Towards a Complete Formalization of PLN
(DRAFT)

Nil Geisweiller

July 15, 2025

1 Introduction

The goal is similar to Solomonoff Universal Induction [**?**], that is we want to approach a first order (unknown but computable) distribution μ given observations, using a second order (uncomputable but known) distribution ν, called the Universal Distribution. In Solomonoff Induction, observations are bit strings produced by a Turing machine[1]. In PLN however, observations are outcomes from an indexed boolean random variable, representing the outputs of evaluating a predicate on some inputs. Such predicate is called the *observable predicate*. In practice PLN allows multiple observable predicates however one can assume one predicate without loss of generality. Indeed, to emulate multiple predicates, one can introduce an extra component in the predicate's domain to "select" the predicate of interest. Also, since it is observed by a random variable, such predicate is not necessarily deterministic (though it could be). As such, one may think of the observable predicate as being a program drawn from a certain Probabilistic Programming Language. In the following section we formally define the above.

Because this reformulation of PLN departs somewhat from the definition of PLN in the PLN book [**?**], we give it a new name, νPLN.

2 Definitions

In its original form, PLN purposely avoids relying on an underlying global probability distribution. I am not against this in principle. I will simply admit that I cannot conceive a complete definition of PLN that does not rely on such global probability distribution. I would also point out that a publication released after the PLN book by the principal authors of the PLN book, Ben Goertzel and Matt Ikle, very much aligns with the idea of a global probability distribution [**?**], and was in fact a great source of inspiration for writing this very document. The

[1]Note that even though the sample space of ν is made of deterministic Turing machines, ν can approximate any computable distribution μ (thus non-deterministic) by maintaining an ensemble of such Turing machines.

next subsection is dedicated to define the global probability distribution which νPLN is intended to derive from.

2.1 Global Probability Distribution

Let $(\Omega, \mathcal{F}, \nu)$ be a probability space such that

- $\mathcal{F}$ is the event space, a σ-algebra on Ω.
- $\nu : \mathcal{F} \to [0, 1]$ is a universal distribution, further defined below.
- Ω, the set of possible worlds, is the sample space associated to ν, such that each element $\hat{\omega} \in \Omega$ contains
  1. a probabilistic predicate $\hat{\mu} \in \mathcal{L}$ over a domain $\mathcal{D}$, described in a probabilistic programming language $\mathcal{L}$,
  2. a mapping of $\hat{\mu}$ from $\mathcal{D}$ to Boolean. For instance if $\mathcal{D}$ is $\mathbb{N}$, then a possible mapping could be $(\hat{\mu}\ 0) = \texttt{True}$, $(\hat{\mu}\ 1) = \texttt{False}$, $(\hat{\mu}\ 2) = \texttt{True}, \ldots$, corresponding so far to a predicate indicating the evenness of a natural number. Note that the world $\hat{\omega}$ contains the entire, potentially infinite, mapping, even though in reality an observer can only have access to a finite subset of it.

An observer lives in a particular world $\omega \in \Omega$, called the *true world*, which includes the *true generator* or *true predicate*, μ, and the *true history*, which is the complete mapping of evaluations of μ over the domain $\mathcal{D}$. Note that since μ is probabilistic, it does not deterministically determine the history, thus the true history is just one possible history among an infinity of histories compatible with μ. Note that throughout the document ω and μ refer to the true world and true generator respectively, which is to be contrasted with $\hat{\omega}$ and $\hat{\mu}$ which refer to arbitrary elements of Ω and $\mathcal{L}$ respectively. Although when it is clear that the definitions are generic and apply to any arbitrary world and the true world alike, we will use ω and μ as well. Since μ is a probabilistic predicate, its type signature cannot merely be

$$\mu : \mathcal{D} \to \text{Bool}$$

where Bool $= \{\texttt{False}, \texttt{True}\}$. To capture its probabilistic nature we give it the following type signature

$$\mu : \mathcal{D} \to \Omega \to \text{Bool}$$

in a curried fashion. Meaning that given its argument, it produces a boolean random variable. Therefore the observable predicate can be viewed as an indexed boolean random variable. Of course, μ never gets to be evaluated on a different world than the one it belongs to, thus we can write $(\mu\ a)$ while meaning $(\mu\ a\ \omega)$, but we still need to keep the Ω argument in order to reason about possible worlds since the true world is unknown. Also, in cases where the domain

