Ipseity

An open-source platform for synthesizing and validating artificial cognitive systems in multi-agent systems

– *Application to Reinforcement Learning* –

Fabrice LAURI

IRTES-SeT

2014, 20 January – LORIA, Nancy

Outline

1. Introduction and motivations
2. Objectives and Kernel Concepts
3. Available Elements
4. Technical details and Architecture
5. Demo
6. Conclusion and Future Works

Introduction

As pointed out by **[Kovacs, 2011]**, solving sequential decision problems has empirical aspects which involves:

- writing codes,
- conducting experiments, and
- comparing results obtained from alternative approaches.

There are a wide range of environments, agent architectures, decision-taking algorithms and performance measures.

Introduction: range of environments

Multi-agent environments are defined according to many features:

- assumptions on the state/action/time spaces
- deterministic or stochastic state transitions
- static or dynamic decision-taking processes
- populated by independent learner agents, cooperative agents or competitive agents...

Introduction: range of decision-taking techniques

RL algorithms have many dimensions **[Sutton, 1998; Busoniu, 2010]**.

The decision-taking process of learning agents may:

- be based on Value-Iteration, Policy Iteration or Policy Search
- be model-based or model-free
- be on-policy or off-policy
- be offline or online
- use a wide range action selection method
- use a wide range of function approximators

But there are also alternative approaches coming from other fields...

Introduction: experiments and statistics

Besides, experiments may generate multiple statistics, such as:

- the agents' interactions over time,
- the discounted return per agent over time,
- the number of steps taken by some agent to reach a goal,
- the Q-function.

Motivations

- There are currently no fully-featured experimental platforms, such as WEKA for Supervised Learning, dedicated to Reinforcement Learning.
- "Science is founded on replicability. Having access to the source code used by others is invaluable and makes for better science.", **[Kovacs, 2011]**
- Application of RL is still more an art than a science...

Objectives

- *Ipseity* has been especially designed to facilitate the synthesis and the validation of decision-making mechanisms within Multi-Agent Systems.
  Artificial Cognitive Systems embed a set of Artificial Intelligence techniques that may be executed alternatively or concurrently, in parallel or sequentially.
- *Ipseity* is targeted at a broad range of users interested in artificial intelligence in general, including industrial practitioners, as well as machine learning researchers, students and teachers.

Kernel Concepts

- Agent
- Taxon
- Multi-Agent System
- Cognitive System
- System Scheduling
- Scenario
- Simulation
- Workspace

Kernel Concept: Agent

Definition

Agent: Any agent i is an autonomous entity that can perceive its environment thanks to its sensors from *stimulus signals* $\mathcal{S}_i$, and act upon it through its effectors using *action signals* $\mathcal{A}_i$.
Every agent performs actions according to the decisions elaborated within its *cognitive system*.

Kernel Concept: Agent

Properties:

- Agents are grouped into *taxons*.
- Any agent belongs to one and only one taxon during its entire virtual life.
- Agents belonging to the same taxon have been grouped together because they possess the same internal structure, they can sense the same type of information (percepts), they can carry out the same type of actions and their decision process can be implemented by the same class of cognitive systems.

Kernel Concept: Taxon

Definition

Taxon: A taxon k is a tuple $\tau_k = (C_k, \mathcal{P}_k, \mathcal{U}_k)$ where:

- C_k: class of cognitive systems, like:
  - Reinforcement Learner with Tabular Q-Learning and ϵ-Greedy,
  - RL with Sarsa using a Linear Function Approximator and SoftMax,
  - Ad-hoc hard-coded controller,
  - A combination of the preceding techniques.
- $\mathcal{P}_k$: *perception set*,
- $\mathcal{U}_k$: *response set*.

Any agent i belonging to taxon k uses the percepts from $\mathcal{S}_i = \mathcal{P}_k$ and selects its actions from $\mathcal{A}_i = \mathcal{U}_k$.

Kernel Concept: Multi-Agent System (MAS)

Definition

Multi-Agent System: Tuple $M = (\mathcal{S}, \mathcal{T}, \overline{f}, g, h, \Lambda, \Omega)$ where:

- $\mathcal{S}$: set of possible environmental states.
- $\mathcal{T} = \{\tau_1, \ldots, \tau_M\}$: set of taxons.
- $\overline{f}$: transition function between states.
- $g : \mathcal{S} \to \mathbf{2}^{\mathbb{N}}$: function mapping populations of agents from states.
- $h : \mathbb{N} \to \mathbb{N}$: function that associates a taxon to any agent. In particular, $h^{-1}(\{k\})$ returns the set of agents belonging to taxon k.

