \tag{135}$$

From (127) and (122) we have that $Y'$ satisfies the body of $r$. Therefore, from (135) we have $Y'$ also satisfies the head of the rule, which is $\neg f(\bar{x}, y_1)$. Therefore, (128) holds.

(e) We show that $\langle X, Y \rangle$ is a solution to $\Pi$ with respect to $U$. We prove the clauses 1-3 of Definition 20 in i - iii respectively.

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

$$X' \text{ is an answer set of } b_{\tau(U)}(\tau(\Pi)) \tag{136}$$

From (c), the fact that $X'$ is an answer set of $b_{\tau(U)}(\tau(\Pi))$, 54 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:

$$Y' \text{ is an answer set of } e_{\tau(U)}(\tau(\Pi) \setminus b_{\tau(U)}(\tau(\Pi)), X') \tag{137}$$

From (137), (d), and (67) 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 $\Pi$. 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 $\Pi$ with respect to $U$, then $A$ is a possible world of $\Pi$.
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

$$\tau(X) \text{ is a possible world of } b_{\tau(U)}(\tau(\Pi)) \tag{138}$$

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

$$X \text{ is a possible world of } b_U(\Pi) \tag{139}$$

Therefore by Lemma 2 we have:

$$\tau(X) \text{ is a possible world of } \tau(b_U(\Pi)) \tag{140}$$

From (54) and (140) we have (138)

(b) We show that

$$\tau(Y) \text{ is a possible world of } e_{\tau(U)}(\tau(\Pi) \setminus b_{\tau(U)}(\tau(\Pi)), \tau(X)) \tag{141}$$

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

$$Y \text{ is a possible world of } e_U(\tau(\Pi) \setminus b_U(\tau(\Pi)), X) \tag{142}$$

Therefore by Lemma 2 we have:

$$\tau(Y) \text{ is a possible world of } \tau(e_U(\tau(\Pi) \setminus b_U(\tau(\Pi)), X)) \tag{143}$$

From (67) and (143) we have (141)

(c) Since $\langle X, Y\rangle$ is a solution of $\Pi$ 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 $\Pi$. If $A$ is a possible world of $\Pi$ such that $A = X \cup Y$ for some solution $\langle X, Y\rangle$ to $\Pi$ 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 19, 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, $Y \cap U = \emptyset$.

## B.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, introduced in lemmas 6 - 25 below.

**Definition 22 (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 23 ($p_T(n)$)**
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 12* Consider the tree $T$ from Figure 1. If $n$ is the node labeled (13), then $p_T(n) = \{1, 3, 8, 13\}$.

□

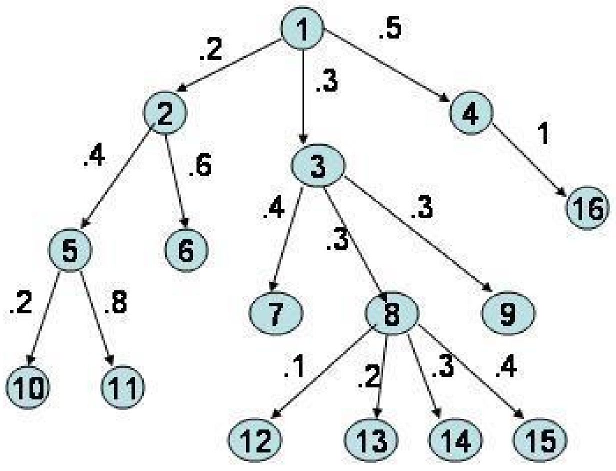


Fig. 1: Unitary tree T

**Definition 24 (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 13* 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 25 (A set of literals compatible with an e-literal)**
A set $S$ of literals of $\Pi$ is *$\Pi$-compatible* with an e-literal $l$ of $\Pi$ if there exists a possible world of $\Pi$ satisfying $S \cup \{l\}$. Otherwise $S$ is *$\Pi$-incompatible* with $l$. $S$ is *$\Pi$-compatible* with a set $B$ of e-literals of $\Sigma$ if there exists a possible world of $\Pi$ satisfying $S \cup B$; otherwise $S$ is *$\Pi$-incompatible* with $B$. □

**Definition 26 (A set of literals guaranteeing an e-literal)**
A set $S$ of literals is said to *$\Pi$-guarantee* an e-literal $l$ if $S$ and $l$ are $\Pi$-compatible and every possible world of $\Pi$ satisfying $S$ also satisfies $l$; $S$ *$\Pi$-guarantees* a set $B$ of e-literals if $S$ $\Pi$-guarantees every member of $B$. □

**Definition 27 (Ready to branch)**
Let $T$ be a tree whose nodes are labeled with atoms of $\Sigma$ and $r$ be a rule of $\Pi$ of the form

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

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

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

If $\Pi$ 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 $\Pi$, and $a(\bar{t}) = y$ is $\Pi$-compatible with $p_T(n)$; and let $W_1$ and $W_2$ be possible worlds of $\Pi$-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 $\Pi$, and $a(\bar{t}) = y$ is $\Pi$-compatible with $p_T(n)$; and let $W_1$ and $W_2$ be possible worlds of $\Pi$ 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 $pr(rn, a(\bar{t}) = y \mid B) = v$ of $\Pi$ such that $W_1$ satisfies $B$. Since $W_1$ also satisfies $p_T(n)$, $B$ is $\Pi$-compatible with $p_T(n)$. It follows from the definition of ready-to-branch that $p_T(n)$ $\Pi$-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 $\Pi$-compatible with $p(z)$. By definition of ready-to-branch, it follows that $p_T(n)$ $\Pi$-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 $\Pi$, and $a(\bar{t}) = y$ is $\Pi$-compatible with $p_T(n)$, and $W$ is a possible world of $\Pi$ 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.