$\mathcal{D}$ can be decomposed into multiple components, a curried notation will be used interchangeably. So instead of

$$\mu : (\mathcal{D}_1 \times \cdots \times \mathcal{D}_n) \to \Omega \to \text{Bool}$$

the following

$$\mu : \mathcal{D}_1 \to ... \to \mathcal{D}_n \to \Omega \to \text{Bool}$$

will be used. This will be convenient to express inheritance relationships between partially applied predicates. As already hinted, the application of μ to an input x is denoted with the traditional functional programming style

$$(\mu\ x)$$

Thus if the domain is decomposed into subdomains $\mathcal{D}_1$ to $\mathcal{D}_n$, applying μ to all its inputs x_1 to x_n will be denoted

$$(\mu\ x_1\ ...\ x_n)$$

Likewise for the ω argument

$$(\mu\ x_1\ ...\ x_n\ \omega)$$

Such functional programming notation is used for μ because it is currying-friendly. For the rest, we keep using the traditional mathematical function application style, such as

$$\nu(E)$$

denoting the application of the probability distribution ν to the event E. In case μ is known to be deterministic, Ω could potentially be dropped, but that is not going to be our working assumption for now. Additionally to the functional programming style, we may use parametric notation, so for instance instead of

$$(\mu\ x_1\ x_2)$$

we may write

$$(\mu_{x_1}\ x_2)$$

where x_1 is viewed as a parameter and x_2 is viewed as the argument of μ_{x_1}. With that, let us now define key random variables to access Ω:

- $M : \Omega \to \mathcal{L}$ with measurable space $(\mathcal{L}, \mathcal{F}_{\mathcal{L}})$, where $\mathcal{L}$ is a certain probabilistic programming language and $\mathcal{F}_{\mathcal{L}}$ is a σ-algebra on $\mathcal{L}$. Thus, M takes a world $\hat{\omega} \in \Omega$ and outputs the probabilistic program $\hat{\mu} \in \mathcal{L}$ generating that world. Note that this random variable is inaccessible from an observer within that world. An observer within that world only has access to a finite record of evaluations of $\hat{\mu}$. However, this random variable is important to reason about multiple worlds, we suspect it is particularly important for higher order reasoning.

- $D_{x \in \mathcal{D}} : \Omega \to \text{Bool}$, a Boolean random variable indexed by values in $\mathcal{D}$. Unlike M, $D_{x \in \mathcal{D}}$ is at least partially accessible from an observer within that world. Meaning, such observer can gather data for a finite subset $\mathcal{S}$ of $\mathcal{D}$. In this case $D_{\mathcal{S}}$ represents a finite family of Boolean random variables, corresponding the set of accessible observations. M and $D_{x \in \mathcal{D}}$ are related by the following equality

  $$(\hat{\mu}\ x\ \hat{\omega}) = (D_x\ \hat{\omega})$$

  where $\hat{\omega} \in \Omega$ such that $(M\ \hat{\omega}) = \hat{\mu}$. Or simply, in curried fashion

  $$(\hat{\mu}\ x) = D_x$$

Due to the equality above, the distribution of observations is entirely determined by a model $\hat{\mu}$. In other words, it suffices to define a distribution over $\mathcal{L}$, the prior distribution over possible models, to define ν (as far as M and D_x are concerned anyway). Then, relating observations to models can be done using regular Bayesian inference. The prior is defined by

$$\nu(M \in L)$$

where $L \in \mathcal{F}_{\mathcal{L}}$. Note how it is expressed in terms of elements of $\mathcal{F}_{\mathcal{L}}$, instead of elements of $\mathcal{L}$. It is because, for the purpose of recovering PLN with the Bayesian approach, $\mathcal{L}$ needs to be continuous. I will explain this in detail, but for now let us use this notation to formulate Bayes' theorem

$$\nu(M \in L | D_{\mathcal{S}}) = \frac{\nu(M \in L) \times \nu(D_{\mathcal{S}} | M \in L)}{\nu(D_{\mathcal{S}})}$$

where $\mathcal{S}$ is a finite subset of $\mathcal{D}$. For the sake of simplicity, we may drop M since it is the only random variable that ranges over $\mathcal{L}$, and simply write

$$\nu(L | D_{\mathcal{S}}) = \frac{\nu(L) \times \nu(D_{\mathcal{S}} | L)}{\nu(D_{\mathcal{S}})}$$

An example of prior will be given in **??**, but in general it can be viewed as a parameter of νPLN. That is, given a certain prior of ν over $\mathcal{L}$, one can derive a certain flavor of νPLN. Since $(\mu\ x) = D_x$, a history data point will be represented as $(\mu\ x) = \texttt{True}$ or $(\mu\ x) = \texttt{False}$, and the usage of the random variable D_x, and especially $D_{\mathcal{S}}$, will be reserved in the formulation of ν to refer to the observed history.

