Temporal Probabilistic Logic Networks for Procedural Reasoning

Nil Geisweiller and Hedra Yusuf

SingularityNET Foundation, The Netherlands
{nil,hedra}@singularitynet.io

Abstract. TODO

Keywords: Temporal Reasoning · Procedural Reasoning · Probabilistic Logic Networks · OpenCog

1 Introduction

The goal of this project is to make an agent as rational as possible, not necessarily as efficient as possible. This stems from the concern that in order to autonomously gain efficiency the agent must first be able to make the best possible decisions, starting first in the outer world, and then in the inner world.

The paper presents

The agent starts in a completely unknown environment

The idea is that reasoning is used at all levels, discovering patterns from raw observations, building plans and making decisions.

It is a work in progress.

Neural networks are excellent at interpolation, but are rather poor at extrapolation, what we need for true intelligence is a system that thinks critically.

Rarely do causes and effects take place over arbitrary temporal scales. For instance it is unlikely to find a cause that may produce the same effect, or an effect at all, after 1ms, 1 century or any time in between. For that reason we focus on a real time temporal logic.

2 Related Work

Event Calculus. Temporal Logic (Next operator). Subjective logic and evidence-based subjective logic.

3 Contributions

The contributions of that paper are:

1. Build upon existing temporal reasoning framework defined in Chap.14 [TODO: cite PLN book].
2. Design an architecture for controlling an agent based on that temporal reasoning extension.

4 Outline

1. Temporal reasoning

5 Probabilistic Logic Networks Recall

PLN stands for *Probabilistic Logic Networks* [1]. It is a mixture of predicate and term logic that has been probabilitized to handle uncertainty. It has at least two types of rules for inferring relationships

1. from direct observations,
2. from existing relationships.

As such it is well suited for building a model of a given environment, and planning in that environment. All it needs is to be properly equipped with a vocabulary for representing temporal and procedural knowledge. The Chapter 14 of the PLN book [1] does just that. However we found that definitions are sometimes ambiguous or incorrect. That paper has been written to remedy that.

5.1 Elementary Notions

Graphically speaking, PLN statements are sub-hypergraphs[1] made of *Links* and *Nodes*, called *Atoms*, decorated with *Truth Values*. Syntactically speaking, PLN statements are not very different from statements expressed in another logic, except that they are usually formatted in prefixed-operator with a tree-style indentation to emphasize their graphical nature and to leave room for their truth values. For instance

$$\begin{array}{l} \mathit{Implication}\ \langle TV \rangle \\ \quad P \\ \quad Q \end{array}$$

represents an implication link between P and Q with truth value TV. For sake of conciseness we also adopt the following representations. We adopt a flattened, as opposed to a tree-style, representation. For instance the implication link above is represented as

$$\mathit{Implication}(P, Q)\ \langle TV \rangle$$

We adopt a more mathematically looking symbolic representation. For instance that same implication can be represented as

$$P \rightarrow Q \stackrel{\text{m}}{=} TV$$

There is a large variety of constructs for PLN. Here, we will focus primarily on the ones for handling predicates. Let us recall that predicates are functions that output Boolean values. The domain of a predicate can be arbitrarily defined, but its range is always Boolean. In that paper, the capital letters A, B, C represent

[1] because links can point to links, not just nodes

atoms of any type, X, Y, Z represent atoms that are variables, while P, Q, R, ... represent atoms that are predicates, typed as follows:

$$P, Q, R, ... : Domain \mapsto \{True, False\}$$

Note that in PLN, predicates are not necessarily crisp because their outputs can be totally or partially unknown, thus potentially described by probabilities.