For examples of ready-to-branch attributes, see [8] (e.g, Example 32 on page 58).

**Definition 28 (Expanding a node)**
In case $n$ is ready to branch on $a(\bar{t})$ via some rule of $\Pi$, the *$\Pi$-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 $\Pi$-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 29 (Expansions of a tree)**
A *zero-step* $\Pi$-expansion of $T$ is $T$. A *one-step $\Pi$-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 $\Pi$-expansion* of $T$ is a one-step $\Pi$-expansion of an $(n-1)$-step $\Pi$-expansion of $T$. A *$\Pi$-expansion* of $T$ is an $n$-step $\Pi$-expansion of $T$ for some non-negative integer $n$.
□

*Example 14* Consider the program $\Pi_7$:

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

along with the following trees:

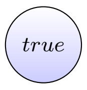


Fig. 2: $T_1$

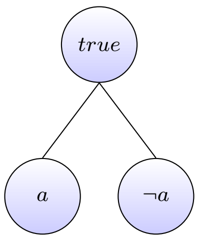


Fig. 3: $T_2$

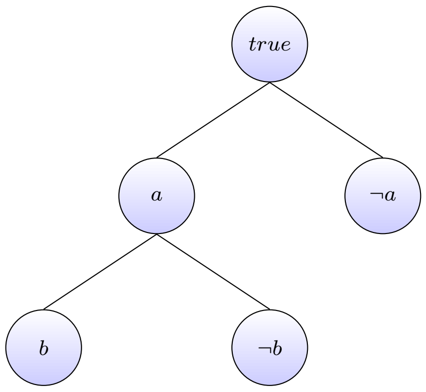


Fig. 4: $T_3$

In all the figures, *true* denotes an atom which is true in any interpretation of $\Sigma$ ($1 = 1$ is an example of such an atom). We can say that each of the trees $T_1$ - $T_3$ is a 0-step $\Pi_7$-expansion of itself, that $T_2$ is a 1-step $\Pi_7$-expansion of $T_1$, and that $T_3$ is a 1-step $\Pi_7$-expansion of $T_2$ and a 2-step $\Pi_7$-expansion of $T_1$.

□

**Definition 30 (Seed)**
A *seed* is a tree with a single node labeled *true*.

□

**Definition 31 (Tableau)**
A *tableau* of $\Pi$ is a $\Pi$-expansion of a seed which is maximal with respect to the subtree relation.

□

For instance, a tree $T_3$ of Figure 4 is a tableau of $\Pi_7$.

**Definition 32 (Node representing a possible world)**
Suppose $T$ is a tableau of $\Pi$. A possible world $W$ of $\Pi$ is *represented* by a leaf node $n$ of $T$ if $W$ is the set of atoms $\Pi$-guaranteed by $p_T(n)$.

□

For instance, a node $b$ of $T_3$ represents the possible world of $\Pi_7$ containing atoms $a$ and $b$.

**Definition 33 (Tree representing a program)**
If every possible world of $\Pi$ is represented by exactly one leaf node of $T$, and every leaf node of $T$ represents exactly one possible world of $\Pi$, then we say $T$ *represents* $\Pi$. □

It is easy to check that the tree $T_3$ represents $\Pi_7$.

**Definition 34 (Probabilistic soundness)**
Suppose $\Pi$ is a P-log program and $T$ is a tableau representing $\Pi$, such that $R$ is a mapping from the possible worlds of $\Pi$ to the leaf nodes of $T$ which represent them. If for every possible world $W$ of $\Pi$ 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 $\Pi$ 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 $\Pi$ be a program, and $\Pi'$ be a program obtained from $\Pi$ by removing activity records. If there exists a unitary tableau $T$ representing $\Pi'$, and this representation is probabilistically sound, then $P_{\Pi'}$ is defined, and for every pair of rules

$$random(r, a(\bar{t}) : \{X : p(X)\}) \leftarrow K. \tag{144}$$

and

$$pr(r, a(\bar{t}) = y \mid B) = v. \tag{145}$$

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

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

Hence $\Pi$ is coherent.

□

*Proof* Since there exists a unitary tableau representing $\Pi'$, by lemma 4 and Definition 34 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 $p_T(n)$ $\Pi'$-guarantees $S$.

Let $\mu$ denote the normalized measure on possible worlds induced by $\Pi'$.

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

$$P_{\Pi' \cup o(B) \cup o(K)}(a(\bar{t}) = y) = \frac{\sum_{\{W \,:\, W \in \Omega \,\wedge\, a(\bar{t})=y \,\in\, W\}} \mu(W)}{\sum_{\{W \,:\, W \in \Omega\}} \mu(W)} \tag{146}$$

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 $\Pi'$, the right-hand side of (146) can be written as $\alpha/\beta$. So we will be done if we can show that $\alpha/\beta = v$.

We first claim

Every $n \in lgar(B \cup K)$ has a unique ancestor $ga(n)$ which branches on $a(\bar{t})$ via *rule* (144). (147)

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 27.

Next, we claim the following:

For every $n \in lgar(B \cup K)$, $p_T(ga(n))$ $\Pi'$-guarantees $B \cup K$. (148)

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 $\Pi'$. So by clause 3 of Definition 27, $ga(n)$ either $\Pi'$-guarantees $B$ or is $\Pi'$-incompatible with $B$. But $p_T(ga(n)) \subset p_T(n)$, and $p_T(n)$ $\Pi'$-guarantees $B$, so $p_T(ga(n))$ cannot be $\Pi'$-incompatible with $B$. Hence $p_T(ga(n))$ $\Pi'$-guarantees $B$. It also follows from clause 2 of Definition 27 that $p_T(ga(n))$ $\Pi'$-guarantees $K$.