2.2 PLN Global Distribution vs Solomonoff Induction

Note that unlike with Solomonoff Universal Induction, $\hat{\mu}$ represents a probabilistic predicate rather than a computable probability function calculating the probability of any event, although the latter can be derived from the former.

The *true distribution* in Solomonoff Induction corresponds to the *real predicate* here. We prefer to use the word *real* when denoting objective reality, rather than *true* because *true predicate* could be understood as a predicate that always outputs true.

3 PLN and the Global Distribution

In Section 4 will proceed to derive PLN rules from the global probability distribution introduced in Section 2.1, but for now let us recall important notions of PLN and how to relate them to the global distribution defined in Section 2.1.

3.1 Concepts vs Predicates

The PLN book describes respectively the notions of *concepts* and *predicates*, and their respective associated relationships *inheritance* and *implication*. As explain in Section 2.6 *Higher-Order Logical Relationships* of the PLN book, there is a perfect isomorphism between concepts and inheritance on one side and predicates and implication on the other side. Concerned with conciseness, we will pick a side, the predicate side, and essentially forget about the other side, the concept side, for the rest of this document. But before we do so let us recall what is a concept, the inheritance between two concepts, and the isomorphism between concepts and predicates.

3.1.1 Concepts and Inheritance

A concept is a fuzzy (or, as I prefer to say, probabilistic, for reasons I will explain in Section **??**) set, and the *extensional* inheritance between two concepts is a probabilitized subset relationship. Originally, in the PLN book, the inheritance relationship is defined as an explicit mixture of extensional and intensional inheritances. We will show however that they are in fact both the same thing, the extensional inheritance is a way to approach inheritance solely via *induction*, and intensional inheritance is a way to approach inheritance solely via *abduction*. Any one side, extensional or intensional, is good enough to define the other, so let us pick one, the extensional side, and define inheritance with it. The extensional inheritance between two concepts can be viewed as a probabilitized subset relationship. It allows to express things like "most members of a set are also members of another set". For instance one could express in PLN that 90% of birds fly, with

$$(\texttt{Inheritance Bird Fly}) \overset{\text{m}}{=} 0.9$$

Such knowledge might have been obtained by observing a finite sample of birds and whether or not they fly. Meaning there could be an uncertainty on the 90% itself. To represent such uncertainty PLN uses a second order distribution, in this case a Beta distribution as it is an ideal choice to represent the posterior of the parameter of a Bernouilli distribution given observations. Under this assumption only two numbers are required to determine the parameters, α and β, of the associated Beta distribution. A *truth value* called *simple truth value* was created for this purpose and is thus described by two numbers: a strength (a proxy for a probability estimate) and a confidence over this strength from which the α and β parameters of the Beta distribution can be recovered. For

instance, given a simple truth value one may express that 90% of birds fly with a confidence of 0.99

$$(\texttt{Inheritance Bird Fly}) \stackrel{\text{m}}{=} <0.9, 0.99>$$

where 0.9 is the strength and 0.99 is the confidence. The confidence is a value between 0 and 1 that actually encodes the sample size that was used to obtained the strength. The higher the sample size, the higher the confidence. More information about that will be provided in Section **??** but for now let us just leave it at that as it is enough for what we are concerned with in this Section which is the isomorphism between concepts and predicates.

3.1.2 Isomorphism between Concepts and Predicates

To every concept one can associate a predicate and vice versa. To go from concept to predicate one can use the *indicator function*, and to go from predicate to concept one can use the *satisfying set*. These notions are well known for crisp predicates and sets and thus will not be detailed any further here. The only difference in PLN is that concepts are probabilistic, meaning that a probability (or potentially a second order probability) can be attached to the membership of an element to a concept. Likewise, predicates are probabilistic in the sense that a (second order) probability can be attached to the evaluation of an argument to a predicate. As one would expect the isomorphism also applies between the inheritance relationship on the concept side and the implication relationship on the predicate side. So for instance, on the predicate side one can express the same inheritance relationship between birds and fly as follows

$$(\texttt{Implication IsBird DoesFly}) \stackrel{\text{m}}{=} <0.9, 0.99>$$

where `IsBird` and `DoesFly` have been obtained by taking the indicator functions of `Bird` and `Fly`. As mentioned earlier, we will drop the concept side and only focus on the predicate side for the remaining of the document.

