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 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 Recall

Let us first recall the minimum portion of PLN we will need to describe the temporal logic used in this paper.

Graphically speaking a PLN statement is a graph made of links and nodes, called *Atoms*, decorated with *Truth Values* [**?**]. It is said that such graph is a hypergraph or a metagraph because links can point to links [**?**]. Syntactically speaking however, a PLN statement is not very different than a statement expressed in another logic. Here will focus primarily on Atoms representing predicates

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

as well as a small subset of connectors operating on these predicates, recalled below.

– Conjunction:

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

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:

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

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:

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

represents the predicate Q conditioned on P, that is only defined for the instances x for which $P(x)$ is true. The truth value TV then represents an estimate of the conditional probability $\mathbf{Pr}(Q|P)$.

Truth values are, fundamentally speaking, 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 via the Beta distribution [TODO: add figure].

4.2 Inference Rules

Direct Introduction Rules NEXT

Indirect Introduction Rules

5 Temporal Logic

The temporal logic used here builds upon what is described in the Chapter 14 of the PLN book [TODO: cite]. Let us define that

5.1 Mapping Temporal into Atemporal Statements

6 Rational OpenCog Controlled Agent (ROCCA)

7 Experiment with Simple Minecraft Environment

8 Conclusion

References