From (148), it follows easily that

$$\text{If } n \in lgar(B \cup K),\ \text{every leaf descended from of } ga(n) \text{ belongs to } lgar(B \cup K). \tag{149}$$

Let

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

In light of (147) and (149), we have

$$lgar(B \cup K) \text{ is precisely the set of leaves descended from nodes in } A. \tag{150}$$

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 $\beta$ will help us complete the proof. Now for $\alpha$. Recall the definition of $\alpha$:

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

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 $lgar(B \cup K)$, (150) 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,

$$\alpha = \sum_{n \in A_y} \sum_{n' \text{ is a leaf descended from } n} pv(n') = \sum_{n \in A_y} pv(n) \tag{151}$$

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 (148), along with Condition 3 of Section 3, and the fact that every node in $A$ branches on $a(\bar{t})$ via rule 144. Thus from (151) 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 $\Pi$ is a program from $\mathcal{B}$ and $U$ is the set of activity records in $\Pi$; 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 $\Pi$ s.t. $\Pi$ is dynamically causally ordered via $\alpha$ and $\Pi_0, \ldots, \Pi_k$ be the dynamic structure of $\Pi \setminus U$ induced by $\alpha$. Let $W_0$ be the possible world of $\Pi_0$.

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 $\Pi_i$ by this term. Let $T = T_k$. We will show that $T_m$ is a tableau of $\Pi$ which represents $\Pi$ and the representation is probabilistically sound.

Our proof will unfold as a sequence of lemmas: Let $\Sigma_i$ be the signature of $\Pi_i$ 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 $\Pi_i$.

**Lemma 7** Let $\Pi$ be a P-log program with signature $\Sigma$, $A$ be a set of attribute terms of $\Sigma$, $L$ be the set of all e-literals of $\Sigma$ formed by attribute terms from $A$. Suppose there exists a set of atoms $W_L \subseteq L$ such that every possible world $W$ of $\Pi$, $W_L \subseteq W$ and $(W \setminus W_L) \cap L = \emptyset$. Let $R \subseteq \Pi$ be a subset of rules of $\Pi$ 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:

$$\Omega_\Pi = \Omega_{\Pi \setminus R} \tag{152}$$

□

*Proof* In 1 we will prove $\Omega_\Pi \subseteq \Omega_{\Pi \setminus R}$. In 2 we will prove $\Omega_{\Pi \setminus R} \subseteq \Omega_\Pi$. (152) follows immediately from 1 and 2.

1. We prove

$$\Omega_\Pi \subseteq \Omega_{\Pi \setminus R} \tag{153}$$

Let $W \in \Omega_\Pi$ be a possible world of $\Pi$. We will show

$$W \in \Omega_{\Pi \setminus R} \tag{154}$$

Consider the reduct $\Pi' = (\Pi \setminus R)^W$. To show (154), 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 $\Pi^W$. 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

$$W' \text{ satisfies } \Pi^W \tag{155}$$

$W'$ satisfies the subset $(\Pi \setminus R)^W$ of $\Pi^W$. Now suppose $r \in R^W$. Let $r^*$ be the rule of $\Pi$ which produced $r$ in $\Pi^W$. 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

$$\Omega_{\Pi \setminus R} \subseteq \Omega_\Pi \tag{156}$$

Let $W \in \Omega_{\Pi \setminus R}$ be a possible world of $\Pi \setminus R$. We will show

$$W \in \Omega_\Pi \tag{157}$$

Consider the reduct $\Pi' = (\Pi)^W$. To show (157), 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 $\Pi$ be a program with signature $\Sigma$ such that the base of $\Pi$ has a unique possible world. We have $\Omega_{red(\Pi)} = \Omega_\Pi$. □

*Proof* Let $L'_0$ be the set of literals of $\Sigma$ each of which does not depend on an a random attribute term of $\Pi$. $L'_0$ is a splitting set of $\Pi$. therefore, by splitting set theorem, for every possible world $W$ of $\Pi$ 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 $\Pi$ 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 $\Pi_i$. 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 $\Pi$'s base signature, and $W_0'$ be the answer set of $\Pi$'s base. By Lemma 9

$$\Omega_{red(\Pi_i)} = \Omega_{\Pi_i} \tag{158}$$

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 $\Pi_m$, then there exists a unique possible world $W_n$ of $\Pi_n$ 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 $\Pi_m$, then there exists a unique possible world $W_0$ of the program $\Pi_0$ such that $(W_m \setminus W_0) \cap L_0 = \emptyset$.
  From the definition of a dynamically causally ordered program, $\Pi_0$ has a unique possible world $W_0$. Therefore, from Lemma (10) every possible world $A_m$ of the program $\Pi_m$ 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 $\Pi_d$ there exists a possible world $W_g$ of the program $\Pi_g$ 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 $\Pi_h$ there exists a possible world $W_j$ of the program $\Pi_j$ 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 $\Pi_j$. 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 $\Pi_j$ 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 $\Pi_j$ is a subset of the rules of the program $\Pi_h$, $r'$ belongs to the rules of $\Pi_h$, and the reduct $\Pi^{W_h}$ will contain the rule $r$. Since $W_j \subset W_h$, $W_h$ satisfies the body of $r$. Therefore, Since $W_h$ is

a possible world of $\Pi_h$, 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 $\Pi_j$ such that

$$W'_j \subsetneq W_j \tag{159}$$

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:

$$U_j = W_h \setminus W_j \tag{160}$$

$$W'_h = W'_j \cup U_j \tag{161}$$

From (160) we have:

$$U_j \cap W_j = \emptyset \tag{162}$$

From (162) and (159) we have:

$$U_j \cap W'_j = \emptyset \tag{163}$$

From (159) - (163) we have:

$$W'_h \subsetneq W_h \tag{164}$$

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 $\Pi_h$. 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 $\Pi_h$ from which $r$ was obtained during the computation of $\Pi_h^{W_h}$. We consider two possible cases.

