Temporal and Procedural Reasoning for Rational Agent Control in OpenCog

Nil Geisweiller and Hedra Yusuf

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

Abstract. TODO

Keywords: Temporal · 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.

2 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.

3 Outline

1. Temporal reasoning
2. ROCCA
3. Minecraft experiment

4 Recall: Probabilistic Logic Networks

PLN, which stands for Probabilistic Logic Networks, is a mixture of predicate and term logic that has been probabilitized to properly handle uncertainty. It has two types of rules

1. one type for introducing relationships from direct observations,
2. the other for introducing relationships from existing relationships.

As such it is especially suited for building an ongoing understanding of an unknown environment (using direct introduction rules), and then planning in that environment (using indirect introduction rules).

4.1 Elementary Notions

Graphically speaking, PLN statements are sub-hypergraphs[1] made of links and nodes, called *Atoms*, decorated with *Truth Values* that can be understood as uncertain probabilities [**?**]. Syntactically speaking however, PLN statements are not very different from statements expressed in another logic, except that they are usually formatted in prefixed-operator indented-argument style to emphasize their graphical nature and leave room for truth values. There is a large variety of constructs for PLN, here we will focus primarily on constructs for manipulating predicates. Let us recall that predicates are functions that take tuples of Atoms and output boolean values

$$P, Q, R, \ldots : Atom^n \mapsto \{True, False\}$$

Within predicate constructs there are two classes of operators

1. one for defining predicate from instances, such as *Evaluation* and *Lambda*,
2. and another for combining existing predicates, such as *And*, *Not* and *Implication*.

Let us present these operators below, corresponding to the minimum subset we will need in the rest of the paper.

- Evaluation:

$$\begin{array}{l} Evaluation\ \langle TV \rangle \\ \quad P \\ \quad E \end{array}$$

  states that $P(E)$ outputs *True* to a degree set by the truth value TV.
- Lambda:

$$\begin{array}{l} Lambda\ \langle TV \rangle \\ \quad x \\ \quad P(x) \end{array}$$

  is a predicate constructor with variable x and predicate body $P(x)$, where the true value TV corresponds to the probability $\mathbf{Pr}(P)$ of $P(x)$ to output *True* for a random input.

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

- Conjunction:

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

  represents the predicate obtained by taking the conjunction of P and Q, or equivalently the indicator function corresponding to the intersection of the *satisfying sets* of P and Q. The truth value TV then represents an estimate of the probability $\mathbf{Pr}(P, Q)$ of the conjunction of P and Q.
- Negation:

$$\begin{array}{l} \mathit{Not}\ \langle TV \rangle \\ \quad P \end{array}$$

  represents the negation of P, or equivalently the indicator function corresponding to the complement of the satisfying set of P. The truth value TV then represents an estimate of the probability $\mathbf{Pr}(\neg P)$ of the negation of P.
- Implication:

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

  represents the predicate Q conditioned on P, that is only defined for instances x for which $P(x)$ is *True*. The truth value TV then represents an estimate of the conditional probability $\mathbf{Pr}(Q|P)$. There is some subtleties to take into account due to the fact $P(x)$ can actually be partially true (stated by the truth values of *Evaluation* links as explained above), but this resolves nicely by assuming degrees of truth are probabilistic. More is explained about that below.

Truth values are fundamentally second order probability distributions. However in practice they are usually represented by two numbers, a strength and a confidence, both ranging from 0 to 1. The strength represents a probability while the confidence represents a precision over that probability. Underneath, strength and confidence can be mapped into a second order distribution such as a Beta distribution [TODO: add figure].

4.2 Inference Rules

Beside operators, inferences rules are used to construct PLN statements and calculate their truth values. They mainly fall into two categories, direct and indirect. Direct rules infer abstract knowledge from direct evidence, while indirect rules infer knowledge by combining existing abstractions, themselves inferred directly or indirectly. There are dozens of inference rules but for now we will only recall two which are needed for the paper:

1. *Implication Direct Introduction Rule*
2. *Deduction Rule*

The Implication Direct Introduction Rule (IDI) takes *Evaluation* links as premises and produces an *Implication* link as conclusion, formally depicted by the following proof tree

$$\frac{\begin{array}{l}\textit{Evaluation}\ \langle TV_i^P\rangle\\ \quad P\\ \quad E_i\end{array} \qquad \dots \qquad \begin{array}{l}\textit{Evaluation}\ \langle TV_i^Q\rangle\\ \quad Q\\ \quad E_i\end{array}}{\begin{array}{l}\textit{Implication}\ \langle TV\rangle\\ \quad P\\ \quad Q\end{array}}\ \text{(IDI)}$$

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

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

$$TV.c = \frac{n}{n+k}$$

where $TV.s$ and $TV.c$ respectively represent the strength and the confidence of TV, k is a system parameter, and $f_\wedge$ is a function embodying a probabilistic assumption about the intersection of the events corresponding to the *probabilitized degrees of truth* of $P(E_i)$ and $Q(E_i)$. Such function typically ranges from the product (perfect independence) to the min (perfect overlap).