Truth values are, in essence, probability distributions of probabilities, also called second order probability distributions. They are often described by two numbers: a strength, s, representing a probability, and a confidence, c, representing the confidence over that probability. Such truth values are called *Simple Truth Values* and are denoted as follows

$$< s, c >$$

Alternatively, the strength and the confidence of a simple truth value TV can be denoted $TV.s$ and $TV.c$ respectively. Underneath, a simple truth value corresponds to a beta distribution [?] with parameters $\alpha(s, c) = \alpha_0 + \frac{s.c.k}{1-c}$ and $\beta(s, c) = \beta_0 + \frac{(1-s).c.k}{1-c}$, where k is a PLN parameter called the *Lookahead*, and α_0 and β_0 are usually set to 0.5 corresponding to Jeffreys prior. For truth values obtained from direct evidence, a simple truth value makes perfect theoretical sense [?]. For truth values obtained from indirect evidence, not so much, even though they are often used in practice. When more precision is needed, to represent a multi-modal truth value for instance, a mixture of simple truth values can be used. Through out the paper, sometimes we may say *probability*, while what we really mean is *second order probability distribution*.

Below is a table of the constructs used in that paper with their flattened and symbolic representations, as well as precedence to minimize parenthesis usage when using the symbolic representation.

Flattened	Symbolic	Precedence
$Evaluation(P, A)$	$P(A)$	0
$Not(P)$	$\neg P$	1
$And(P, Q)$	$P \wedge Q$	2
$Or(P, Q)$	$P \vee Q$	2
$Implication(P, Q)$	$P \rightarrow Q$	4
$A\langle TV \rangle$	$A \stackrel{\text{m}}{=} TV$	10

For representing n-ary predicates one may simply use $P(A_1, ..., A_n)$ which may be understood as unary predicates applied on tuples. Let us now explain their semantics and how to interpret their truth values.

– $\neg P$ is the predicate resulting from the pointwise negation of P.

- $P \wedge Q$ is the predicate resulting from the pointwise conjunction of P and Q.
- $P \vee Q$ is the predicate resulting from the pointwise disjunction of P and Q.
- $P(A) \stackrel{\text{m}}{=} TV$ states that $P(A)$ outputs *True* with a second order probability described by TV.
- $P \rightarrow Q \stackrel{\text{m}}{=} TV$ states that if $P(A)$ is *True* for some A in the domain of P, then $Q(A)$ is *True* with a second order probability described by TV. In simple probability terms, it represents $\mathcal{P}r(Q|P)$, the conditional probability of Q knowing P, referring to their satisfying sets to be precise.
- $P \stackrel{\text{m}}{=} TV$ states that the prevalence of P being *True* has the second order probability described by TV.

5.2 Inference Rules

Inferences rules are used to construct PLN statements and calculate their truth values. They fall into two groups, direct evidence based or otherwise. Rules from the former group infer abstract knowledge from direct evidence, while rules from the latter infer knowledge by combining existing abstractions. There are dozens of inference rules, for now we only recall two, *Implication Direct Introduction* and *Deduction*.

The Implication Direct Introduction Rule (IDI) takes evaluations as premises and produces an implication as conclusion, formally depicted by the following proof tree

$$\frac{P(A_1) \stackrel{\text{m}}{=} TV_1^P \quad \ldots \quad P(A_n) \stackrel{\text{m}}{=} TV_n^P}{P \rightarrow Q \stackrel{\text{m}}{=} TV} \text{ (IDI)}$$

Assuming perfectly reliable direct evidence[2] then the resulting simple truth value can be calculated as follows

$$TV.s = \frac{\sum_{i=1}^{n} f_{\wedge}(TV_i^P.s, TV_i^Q.s)}{\sum_{i=1}^{n} TV_i^P.s} \qquad TV.c = \frac{n}{n+k}$$

where $f_{\wedge}$ is a function embodying a probabilistic assumption about the conjunction of the events. Such function typically ranges from the product (perfect independence) to the min (perfect overlap). Note that this inference rule takes an arbitrary number of premises. In practice it is not a problem as it is decomposed into two rules covering the base and the recursive cases, while storing evidence to avoid double counting.

[2] Perfectly reliable evidence have a confidence of 1. Dealing with unreliable evidence involves using convolution products and is outside of the scope of this paper.