i. $r'$ is a rule of $\Pi_j$. 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 $\Pi_j$. 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

$$W_h \text{ satisfies the body of } r' \tag{165}$$

Let us denote the head of $r'$ by $l_0$. Since $W_h$ is a possible world of $\Pi_h$, from (165) we have

$$W_h \text{ satisfies } l_0 \tag{166}$$

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 $random(rn, a_q, p)$ 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 $random(rn, a_r, p)$, 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 $random(rn, a_s, p)$ 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 $\Pi$ 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 $\Pi_0$ 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 165.
- We show that the level of $body(r')$ in $\Pi$ 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 $\Pi_j$, 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

$$l_0 \text{ is either formed by } a_q \text{ or is of the form } random(rn, a_q, p) \tag{167}$$

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 $\Pi_{q-1}$ such that

$$W_h = W_{q-1} \cup U_{q-1}, \tag{168}$$

and

$$U_{q-1} \cap L_{q-1} = \emptyset \tag{169}$$

Since $r'$ does not belong to $\Pi_j$,

$$r' \text{ contains at least one literal which does not belong to } L_j. \tag{170}$$

Since $q \leq j$,

$$L_q \subseteq L_j. \tag{171}$$

From (170) and (171) it follows that

$$r' \text{ contains at least one literal which does not belong to } L_q \tag{172}$$

Since the head of $r'$ is formed by $a_q$, from (172) it follows that

$$\text{the body of } r' \text{ contains a literal which does not belong to } L_q \tag{173}$$

From (173) it follows that

$$W_{q-1} \text{ does not satisfy the body of } r'. \tag{174}$$

Therefore, by clause 1 of Definition 7 from (174) 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 $\Pi_{q-1}$ such that $W_{q-1}$ does not satisfy it. From (168) and (169) 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 (165).
- $r'$ is a general axiom, which is, given that it's head is either $random(rn, a_q, p)$ or $a_q$, must be of the form:

$$a_q = y \leftarrow random(rn, a_q, p), not\ a_q = y_1, \ldots, not\ a_q = y_k \quad (175)$$

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

B. $l_0$ is not a member of $L_j$. In this case, since, by (166), $W_h$ 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 $\Pi_j$ such that

$$W_h = W_j^1 \cup U_j^1 \quad (176)$$

$$W_h = W_j^2 \cup U_j^2 \quad (177)$$

$$U_j^1 \cap L_j = \emptyset \quad (178)$$

$$U_j^2 \cap L_j = \emptyset \quad (179)$$

Since $L_j$ contains all the literals that can be constructed from the signature of $\Pi_j$, from equations (176) and (178) it follows that

$$W_j^1 = W_h|_{L_j} \quad (180)$$

Similarly, from equations (177) and (179) it follows that

$$W_j^2 = W_h|_{L_j} \quad (181)$$

From equations (180) and (181) 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 $\Pi_j$.

**Lemma 12** Let $\Pi$ 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 $\Pi$, $W$ satisfies $\Pi^W$. 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 $\Pi_i$. Let $\Pi'$ 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 $\Pi'$, $W_0 \subseteq W$ and $W \setminus W_0 \cap L_0 = \emptyset$. □

*Proof* Let $L'_0$ be the set of literals from the base $S$ of $\Pi'$, and $W'_0$ be the answer set of $S$. We first show

$$\Omega_{red(\Pi')} = \Omega_{\Pi'} \tag{182}$$

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 $\Pi'_i$ 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 $W'_0$. (182) 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 (182), the lemma holds.

**Lemma 14** Let $i \in \{0..k-1\}$ be an integer and $V$ be a possible world of $\Pi_i$. Let $\Pi'$ 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 $\Pi'$, $W \setminus V$ does not contain literals from $L_i$. □

*Proof* We define $X$ as follows:

$$X = (W \setminus V)|_{L_i} \tag{183}$$

We show:

$$W \setminus X \text{ satisfies the rules of } \Pi_{i+1}^W \tag{184}$$

Let $r$ be a rule of $\Pi_{i+1}^W$. 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 $\Pi_{i+1}^W$, 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 $random(rn, a_j, p)$, for a random attribute term $a_j$, where $j \leq i$. By Lemma 11, there must exists a possible world $V_{j-1}$ of $\Pi_{j-1}$ such that

$$V_{j-1} \subseteq V \tag{185}$$

and

$$(V \setminus V_{j-1}) \cap L_{j-1} = \emptyset \tag{186}$$

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

i.
$$V_{j-1} \text{ satisfies the body of } r' \tag{187}$$
and
$$\text{all the literals occurring in } r' \text{ are in } L_{j-1} \tag{188}$$
Since $j \leq i$, $L_{j-1} \subseteq L_i$, we have
$$r' \text{ belongs to } \Pi_i \tag{189}$$
From (185), (186), (187) and (188) we have that $V$ satisfies the body of $r'$.
Since $V$ is a possible world of $\Pi_i$ from (189) 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 $\Pi^W_{i+1}$ is obtained from $r'$.