Definition

Multi-Agent System (cont.):

- $\Lambda = (\lambda_k)_{k \in [M]}$: family of functions $\lambda_k : \mathcal{S} \times h^{-1}(\{k\}) \rightarrow \mathcal{P}_k$. $\lambda_k(s, i)$ extracts a stimulus for agent $i \in h^{-1}(\{k\})$ from a given state s.
- Ω: system scheduling.

Kernel Concept: System Scheduling

Definition

System Scheduling: Specifies the frequency at which the environment updates, the frequency at which each agent perceives and the frequency at which each agent makes a decision.
It can be defined by a tuple $\Omega = (\upsilon, \mu, \nu, \Delta_t)$, where:

- $\upsilon : \mathbb{R} \rightarrow \mathbf{2}$. $\upsilon(t)$ indicates whether the environment updates at time t.
- $\mu, \nu : \mathbb{R} \times \mathcal{G} \rightarrow \mathbf{2}^{\mathcal{G}}$, with $\mathcal{G} \subset \mathbf{2}^{\mathbb{N}}$. $\mu(t, G)$ (respectively $\nu(t, G)$) specifies the subset of agents that perceive (respectively take decisions) at time t from the agent population $G \in \mathcal{G}$.
- Δ_t is the simulation frequency specified by the user.

Kernel Concept: Scenario

Definition

Scenario: A scenario σ is the data of a tuple $\sigma = (I, \tilde{\mathcal{S}})$, where:

- $I \in \mathbb{N}$: UID of the scenario,
- $\tilde{\mathcal{S}} \subset \mathcal{S}$: set of initial environmental states.

When a scenario σ is selected, an initial state $s_0 \in \tilde{\mathcal{S}}$ is used to initialize the environment.

Kernel Concept: Simulation

Definition

Simulation: A simulation consists of a sequence of N scenarios $\{\sigma_1, \sigma_2, \ldots, \sigma_N\}$ aimed at studying the performance of the agents' behaviors under different conditions.

Kernel Concept: Workspace

Definition

Workspace: A workspace is a repository where simulation data (agents' interactions, performance statistics) is stored.

Available Elements

Available environments:

- Single-Agent: Acrobot, Cartpole, DoubleIntegrator, InvertedPendulum, MountainCar, RasendeRoboter, Rubik's Cube
- Multi-Agent: SmartGrid, Delirium2*

Available Learning Techniques:

- Online model-free: Q-Learning (off-policy), Sarsa (on-policy)
- Offline model-free: RCAL **[Piot, 2014]**

Available Action Selection Techniques:

- ϵ-Greedy
- Softmax

Available Elements

Available Q-Memory:

- Static (preallocated) Lookup Table (for finite state-action space whose size is known)
- Dynamic Lookup Table (for finite state space whose size is unknown, but "manageable")
- Linear Function Approximator of the Q-function: $\tilde{Q}(s, a) = \theta^T \phi(s, a)$

Feature extraction methods:

- Cerebellar Model Articulation Controller (CMAC)
- Customized feature vectors

Available Elements

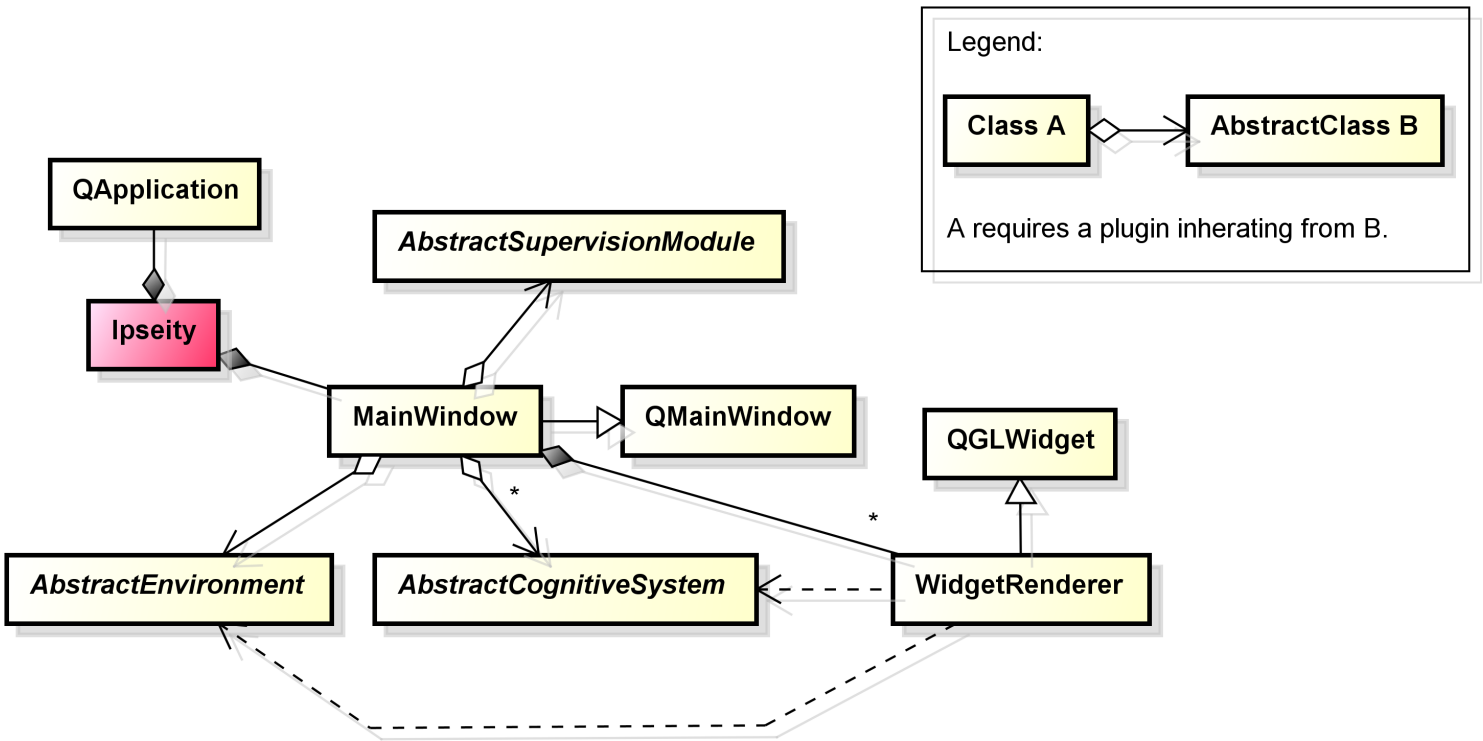
Controllers for the environments:

- Delirium2 (*SWI-Prolog*)
- RasendeRoboter (*SWI-Prolog*)
- Rubik's Cube (*SWI-Prolog*)