The Deduction Rule (D) takes two implications as premises and produces a third one. Depending on the assumptions made there exists different variations of that rule. The simplest one is based on the Markov property

$$\mathcal{P}r(R|Q,P) = \mathcal{P}r(R|Q)$$

which gives rise to the rule depicted by the following proof tree

$$\frac{P \to Q \stackrel{\text{m}}{=} TV^{PQ} \qquad Q \to R \stackrel{\text{m}}{=} TV^{QR} \qquad P \stackrel{\text{m}}{=} TV^{P} \qquad Q \stackrel{\text{m}}{=} TV^{Q} \qquad R \stackrel{\text{m}}{=} TV^{R}}{P \to R \stackrel{\text{m}}{=} TV} \text{ (D)}$$

The reader may notice that three additional premises have been added, corresponding to the probabilities $\mathcal{P}r(P)$, $\mathcal{P}r(Q)$ and $\mathcal{P}r(R)$. This is a consequence of the Markov property. The exact formula for that variation will not be recalled here but it merely derives from

$$\mathcal{P}r(R|P) = \mathcal{P}r(R|Q,P) \times \mathcal{P}r(Q|P) + \mathcal{P}r(R|\neg Q,P) \times \mathcal{P}r(\neg Q|P)$$

More information can be found in Chapter X of [**?**].

6 Temporal Logic

The temporal logic define in [**?**] is somewhat partial and ambiguous. In that section we provide a possible completion. Let us begin by defining *Temporal Predicates*, also called *Fluents* as predicates tends to denote atemporal relationships, as regular predicates with a temporal dimension

$$P, Q, R, ... : Atom^n \times Time \mapsto \{True, False\}$$

Time here is considered discrete, formally defined as a natural number.

6.1 Temporal Operators

Given temporal predicates we can now define a small set of temporal operators.

Lag* and *Lead are temporal operators to shift the time dimension of a temporal predicate. *Lag*, respectively *Lead*, is similar to the metric variation of the *Past* operator denoted P_n, respectively the *Future* operator denoted F_n, of Temporal Logic [**?**, **?**], with the distinction that it applies over an expression that evaluates into a temporal predicate, as opposed to an expression that evaluates into boolean.

The *Lag* operator is formally defined as follows

$$\begin{array}{c} Lag \\ P \\ T \end{array}$$

:=

$$\begin{array}{l} \textit{Lambda} \\ \quad x_1, ..., x_n, t \\ \quad P(x_1, ..., x_n, t - T) \end{array}$$

where the first line of the *Lambda* link is the variable declaration of the temporal predicate and the second line is the body representing a temporally shifted reconstruction of P. Informally speaking, the *Lag* operator gives a peek into the past, or equivalently, brings the past into the present. This is a major difference with [?] that defined sequential and using notions of the Event Calculus [?]. Here we do not do that (TODO: maybe such operator should be called PrecedeAnd, or such). That is because using this simpler notion of sequential and allows more flexibility to construct various forms of descriptions of how events initialization, termination, etc, are sequenced.

The *Lead* operator is the inverse of the *Lag* operator and is formally defined as follows

$$\begin{array}{l} \textit{Lead} \\ \quad P \\ \quad T \end{array}$$

:=

$$\begin{array}{l} \textit{Lambda} \\ \quad x_1, ..., x_n, t \\ \quad P(x_1, ..., x_n, t + T) \end{array}$$

As the inverse of the *Lag* operator, the *Lead* operator gives a peek into the future, or equivalently, brings the future into the present. Finally, as being the inverse of one another, the following equivalence holds

$$\begin{array}{l} \textit{Lag} \\ \quad \textit{Lead} \\ \quad\quad P \\ \quad\quad T \\ \quad T \end{array} \equiv P$$

SequentialAnd is a temporal conjunction where one of the temporal predicate arguments have been temporally shifted. There are two variations one can define. A *BackSequentialAnd* variation where the past of one of the temporal predicates is brought into the present, using the *Lag* operator, formally defined as

$$\begin{array}{l} \textit{BackSequentialAnd} \\ \quad T \\ \quad P \\ \quad Q \end{array} := \begin{array}{l} \textit{And} \\ \quad \textit{Lag} \\ \quad\quad P \\ \quad\quad T \\ \quad Q \end{array}$$

resulting into a temporal predicate such that in order to be true at time t requires that P be true at time t and Q be true at time $t + T$.