3.2 Relating Predicates to μ

As explained in Section 2.1 there is only one global predicate, μ. To manipulate multiple predicates we can simply introduce a additional parameter in the domain of μ indicating predicates. Let us call this subdomain $\mathcal{P}$ which ranges over symbols representing predicates, in this case μ may have the type signature

$$\mu : \mathcal{P} \to \mathcal{D} \to \Omega \to \text{Bool}$$

So for instance to express that a cat is a mammal, traditionally represented as

$$Cat \Rightarrow Mammal$$

one may use the following parametric notation of μ

$$\mu_{Cat} \Rightarrow \mu_{Mammal}$$

Likewise, to represent the evaluation of the predicate *Cat* over a certain cat instance, cat_{123}, one may use

$$(\mu_{Cat}\ cat_{123})$$

corresponding to the traditional representation

$$(Cat\ cat_{123})$$

where $cat_{123} \in \mathcal{D}$.

3.3 Truth Value as Posterior Distribution

3.4 Statement versus Judgement

Like in NAL, a PLN *statement* designates a logical statement without truth value, such as

$$P \Rightarrow Q$$

While a PLN *judgment* designates a PLN statement with a truth value attached to it, such as

$$P \Rightarrow Q \ \stackrel{m}{=} <0.9, 0.8>$$

4 Deriving PLN Rules

Our goal here is to derive every PLN rules in the PLN book from the global distribution that has been defined in Section 2.1. By doing so we hope not only to provide a clear unambiguous definition for each rule, but also an ideal to approach as using a global distribution should give us the means to derive a convergence theorem a la Solomonoff.

4.1 Predicate Direct Introduction

This rule is meant to calculate the truth value corresponding to a Predicate from direct observations. Its truth value can be understood as the marginal probability of a predicate to be true, irrespective of the inputs. It is a subcase of the Implication Direct Introduction presented in Section 4.2 where the implicant is the *Universal Predicate* (a predicate that is always true), but is described here as its own rule to simplify the presentation. Indeed, it should be easier to understand the Implication Direct Introduction rule after understanding the Predicate Direct Introduction rule.

To derive the predicate direct introduction rule we consider the posterior probability of a Bernouilli process with parameter p given all available observations of the predicate in question. Let us begin with μ itself. In order to formulate this posterior we need to assume that $\mathcal{L}$ has a Bernouilli sampler in its set of operators. Let $\mathcal{B}$ be the subset of programs of $\mathcal{L}$ consisting of a single Bernouilli call, thus comprised of the following programs

```
(Bernouilli p)
```

where p is the parameter of the Bernouilli distribution ranging over $[0, 1]$. Such probabilistic program, when executed, outputs `True` with a probability of p or `False` with a probability of 1 - p. Let $\mathcal{F}_{\mathcal{B}}$ be a Borel σ-algebra on $\mathcal{B}$. Let us assume that $\mathcal{F}_{\mathcal{B}}$ is a subfield of $\mathcal{F}_{\mathcal{L}}$. We claim that that if μ is restricted to the programs in $\mathcal{B}$ then the posterior of p corresponds to the truth value of μ, expressed as follows

$$\nu(L_{\mathcal{B}}|D_{\mathcal{S}}) = \frac{\nu(L_{\mathcal{B}}) \times \nu(D_{\mathcal{S}}|L_{\mathcal{B}})}{\nu(D_{\mathcal{S}})}$$

where $L_{\mathcal{B}} \in \mathcal{F}_{\mathcal{B}}$. NEXT

4.2 Implication Direct Introduction

This rule is not explicitly stated as such in the PLN book but can be derived from iteratively applying induction and revision, and reflects the formula of extensional inheritance/implication given in Section 2.4.1 *The Semantics of Inheritance* of the PLN book, at least the strength part. To obtain the confidence part we assume that the second order distribution is a Beta distribution, like in the Section 4.2 *From Imprecise Probabilities to Indefinite Probabilities* of the PLN book. In order to derive a Beta distribution as second order distribution we can assume an underlying Bernouilli process with parameter p with a Beta distribution as prior. The Jeffreys prior where $\alpha = \beta = \frac{1}{2}$ is often the default choice. Given the PLN statement

$$P \Rightarrow Q$$

Let us express its truth value as the posterior probability of the parameter p of the underlying Bernouilli process. In order to map a Bernouilli process onto an implication we zoom-in to the data points where P is true.

NEXT

5 conclusion

References