iii. $r'$ is a general axiom of the form
$$a_j = y \leftarrow random(rn, a_j, p), not\ a_j = y_1, \ldots, not\ a_j = y_k$$
and $r$ is of the form
$$a_j = y \leftarrow random(rn, a_j, p)$$
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
$$random(rn, a_j, p) \in V \tag{190}$$
Since $V \subseteq W$, $random(rn, a_j, p) \in W$. Since $r$ belongs to the reduct $\Pi^W_{i+1}$, $\{a_j = y_1, \ldots, a_j = y_k\} \cap W = \emptyset$. Since $V \subseteq W$,
$$\{a_j = y_1, \ldots, a_j = y_k\} \cap V = \emptyset \tag{191}$$
Since $r' \in \Pi_i$ (all the literals are clearly in $L_j \subseteq L_i$), and $V$ is a possible world of $\Pi_i$, from (190) and (191) 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 $\Pi$ is $\leq i$. Let $r'$ be the rule of $\Pi^W_{i+1}$ 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:

$$W_0 \subseteq W \tag{192}$$

$$(W \setminus W_0) \cap L_0 = \emptyset \tag{193}$$

From (193) 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 $\Pi^W_{i+1}$, $W$ 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 $\Pi_i$. Since $V$ is a possible world of $\Pi_i$, 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 $\Pi'$ 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 $\Pi^W_{i+1}$. 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 $\Pi_{i+1} \cup V \cup \{\leftarrow \mathit{not}\ a_{i+1} = y\}$. 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 $\Pi^W_{i+1}$. Since $V \subseteq W \setminus X$, $W \setminus X$ satisfies $V$. Since $W$ is a possible world of $\Pi_{i+1} \cup V \cup \{\leftarrow \mathit{not}\ a_{i+1} = y\}$, $W$ contains $a_{i+1} = y$. Therefore, $(\Pi_{i+1} \cup V \cup \{\leftarrow \mathit{not}\ a_{i+1} = y\})^W = (\Pi_{i+1} \cup V)^W$.Therefore, $(W \setminus X) \subsetneq W$ satisfies all the rules of $(\Pi_{i+1} \cup V \cup \{\leftarrow \mathit{not}\ a_{i+1} = y\})^W$, which contradicts the fact that $W$ is a possible world of $(\Pi_{i+1} \cup V \cup \{\leftarrow \mathit{not}\ a_{i+1} = y\})$.
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 $\Pi^W_{i+1}$. 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 $\Pi_i$.

1. if $V$ falsifies the bodies of all random selection rules with $a_{i+1}$ in the head, then every possible world of $V \cup \Pi_{i+1}$ is a possible world of $\Pi_{i+1}$.

2. if there is a random selection rule of the form

$$random(rn, a_{i+1}, p) \leftarrow B \tag{194}$$

s.t $V$ satisfies $B$, 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 $\Pi_{i+1}$.

□

*Proof* Let $\Pi_{ext}$ 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 $\Pi_{ext}$. We show that $W$ is a possible world of $\Pi_{i+1}$. We first show that $W$ satisfies the rules of $\Pi^W_{i+1}$, and then we show that $W$ is minimal.

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

$$W' \text{ satisfies } V^W \tag{195}$$

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 $\Pi$ such that

$$a = y \in V \tag{196}$$

and

$$a = y \notin W' \tag{197}$$

Let us define $V'$ to be:

$$V' = W'|_{L_i} \tag{198}$$

From (197) and (198) we have:

$$a = y \notin V' \tag{199}$$

By lemma 14 we have

$$W \setminus V \text{ does not contain literals from } L_i \tag{200}$$

Therefore, since $V$ is a possible world of $\Pi_i$, we have:

$$V = W|_{L_i} \tag{201}$$

Since $W' \subsetneq W$, from (201) and (198) we have:

$$V' \subseteq V \tag{202}$$

From (202), (196) and (199) we have:

$$V' \subsetneq V \tag{203}$$

We next show

$$V' \text{ satisfies the rules of } \Pi^V \tag{204}$$

From (200) we have:

$$\Pi_i^V \subseteq \Pi_{i+1}^W \tag{205}$$

Let $r$ be a member of $\Pi_i^V$ such that $V'$ satisfies the body of $r$. We show that $V'$ satisfies the head of $r$. From (198) we have that the body of $r$ is satisfied by $W'$.
Since $W'$ satisfies the rules of $\Pi_{i+1}^W$, from (205) $W'$ satisfies the head of $r$. Since $r$ is a member of $\Pi_i^V$, $head(r) \in L_i$. Therefore, from (198), $V'$ satisfies $head(r)$. Therefore, (204) holds, and, considering (202), we have a contradiction.

**Lemma 16** For every $i \in \{0, \ldots, k\}$ and every leaf node $n$ of $T_i$ program $\Pi_i$ 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 7 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 $\Pi_{i-1}$ containing $p_{T_{i-1}}(m) \setminus \{true\}$.

(i) First we will show that every possible world $W$ of $\Pi_i$ containing $p_{T_{i-1}}(m)$ also contains $V$. By lemma 11, set $V' = W|_{L_{i-1}}$ is a possible world of $\Pi_{i-1}$. 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 $\Pi$ of the form

$$random(rn, a_i, p) \leftarrow K \tag{206}$$

$V$ falsifies $K$. We will show that in this case $m$ is not ready to branch on $a_i$ w.r.t $\Pi_i$. It is sufficient to show that for every random selection rule of the form (206), $V$ is not $\Pi_i$-compatible with $K$. Since $V$ falsifies $K$, there exists an e-literal $l \in K$ such that:

$$V \text{ does not satisfy } l \tag{207}$$

$$l \in L_i \tag{208}$$

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

$$W \text{ satisfies } V \cup K \tag{209}$$

By Lemma 11 we have:

$$(W \setminus V) \cap L_i = \emptyset \tag{210}$$

From (207) , (208) and (210) we have:

$$W \text{ does not satisfy } l \tag{211}$$

Therefore, since $l \in K$, we have a contradiction to (209). Therefore, $m$ is not ready to branch on $a_i$ w.r.t $\Pi_i$, and, by construction of $T_i$, $m = n$. By condition 2.b.III of Definition 7, we have $V \cup \Pi_{i-1}$ has exactly one possible world, $W$. By lemma 15, $W$ is a possible world of $\Pi_i$. 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 2.b.iii of Definition 7.