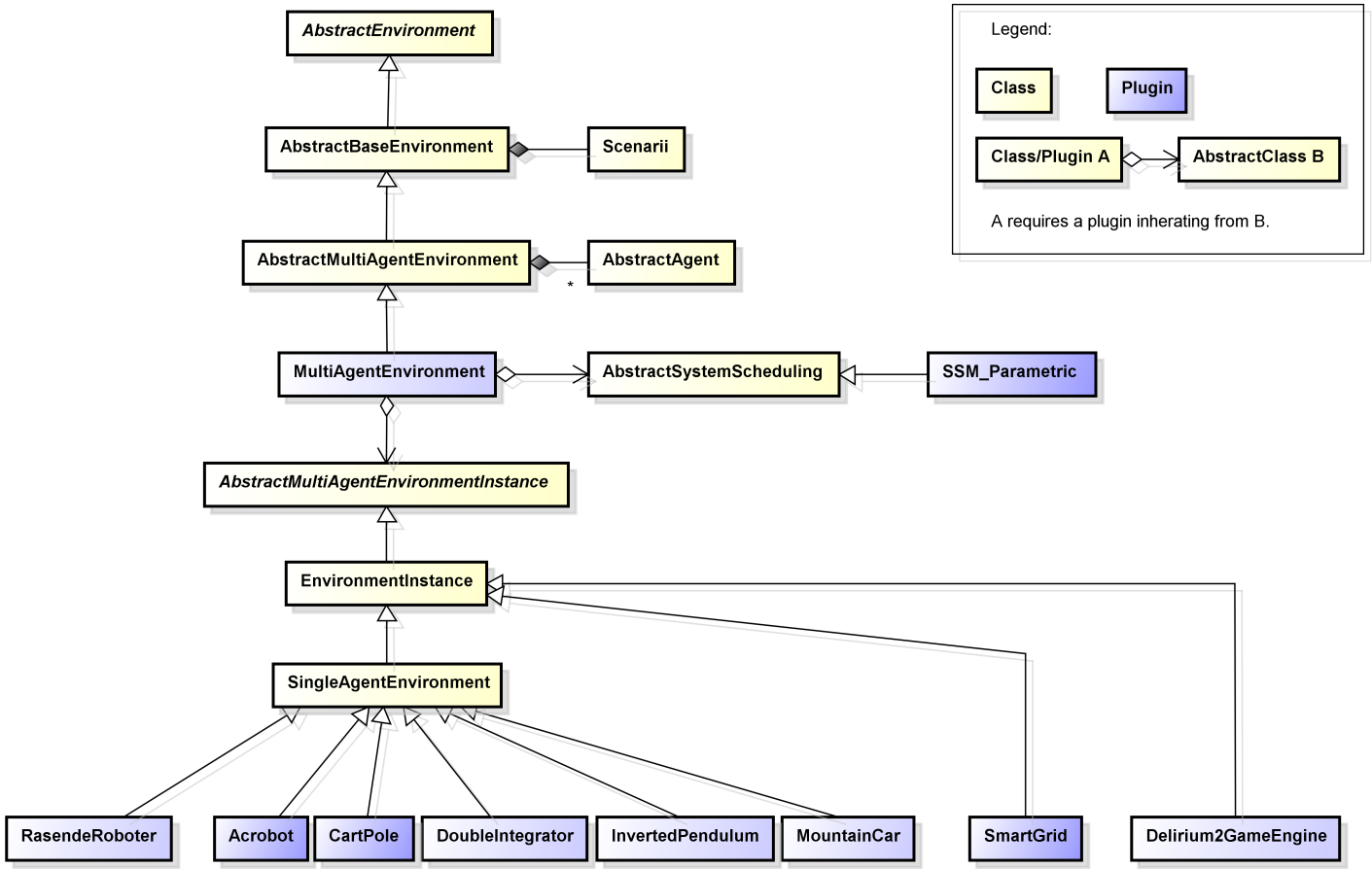
Technical details

- Environments (MAS), cognitive systems, system schedulings and scenario supervisors can be plugged/unplugged easily under *Ipseity*.
- They are implemented into Dynamic Link Libraries (DLLs).
- The cognitive systems and the environments available under *Ipseity* must be defined within XML files.
- *Ipseity* has been developed in C++ under the *Qt* framework (possibly multi-platform).
- Currently, available only under *Windows*.
- It includes cognitive system modules (predefined controllers) implemented in *SWI-Prolog*, in *Python* or communicating with a *Java*-based simulator.