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

$$\mathbf{Pr}(R|Q,P) = \mathbf{Pr}(R|Q)$$

can be formally depicted by the following proof tree

$$\frac{\begin{array}{l}\textit{Implication}\ \langle TV^{PQ}\rangle\\ \quad P\\ \quad Q\end{array} \qquad \begin{array}{l}\textit{Implication}\ \langle TV^{QR}\rangle\\ \quad Q\\ \quad R\end{array} \qquad P\langle TV^P\rangle \qquad Q\langle TV^Q\rangle \qquad R\langle TV^R\rangle}{\begin{array}{l}\textit{Implication}\ \langle TV\rangle\\ \quad P\\ \quad R\end{array}}\ \text{(D)}$$

The reader may notice that three additional premises have been added, corresponding to the probabilities $\mathbf{Pr}(P)$, $\mathbf{Pr}(Q)$ and $\mathbf{Pr}(R)$. This is a consequence of the Markov property. The exact formula for that variation will not be recalled[3] here but it merely derives from

$$\mathbf{Pr}(R|P) = \mathbf{Pr}(R|Q,P) \times \mathbf{Pr}(Q|P) + \mathbf{Pr}(R|\neg Q,P) \times \mathbf{Pr}(\neg Q|P)$$

---

[2] Dealing with unreliable direct evidence involves expensive convolution products and is outside of the scope of this paper.

[3] More information can be found in [**?**]

5 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* as regular predicates with a temporal dimension

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

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

5.1 Temporal Operators

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

Lead is a temporal operator used to shift the temporal dimension of a temporal predicate. It is formally defined as follows

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

:=

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

where the first line of the *Lambda* link is the variable declaration and the second line is the body representing a temporally shifted reconstruction of P. Informally one might say the *Lead* operator gives a peek into the future, or equivalently, that it brings the future into the present.

SequentialAnd is a temporal conjunction, formally defined as

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

:=

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

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

PredictiveImplication likewise defines a temporal implication, formally

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

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

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.

5.2 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
$\mathit{Evaluation}(P, \mathit{List}(X_1, ..., X_n))$	$P(X_1, ..., X_n)$	1
$\mathit{Lambda}(t, \mathit{AtTime}(\mathit{Execution}(A), t))$	$\widehat{A}$	1
$\mathit{Lead}(P, T)$	$\overleftarrow{P}^T$	1
$\mathit{And}(P, Q)$	$P \wedge Q$	2
$\mathit{SequentialAnd}(T, P, Q)$	$P \prec^T Q$	3
$\mathit{Implication}(P, Q)$	$P \rightarrow Q$	4
$\mathit{PredictiveImplication}(T, P, Q)$	$P \leadsto^T Q$	4

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

5.3 Temporal Rules

Given that we can now introduce our temporal rules, (PI), (IP), (S) and the most important one Temporal Deduction (TD)

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

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

$$\dfrac{\dfrac{P \leadsto^{T_1} Q}{P \rightarrow \overleftarrow{Q}^{T_1}}\,(\mathrm{PI}) \quad \dfrac{\dfrac{Q \leadsto^{T_2} R}{Q \rightarrow \overleftarrow{R}^{T_2}}\,(\mathrm{PI})}{\overleftarrow{Q}^{T_1} \rightarrow \overleftarrow{R}^{T_1+T_2}}\,(\mathrm{S}) \quad P \quad \dfrac{Q}{\overleftarrow{Q}^{T_1}}\,(\mathrm{S}) \quad \dfrac{R}{\overleftarrow{R}^{T_1+T_2}}\,(\mathrm{S})}{\dfrac{P \rightarrow \overleftarrow{R}^{T_1+T_2}}{P \leadsto^{T_1+T_2} R}\,(\mathrm{IP})}\,(\mathrm{D})$$