(iii) There is a random selection rule $r$ of the form (206) s.t.

$$V \text{ satisfies } K \tag{212}$$

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

Condition (1) of the definition of "ready to branch" (Definition 27) follows immediately from construction of $T_{i-1}$.

To prove Condition (2) we need to show that $p_{T_{i-1}}(m)$ $\Pi_i$-guarantees $K$. Since $r$ is active in $V$, by Condition 2.b.II of Definition 7 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, $W_0$.

From 212, 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,

$$\text{V is } \Pi_i\text{-compatible with } K \tag{213}$$

Now consider a possible world $W$ of $\Pi_i$ 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 $pr(rn, a_i = y \mid B) = v$ from $\Pi_i$ such that $B$ is $\Pi_i$-compatible with $p_{T_{i-1}}(m)$. $\Pi_i$-compatibility implies that there is a possible world $W_0$ of $\Pi_i$ 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 2 of Definition 7 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 $\Pi_i$ 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 $p_{T_{i-1}}(m)$ $\Pi_i$-guarantees $p(y_0)$. Condition (2) of Definition 7 guarantees that $\Pi_i$ 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 $\Pi_i$ compatible. From (i) we have that $p_{T_{i-1}}(m)$ $\Pi_i$-guarantees $p(y_0)$. Similar argument shows that if $p_{T_{i-1}}(m)$ is $\Pi_i$-compatible with $p(y)$ then $p(y)$ is also $\Pi_i$-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 $\Pi_{i+1}$. 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.b.II of Definition 7, program $V \cup \Pi_i \cup \{\leftarrow \mathit{not}\ a_i = y_0\}$ has exactly one possible world, $W$. By lemma 15 we have that $W$ is a possible world of $\Pi_i$. Clearly $W$ contains $p_{T_i}(n)$. Uniqueness follows immediately from (i) and Condition 2.b.II of Definition 7.

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

*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 $\Pi_i$. By Lemma 11, $\Pi_{i-1}$ has a possible world $V$ such that:

$$V \subseteq W \tag{214}$$

$$(W \setminus V) \cap L_{i-1} = \emptyset \tag{215}$$

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

$$\mathit{random}(rn, a_i, p) \leftarrow K \tag{216}$$

$K$ 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

$$\mathit{random}(rn, a_i : \{X : p(X)\}) \leftarrow K \tag{217}$$

such that

$$V \text{ satisfies } K \tag{218}$$

and, therefore

$$\text{every literal from } K \text{ is in } L_{i-1} \tag{219}$$

From (218), (219), (214) and (215) we have:

$$W \text{ satisfies } K \tag{220}$$

From clause 2 of Definition 7 and the fact that $a_i \in L_i$ we have:

$$r \in \Pi_i \tag{221}$$

Since $W$ is a possible world of $\Pi_i$, it must satisfy $r$ together with general axioms from $\Pi_i$:

$$\begin{aligned} a_i = y_1 \mid \ \ldots \ \mid a_i = y_m &\leftarrow \mathit{random}(rn, a_i, p) \\ &\leftarrow a_i = Y, \mathit{not}\ p(Y), \mathit{random}(rn, a_i, p) \end{aligned}$$

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

$$a_i = y \in W \tag{222}$$

and

$$p(y) \in W \tag{223}$$

From 218, by clause 2.b.II of Definition 7 we must have $p(y) \in L_{i-1}$. From (223) and (215) we have:

$$p(y) \in V \tag{224}$$

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 $\Pi_i$. Since $p_{T_{i-1}}(m) \subseteq V \subseteq W$, $p_{T_{i-1}}(m)$ is $\Pi_i$-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 $\Pi_i$-compatible with $B$ iff $p_{T_{i-1}}(n)$ is $\Pi_j$-compatible with $B$.

□

*Proof* →
Suppose that $p_{T_{i-1}}(n)$ is $\Pi_i$-compatible with $B$. This means that there is a possible world $V$ of $\Pi_i$ 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, $\Pi_j$ 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 $\Pi_i$ 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 $\Pi_j$-compatible with $B$.

←
Let $W$ be a possible world of $\Pi_j$ 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 $\Pi_i$ 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 $\Pi_i$-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 $p_{T_{i-1}}(n)$ $\Pi_i$-guarantees $B$ iff $p_{T_{i-1}}(n)$ $\Pi_j$-guarantees $B$. □

*Proof* →
Let us assume that $p_{T_{i-1}}(n)$ $\Pi_i$-guarantees $B$. This implies that $p_{T_{i-1}}(n)$ is $\Pi_i$-compatible with $B$, and hence, by Lemma 18 $p_{T_{i-1}}(n)$ is $\Pi_j$-compatible with $B$. Now let $W$ be a possible world of $\Pi_j$ satisfying $p_{T_i}(n)$. By Lemma 11 $W = V \cup U$ where $V$ is a possible world of $\Pi_i$ and $U \cap L_i = \emptyset$. This implies that $V$ satisfies $p_{T_{i-1}}(n)$. Since $p_{T_{i-1}}(n)$ $\Pi_i$-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 $p_{T_{i-1}}(n)$ $\Pi_j$-guarantees $B$. This implies that $p_{T_{i-1}}(n)$ is $\Pi_i$-compatible with $B$. Now let $V$ be a possible world of $\Pi_i$ containing $p_{T_{i-1}}(n)$. By Lemma 17, there exists a unique leaf node $n'$ of $T_i$ such that

$$V \text{ satisfies } p_{T_i}(n') \tag{225}$$

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

$$W \text{ satisfies } p_{T_i}(n') \tag{226}$$

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

$$V' \text{ satisfies } p_{T_i}(n') \tag{227}$$

From (225) and (227) we by Lemma 16 we have:

$$V = V' \tag{228}$$

Since $p_{T_{i-1}}(n) \setminus \{true\} \subseteq V = V' \subseteq W$, we have $W$ satisfies $p_{T_{i-1}}(n)$. Therefore, since $p_{T_{i-1}}(n)$ $\Pi_j$-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 $p_{T_{i-1}}(n)$ $\Pi_i$-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 $\Pi_i$ iff $n$ is ready to branch on $a_i$ relative to $\Pi_j$. □