Inversely, there is a *ForeSequentialAnd* variation where the future of the other temporal predicate is brought into the present, formally defined as

$$\begin{array}{l} \mathit{ForeSequentialAnd} \\ \quad T \\ \quad P \\ \quad Q \end{array} := \begin{array}{l} \mathit{And} \\ \quad P \\ \quad \mathit{Lead} \\ \quad\quad Q \\ \quad\quad T \end{array}$$

resulting into a temporal predicate such that in order to be true at time t requires that P be true at time $t - T$ and Q be true at time t.

Finally one may notice that *BackSequentialAnd* and *ForeSequentialAnd* equivalent up to temporal shifting, as follows

NEXT

$$\begin{array}{l} \mathit{BackSequentialAnd} \\ \quad T \\ \quad P \\ \quad Q \end{array} \equiv \begin{array}{l} \mathit{BackSequentialAnd} \\ \quad T \\ \quad P \\ \quad Q \end{array}$$

PredictiveImplication likewise defines a temporal implication, formally

$$\begin{array}{l} \mathit{PredictiveImplication} \\ \quad T \\ \quad P \\ \quad Q \end{array}$$

:=

$$\begin{array}{l} \mathit{Implication} \\ \quad P \\ \quad \mathit{Lead} \\ \quad\quad Q \\ \quad\quad T \end{array}$$

resulting into a conditional predicate, that in order to be defined at time t requires that P be true at time t, and in order to be true at t requires that Q be true at $t + T$.

We now have everything we need to define temporal inference rules, but before that let us first introduce some notations in order to be easier to lay out.

6.2 Notations

The following notations can afford to ignore truth values, that is because no new formula is required for temporal reasoning. All that is required are the definitions above mapping temporal expressions into equivalent atemporal ones. The notations are summarized in the table below, ranked by syntactic precedence

to minimize the number of required parenthesis.

Atomese	Notation	Precedence
$Evaluation(P, List(X_1, \ldots, X_n))$	$P(X_1, \ldots, X_n)$	1
$Lambda(t, AtTime(Execution(A), t))$	$\widehat{A}$	1
$Lag(P, T)$	$\vec{P}^T$	1
$Lead(P, T)$	$\overleftarrow{P}^T$	1
$And(P, Q)$	$P \wedge Q$	2
$Or(P, Q)$	$P \vee Q$	2
$SequentialAnd(T, P, Q)$	$P \nearrow^T Q$	3
$SequentialOr(T, P, Q)$	$P \veebar^T Q$	3
$Implication(P, Q)$	$P \to Q$	4
$PredictiveImplication(T, P, Q)$	$P \rightsquigarrow^T Q$	4
$PredictiveImplication(T, P, Q)$	$P \rightsquigarrow^T Q$	4

The precedence of everything else (predicates nodes, etc) is 0.

For instance

TODO: show examples of notational format with the effect of precedence

6.3 Temporal Rules

TODO: *PredictiveImplication* direct introduction.

Given that we can now introduce our temporal rules, (PI), (IP), (S)

TODO: detail (PI), (IP), (S) rules

and the most important one Temporal Deduction (TD)

$$\frac{P \rightsquigarrow^{T_1} Q \qquad Q \rightsquigarrow^{T_2} R \qquad P \qquad Q \qquad R}{P \rightsquigarrow^{T_1+T_2} R} \text{(TD)}$$

To determine the formula to calculate the resulting truth value of such rule, we only need to map such temporal deduction into a regular deduction as follows

$$\dfrac{\dfrac{\dfrac{P \rightsquigarrow^{T_1} Q}{P \to \bar{Q}^{T_1}}\text{(PI)} \qquad \dfrac{\dfrac{Q \rightsquigarrow^{T_2} R}{Q \to \bar{R}^{T_2}}\text{(PI)}}{\bar{Q}^{T_1} \to \bar{R}^{T_1+T_2}}\text{(S)} \qquad P \qquad \dfrac{Q}{\bar{Q}^{T_1}}\text{(S)} \qquad \dfrac{R}{\bar{R}^{T_1+T_2}}\text{(S)}}{P \to \bar{R}^{T_1+T_2}}\text{(D)}}{P \rightsquigarrow^{T_1+T_2} R}\text{(IP)}$$

