# Rational OpenCog Controlled Agent

Nil Geisweiller and Hedra Yusuf

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

**Abstract.** TODO

**Keywords:** Symbolic Reinforcement Learning · Procedural Reasoning · OpenCog · Minecraft

## 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.

### 1.1 Related Work

### 1.2 Contributions

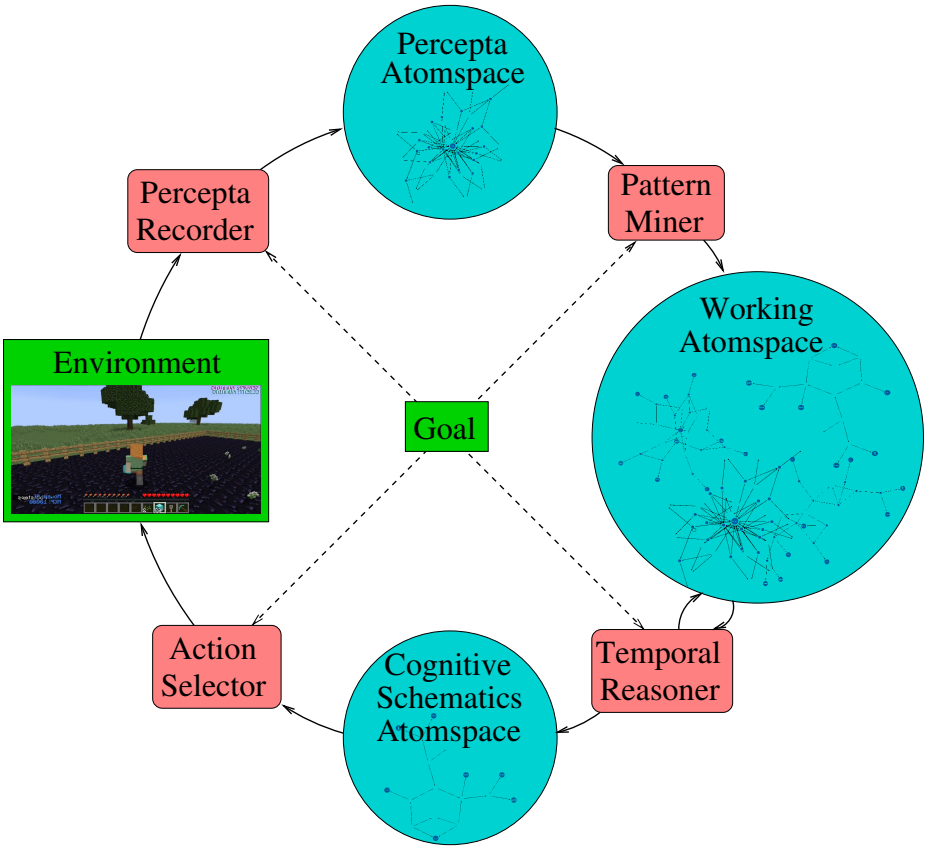
The contributions of that paper are:

1. Design an architecture for controlling an agent based on that temporal reasoning extension.

### 1.3 Outline

1. ROCCA
2. Minecraft experiment

## 2 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.

### 2.1 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

$$C \wedge A \rightsquigarrow^{T} G$$

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.

### 2.2 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 predictive and cover as many cases as possible.

TODO: pattern mining and reasoning.

## 3 Experiment with Simple Minecraft Environment

In this experiment we built a minecraft environment using Malmo, which is a platform for Artificial Intelligence experimentation and research built on top of Minecraft. The demo environment consists of a small house with a locked door, diamonds inside and a key to get into the house. The agent, initially located outside of the house, can perform different actions like getting a key, opening a door of the house and collecting the diamonds in order to achieve a reward.

The aim of this experiment is to make the ROCCA agent learn from the actions and perceptions in the minecraft environment and do efficient planning so as to be able to collect as many diamonds as possible and accumulate reward. The ROCCA agent will be able to perform a series of possible actions with a goal of achieving a reward and learns from them by applying PLN (Probabilistic Logic Networks) and Pattern Miner, which can be seen as a specialized form of PLN reasoning. The Planning, the discovery of cognitive schematics, is also handled by PLN and its temporal reasoning rule base.

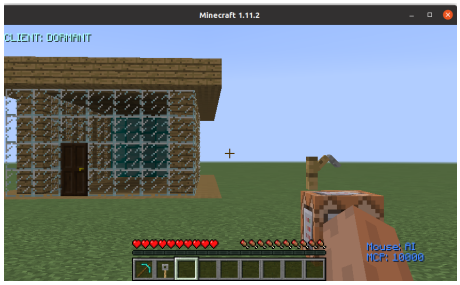



**Fig. 1.** Simple Minecraft demo with a house and a key.

There are lists of allowed actions provided by minecraft that an agent can perform like moving, turning, picking etc.. but due to the limited processing capacity we have to handle the observations from each action and to reduce complexity, we proposed to have an abstract action and perception where unnecessary details have been omitted. With that we generate three abstract actions namely go-to-key, go-to-house, go-to-diamonds where each of them contains a series of actions and returns an abstract perception about where the agent is (inside house, outside house, next to closed door etc..), about its inventory (has key, diamond pickaxe etc..) and the reward of completing a given action.

We perform various experiments tuning different parameters. A typical experiment has two iterations of the learning-training process with a duration of fifty iterations for each training. In the first cycle the agent will not have prior knowledge hence no learning will take place. The agent pursues the environment and builds its knowledge base by trying a combination of fifty randomly weighted actions. At the end of the first cycle the agent will have enough background knowledge to apply Pattern miner and PLN temporal reasoning. Hence,

during the second cycle, the agent will be able to learn and plan the desired cognitive schematics which leads to a positive goal of getting a reward.
The ROCCA agent is able to learn the following cognitive schematics with higher strength.

$$outside(self, house) \wedge \widehat{go\_to(key)} \rightsquigarrow^1 hold(self, key)$$

$$hold(self, key) \wedge \widehat{go\_to(house)} \rightsquigarrow^1 inside(self, house)$$

$$inside(self, house) \wedge \widehat{go\_to(diamond)} \rightsquigarrow^1 reward(1)$$

In this experiment, we measure the agent's performance by the cognitive schematics learned and accumulated rewards achieved. The ROCAA agent is successful in learning the required cognitive schematics which leads the agent to collect more rewards in the second cycle. However, these findings with a simple minecraft environment with only few actions might not tell the overall performance of ROCCA. As a future work, further extensive experiments are needed to conclude the performance achieved.

## 4 Conclusion

## References