*Proof* $\rightarrow$
Suppose $n$ is ready to branch on $a_i$ via rule $r$

$$random(rn, a_i, p) \leftarrow K \tag{229}$$

relative to $\Pi_i$. We show that $n$ is ready to branch on $a_i$ via $r$ relative to $\Pi_j$. 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 $\Pi_i$.
2. We prove condition 2:

$$p_{T_{i-1}}(n) \ \Pi_j\text{-guarantees } K \tag{230}$$

By Lemma 16, there is a unique possible world $W_{i-1}$ of $\Pi_{i-1}$ such that

$$W_{i-1} \text{ satisfies } p_{T_{i-1}}(n) \tag{231}$$

We prove that

$$W_{i-1} \text{ satisfies } K \tag{232}$$

We prove by contradiction. Suppose 232 doesn't hold. By condition 2.b.I of Definition 7 we have $W_{i-1}$ falsifies $K$. That is, there is a literal $l \in L_{i-1} \cap K$ such that $W_{i-1}$ does not satisfy $l$. Then, by conditions 2.b.ii and 2.b.iii of

Definition 7 and Lemma 15 we have that $\Pi_i$ 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 $\Pi_i$. Therefore, (232) holds. Therefore, the literals occurring in the body of $r$ are from $L_{i-1}$, and by lemma (19) we conclude (230).

3. We prove condition 3. Let $pr(rn, a_i = y \mid B) = v$ be a pr-atom from $\Pi_j$. We show that

$$p_{T_{i-1}}(n) \text{ either } \Pi_j\text{-guarantees } B \text{ or is } \Pi_j\text{-incompatible with } B \tag{233}$$

Since $n$ is ready to branch on $a_i$ via rule $r$ relative to $\Pi_i$, we have 3 cases:

(a) $pr(rn, a_i = y \mid B) = v$ is a pr-atom from $\Pi_i$, and $B$ is $\Pi_i$-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) $pr(rn, a_i = y \mid B) = v$ is a pr-atom from $\Pi_i$, and $B$ is $\Pi_i$-incompatible with $p_{T_{i-1}}(n)$. That is,

$$\text{every possible world } W_i \text{ or } \Pi_i \text{ satisfying } p_{T_{i-1}}(n) \text{ does not satisfy } B \tag{234}$$

By Lemma 16, there is a unique possible world $W_{i-1}$ of $\Pi_{i-1}$ such that

$$W_{i-1} \text{ satisfies } p_{T_{i-1}}(n) \tag{235}$$

We prove

$$W_{i-1} \text{ falsifies } B \tag{236}$$

For the sake of contradiction, suppose (236) is false. By clause 2.b.I of Definition 7 we have:

$$W_{i-1} \text{ satisfies } B \tag{237}$$

By clauses 2.b.II and 2.b.III of 7 and Lemma 15 we have that $\Pi_i$ has a possible world $W_i$ containing $W_{i-1}$. Since $B$ is in $L_i$, by Lemma 11 we have

$$W_i \text{ satisfies } B \tag{238}$$

Since $p_{T_{i-1}}(n) \setminus \{true\} \subseteq W_{i-1} \subseteq W_i$, we have a contradiction from (238) and (234).

Therefore, (236) holds. Now let $W_j$ be a possible world of $\Pi_j$ satisfying $p_{T_{i-1}}(n)$. By Lemma 11, there is a possible $W'_{i-1}$ of $\Pi_{i-1}$ such that $W'_{i-1} \subseteq W_j$ and

$$(W'_{i-1} \cap W_j) \cap L_{i-1} = \emptyset \tag{239}$$

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

$$W'_{i-1} = W_{i-1} \tag{240}$$

From (240), (236), (239) we have that $W_j$ does not satisfy $B$. Therefore, $p_{T_{i-1}}(n)$ is $\Pi_j$-incompatible with $B$

(c) $pr(rn, a_i = y \mid B) = v$ does not belong to $\Pi_i$. That is,
$B$ contains an e-literal $l$

$$l \notin L_i \tag{241}$$

By Lemma 16, there is a unique possible world $W_{i-1}$ of $\Pi_{i-1}$ such that

$$W_{i-1} \text{ satisfies } p_{T_{i-1}}(n) \tag{242}$$

Since $B$ has $l$ s.t. (241), $W_{i-1}$ cannot satisfy $B$. Therefore by condition 2.b.I of Definition 7,

$$W_{i-1} \text{ falsifies } B \tag{243}$$

Similarly to 2, given (243), 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 $\Pi_j$-incompatible with $B$

4. We prove condition 4. By Lemma 16, there is a unique possible world $W_{i-1}$ of $\Pi_{i-1}$ such that:

$$W_{i-1} \text{ satisfies } p_{T_{i-1}}(n) \tag{244}$$

As in 1, we can show that $r$ is active in $W_{i-1}$. Therefore, by condition 2.b.II of Definition 7, 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 $\Pi_i$ and lemmas (18), (19).