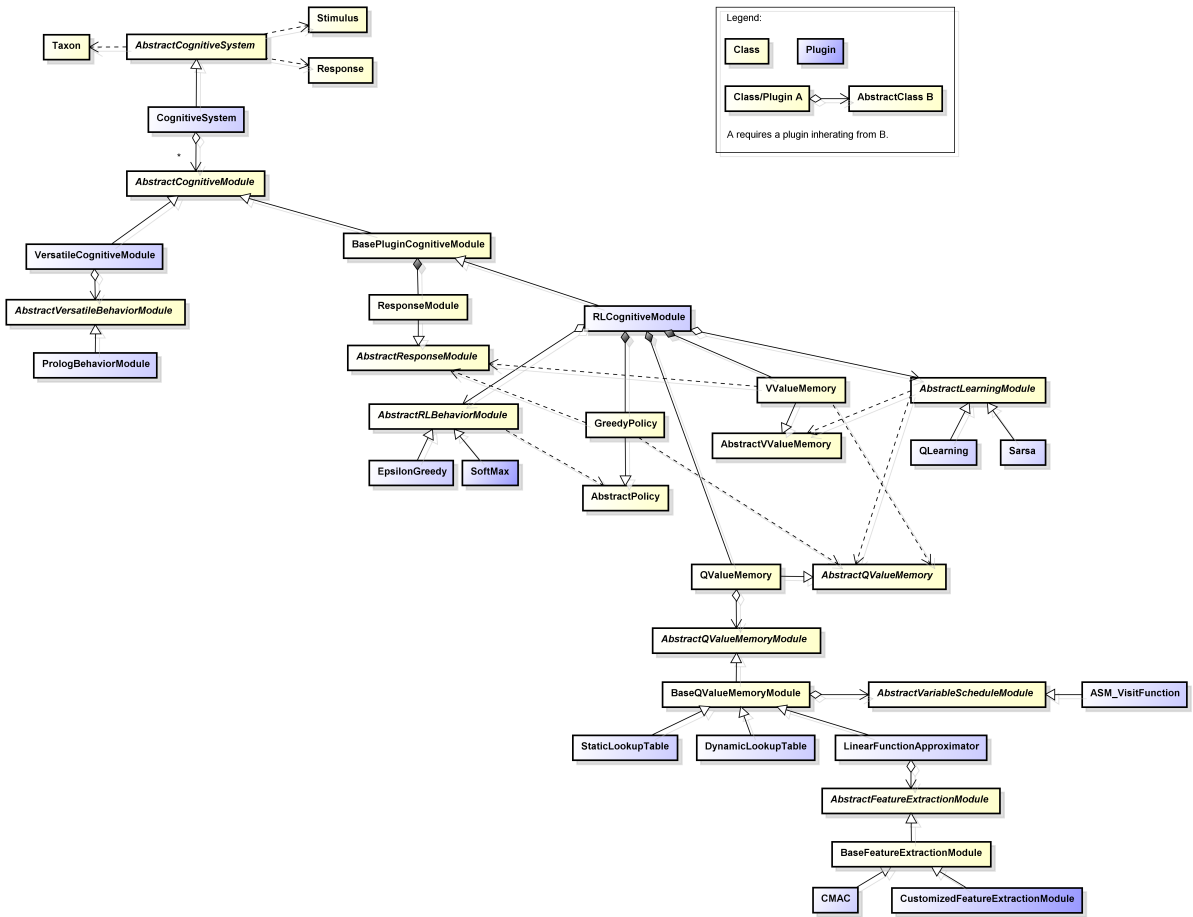
Architecture: *Ipseity* application



Architecture: Environment



Architecture: Cognitive system



Demo...

Conclusion

- *Ipseity* is daily used as a course support in AI and RL.
- It has already been used successfully to manage power flows in simulated microgrids using multi-agent reinforcement learning **[Lauri, 2013b]**.
- *Ipseity* can be freely downloaded from:

  `http://www.ipseity-project.com`

  under a GNU *GPLv3* open-source licence.
- It is still an experimental platform: it currently provides mostly online model-free RL algorithms.
- Though *Ipseity* is highly modular and broadly extensible.

Future Works

Possible enhancements of *Ipseity* include:

1. Allowing the user to easier combine the modules of a cognitive system
2. Scripting the experiments
3. Implementing efficient Policy Search algorithms for solving continuous sequential decision problems
4. Providing tools for facilitating the comparison between different decision-taking techniques
5. Establishing a database for storing and ease the sharing of experiment data
6. Providing a web-interface for accessing the results of experiments under progress

References

- **[Sutton, 1998]** Sutton *et al.*, "Reinforcement Learning: an introduction", MIT Press, 1998.
- **[Busoniu, 2010]** Busoniu *et al.*, "Reinforcement Learning and Dynamic Programming using Function Approximators", CRC Press, 2010.
- **[Kovacs, 2011]** Kovacs *et al.*, "On the analysis and design of software for RL, with a survey of existing systems", Machine Learning, 2011.
- **[Lauri, 2013a]** Lauri *et al.*, "Ipseity - A Laboratory for Synthesizing and Validating Artificial Cognitive Systems in Multi-Agent Systems", ECML, 2013.
- **[Lauri, 2013b]** Lauri *et al.*, "Managing Power Flows in Microgrids using Multi-Agent Reinforcement Learning", Agent Technologies in Energy Systems (workshop of AAMAS), 2013.
- **[Piot, 2014]** Piot *et al.*, "Boosted and Reward-regularized Classification for Apprenticeship Learning", AAMAS, 2014.