5.4 Procedural Reasoning

Likewise, we can use the same temporal to regular deduction mapping to build inference rules for procedural reasoning

$$\dfrac{\dfrac{\dfrac{P \wedge \widehat{A} \leadsto^{T_1} Q}{P \wedge \widehat{A} \rightarrow \overleftarrow{Q}^{T_1}}\,(\mathrm{PI}) \quad \dfrac{\widehat{B}}{\overleftarrow{\widehat{B}}^{T_1}}\,(\mathrm{S})}{P \wedge \widehat{A} \wedge \overleftarrow{\widehat{B}}^{T_1} \rightarrow \overleftarrow{Q}^{T_1} \wedge \overleftarrow{\widehat{B}}^{T_1}}\,(\mathrm{C}) \quad \dfrac{\dfrac{Q \wedge \widehat{B} \leadsto^{T_2} R}{Q \wedge \widehat{B} \rightarrow \overleftarrow{R}^{T_2}}\,(\mathrm{PI})}{\overleftarrow{Q}^{T_1} \wedge \overleftarrow{\widehat{B}}^{T_1} \rightarrow \overleftarrow{R}^{T_1+T_2}}\,(\mathrm{S}) \quad P \wedge \widehat{A} \wedge \overleftarrow{\widehat{B}}^{T_1} \quad \dfrac{Q \wedge \widehat{B}}{\overleftarrow{Q}^{T_1} \wedge \overleftarrow{\widehat{B}}^{T_1}}\,(\mathrm{S}) \quad \dfrac{R}{\overleftarrow{R}^{T_1+T_2}}\,(\mathrm{S})}{\dfrac{P \wedge \widehat{A} \wedge \overleftarrow{\widehat{B}}^{T_1} \rightarrow \overleftarrow{R}^{T_1+T_2}}{P \wedge \widehat{A} \prec^{T_1} \widehat{B} \leadsto^{T_1+T_2} R}\,(\mathrm{IP})}\,(\mathrm{D})$$

6 Rational OpenCog Controlled Agent

To experiment with temporal and procedural reasoning in the context of embodied virtual agents in unknown environments we have implemented a project called ROCCA, which stands for *Rational OpenCog Controlled Agent*. ROCCA essentially acts as an interface between virtual environments such as Malmo [**?**] or OpenAI Gym [**?**] and OpenCog. It provides an Observation-Planning-Action control loop as well as various launchers to run OpenCog processes such as PLN reasoning, pattern mining, etc. Provided a top goal, such as maximizing a reward, ROCCA orchestrates the necessary learning and the planning to fulfill that goal. One may possibly see ROCCA as a reinforcement learning agent with the particularity that learning and planning are, at least in principle, entire done via reasoning. In that respect it is similar in spirit to OpenNARS for Applications (ONA) [**?**] but uses PLN as its core reasoning logic rather than NAL [**?**].

ROCCA is composed of two main processes, one for real-time agent control and another one for non-reactive background learning. In principle these two processes could happen in parallel, though as of right now they occur as distinct alternating phases.

6.1 Control Phase

The control phase is composed of control cycles, each decomposed into Observation, Planning and Acting steps, more precisely

1. Observation step:
   (a) receives and timestamps observations from the environment,
   (b) stores the timestamped observations in the atomspace.
2. Planning step:
   (a) selects the goal for that iteration,
   (b) finds plans fulfilling that goal,

   (c) given these plans, deduces a probabilistic distribution of actions,
   (d) selects the next action according to the deduced probabilistic distribution.
3. Acting step:
   (a) timestamps and stores in the atomspace the selected action,
   (b) runs the selected action and by that updates the environment,
   (c) receives the reward from the environment,
   (d) timestamps and stores the reward in the atomspace.

None of these steps are difficult to carry with the exception of deducing a probabilistic distribution of actions. For that we use a variation of Solomonoff induction described in [**?**] which is especially suited for plans described by conditional second order distributions, in other words *PredictiveImplication* links. More specifically plans are *PredictiveImplication* links of the form

$$C \wedge A \leadsto^T G$$

called *Cognitive Schematics*. Which can be read as *"in some context C, if some action (elementary or composite) A is executed, then after T time units, the goal G is likely to be fulfilled"*. The degree of expected fulfillment is specified by the truth value of the *PredictiveImplication* link, not indicated in that notational format but present in the extended Atomese format. The difficulty then comes down to discovering cognitive schematics that are as informative and applicable as possible.

6.2 Learning Phase

As hinted above, the ultimate goal of the learning phase is to discover maximally useful cognitive schematics, and by useful it is specifically meant that they are as XXX and cover as many situations as possible.

NEXT

7 Experiment with Simple Minecraft Environment

8 Conclusion

References