$\leftarrow$

Now suppose $n$ is ready to branch on $a_i$ via rule $r$

$$random(rn, a_i : \{X : p(X)\}) \leftarrow K \tag{245}$$

relative to $\Pi_j$. We show that $n$ is ready to branch on $a_i$ via $r$ relative to $\Pi_i$. 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 $\Pi_i$.
2. We prove Condition 2:

$$p_{T_{i-1}}(n) \ \Pi_i\text{-guarantees } K \tag{246}$$

By Lemma 16, there is a unique possible world $W_{i-1}$ of $\Pi_{i-1}$ such that

$$W_{i-1} \text{ satisfies } p_{T_{i-1}}(n) \tag{247}$$

We prove that

$$W_{i-1} \text{ satisfies } K \tag{248}$$

We prove by contradiction. Suppose $W_{i-1}$ does not satisfy $K$. By condition 2.b.I, $W_{i-1}$ 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.b.II, 2.b.III of Definition 7 and Lemma 15 we have that $\Pi_j$ 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 $\Pi_j$. Therefore, (248) holds. Therefore, the literals occurring in the body of $r$ are from $L_{i-1}$, and by Lemma (19) we conclude (246).

3. We prove condition 3. Let $pr(rn, a_i = y \mid B) = v$ be a pr-atom from $\Pi_i$. We show that

$$p_{T_{i-1}}(n) \text{ either } \Pi_i\text{-guarantees } B \text{ or is } \Pi_i\text{-incompatible with } B \tag{249}$$

Since $n$ is ready to branch on $a_i$ via rule $r$ relative to $\Pi_j$, we have 2 cases:

(a) $B$ is $\Pi_j$-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 $p_{T_{i-1}}(n)$ $\Pi_j$-guarantees $B$

(b) $B$ is $\Pi_j$-incompatible with $p_{T_{i-1}}(n)$. That is,

$$\text{every possible world } W_j \text{ or } \Pi_j \text{ satisfying } p_{T_{i-1}}(n) \text{ does not satisfy } B \tag{250}$$

By Lemma 16, there is a unique possible world $W_{i-1}$ of $\Pi_{i-1}$ such that

$$W_{i-1} \text{ satisfies } p_{T_{i-1}}(n) \tag{251}$$

We prove

$$W_{i-1} \text{ falsifies } B \tag{252}$$

For the sake of contradiction, suppose (236) is false. By clause 2.b.I of definition (7) we have:

$$W_{i-1} \text{ satisfies } B \tag{253}$$

By clause 2.b.II of Definition 7 and Lemma 15 we have that $\Pi_j$ has a possible world $W_j$ containing $W_{i-1}$. Since $B$ is in $L_i$, by Lemma 11 we have

$$W_j \text{ satisfies } B \tag{254}$$

Since $p_{T_{i-1}}(n) \setminus \{true\} \subseteq W_{i-1} \subseteq W_i$, we have a contradiction from (254) and (250).

Therefore, (252) holds. Now let $W_i$ be a possible world of $\Pi_i$ satisfying $p_{T_{i-1}}(n)$. By Lemma 11, there is a possible $W'_{i-1}$ of $\Pi_{i-1}$ such that $W'_{i-1} \subseteq W_j$ and

$$(W'_{i-1} \cap W_i) \cap L_{i-1} = \emptyset \tag{255}$$

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

$$W'_{i-1} = W_{i-1} \tag{256}$$

From (256), (252), (239) we have that $W_i$ does not satisfy $B$. Therefore, $p_{T_{i-1}}(n)$ is $\Pi_i$-incompatible with $B$

4. As in 1, we can show that the body of $r$ is satisfied by $W_{i-1}$. Therefore, by condition 2.b.II of 7, 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 $\Pi_j$ 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 $\Pi$'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 $\Pi$. 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 $\Pi$-guaranteed by $p_T(n)$, and hence $W$ is represented by $n$. Suppose now that $n'$ is a node of $T$ representing $W$. Then $p_T(n')$ $\Pi$-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 $\Pi$ is represented by exactly one leaf node of $T$.

Now let $n$ be a leaf node of $T$. By Lemma 16 $\Pi$ 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 $\Pi$. If $n$ is a node of $T$ which is ready to branch on $a(\bar{t})$ via $r$, then all possible worlds of $\Pi$ 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

$$v_T(R(W)) = \mu(W). \tag{257}$$

By definition of $\mu$, we have:

$$\mu(W) = \frac{\hat{\mu}(W)}{\sum_{W_i \in \Omega(\Pi \setminus U)} \hat{\mu}(W_i)} \tag{258}$$

where

$$\hat{\mu}(W) = \prod_{W(a)=y} P(W, a = y) \tag{259}$$

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 $\Pi \setminus U$

$$v_T(R(W)) = \hat{\mu}(W). \tag{260}$$

To prove 260, 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 $\Pi$ of the form

$$random(rn, a, p) \leftarrow K \tag{261}$$

such that $W$ satisfies $K$, $truly\_random(rn, a)$ 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 (261) of $\Pi$. 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 261 it follows that $W$ satisfies $random(rn, a, p)$, and, since $\Pi \setminus U$ does not contain activity records, $W$ satisfies $truly\_random(rn, a)$.
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

$$W \text{ satisfies } p_T(R(W)) \tag{262}$$

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 262 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 program from class $\mathcal{B}$ is coherent.

*Proof* Suppose $\Pi$ belongs to class $\mathcal{B}$ ordered and $U$ be the set of activity records of $\Pi$. 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 $\Pi$ is probabilistically consistent. Since $\Pi$ belongs to $\mathcal{B}$, $\Pi$ is logically consistent. Therefore, by Definition 2, $\Pi$ is coherent.