6.4 Procedural Reasoning

Likewise, we can use the same temporal to regular deduction mapping to build inference rules for procedural reasoning. Given two cognitive schematics

$$P \wedge \widehat{A} \rightsquigarrow^{T_1} Q$$

expressing that executing A in context P likely leads to context Q after T_1 time units, and

$$Q \wedge \widehat{B} \rightsquigarrow^{T_2} R$$

expressing that executing B in context Q likely leads to context R after T_2 time units, the question becomes how to infer the likelihood that executing A and B in sequence starting from context P leads to context R after $T_1 + T_2$ time units, corresponding to the cognitive schematic

$$P \wedge \widehat{A} \nearrow^{T_1} \widehat{B} \rightsquigarrow^{T_1+T_2} R$$

Using the same rules to map *PredictiveImplication* to regular *Implication* and vice versa, as well as rules about the *Lead* operator, we can construct the following inference tree, in fact the PLN engine can construct it for us

$$\dfrac{\dfrac{\dfrac{\dfrac{P \wedge \hat{A} \rightsquigarrow^{T_1} Q}{P \wedge \hat{A} \rightarrow^{T_1} \bar{Q}^{T_1}}\,(\mathrm{PI})}{P \wedge \hat{A} \wedge \bar{\hat{B}}^{T_1} \rightarrow \bar{Q}^{T_1} \wedge \bar{\hat{B}}^{T_1}}\,(\mathrm{C}) \qquad \dfrac{\dfrac{Q \wedge \hat{B} \rightsquigarrow^{T_2} R}{Q \wedge \hat{B} \rightarrow \bar{R}^{T_2}}\,(\mathrm{PI})}{\bar{Q}^{T_1} \wedge \bar{\hat{B}}^{T_1} \rightarrow \bar{R}^{T_1+T_2}}\,(\mathrm{S}) \qquad P \wedge \hat{A} \wedge \bar{\hat{B}}^{T_1} \qquad \dfrac{Q \wedge \hat{B}}{\bar{Q}^{T_1} \wedge \bar{\hat{B}}^{T_1}}\,(\mathrm{S}) \qquad \dfrac{R}{\bar{R}^{T_1+T_2}}\,(\mathrm{S})}{P \wedge \hat{A} \wedge \bar{\hat{B}}^{T_1} \rightarrow \bar{R}^{T_1+T_2}}\,(\mathrm{D})}{P \wedge \hat{A} \nearrow^{T_1} \hat{B} \rightsquigarrow^{T_1+T_2} R}\,(\mathrm{IP})$$

which reduces to the following inference tree after retaining only the premises and the conclusion

$$\dfrac{P \wedge \hat{A} \rightsquigarrow^{T_1} Q \qquad Q \wedge \hat{B} \rightsquigarrow^{T_2} R \qquad P \wedge \hat{A} \wedge \bar{\hat{B}}^{T_1} \qquad Q \wedge \hat{B} \qquad R}{P \wedge \hat{A} \nearrow^{T_1} \hat{B} \rightsquigarrow^{T_1+T_2} R}\,(\mathrm{ASD})$$

providing an Action Sequence Deduction (ASD) rule ready to be used for procedural reasoning. Three additional premises have been added $P \wedge \widehat{A} \wedge \overleftarrow{\widehat{B}}^{T_1}$, $Q \wedge \widehat{B}$ and R.

7 Conclusion

TODO: implement more temporal and procedural rules, support temporal intervals, behavior trees, introduce Temporal Truth Value. Integrate Event Calculus.

Hint regarding intervals: define a notion of parameterized union, where for temporal interval the parameters would be temporal variables ranging over intervals.

References

1. Goertzel, B., Ikle, M., Goertzel, I.F., Heljakka, A.: Probabilistic Logic Networks. Springer US (2009)