# Integration of General Game Playing with RL-glue

José Luis Benacloch Ayuso
ETSINF, Universitat Politècnica de València, València, Spain.
jobeay@fiv.upv.es

March 15, 2012

**Abstract**

This paper describes the integration between the GGP (General Game Playing) project and RL-glue.

**Keywords:** Reinforcement learning, RL-glue, GGP, Game Description Language, Jocular, RL-GPP.

## 1 Introduction

This note describes how the GGP (General Game Player) project, using the GGP server [4] can be integrated with RL-glue. This is performed through the use of Jocular[5], a platform which creates players for the GGP server. This is part of a bigger project, called RL-GGP, which also integrates some RL algorithms developed by Hado van Hasselt in RL-glue. The final motivation of this project is to be able to compare RL algorithms in the context of games, as suggested by some notions of intelligence evaluation using games and social environments [12][10][13][14], which in turn, follow some previous approaches about machine evaluation [1][2][11][6][8][7][9][3].

In this document, we just focus on the integration between RL-glue and jocular. In another document, we will deal with the integration between RL-glue and Hado van Hasselt's algorithm.

## 2 Integration

We will first describe RL-glue and the GGP server. Next we will see their integration and some hints about installation.

### 2.1 RL-glue

Reinforcement learning (RL) [15] is a relevant area in artificial intelligence which aims at developing methods and algorithms for building agents which make actions in an environment and learn from rewards.

RL-Glue[1] (Reinforcement learning Glue) [16] is a standard interface for the reinforcement learning community developed by Brian Tanner and Adam White. This interface allows researchers and practitioners to connect agents and environments written in very different languages, such as C/C++, Java, Lisp, Matlab or Python, and to design experiments. This is a significant step in the RL community, since a lot of work is usually devoted whenever new algorithms are implemented and they need to be compared or connected to other systems. For instance, RL-Glue is being used

[1]The webpage of the project is `http://glue.rl-community.org/wiki/Main_Page`.

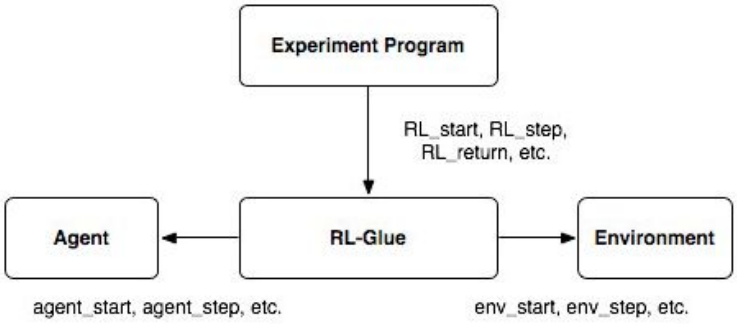


Figure 1: Schematic view of how the Experiment Program, the Agent and the Environment modules connect in RL-Glue. Image taken from the RL-Glue webpage.

in the now regular "Reinforcement learning competition" [17]. Figure 1 shows a schematic view of how RL-Glue connects agents and environments, and experiments.

One of the current limitations of RL-glue is that its RL-Library is still quite limited. While the number of environments to play with is increasing, the number of algorithms is very small. At present, it only includes a random agent (RandomAgentJava) and a version of SARSA (Tile Coding Sarsa Lambda Java).

### 2.2 GGP and Jocular

GGP (General Game Playing) is an area of research which is concerned about the design and evaluation of artificial intelligence programs which can successfully play more than one game.

GGP-Server is a tool developed by the Stanford Logic Group at Stanford University. It is implemented in Java[2]. The goal of the GGP-Server is to provide an easy way to try different generic players (using Jocular) with a wide range of games written in GDL (Game Description Language). The server connects the player which are online and will let them play with the games that are loaded onto the server.

Jocular is a generic player which is written in Java, also developed and maintained by the Stanford Logic Group, for the GGP-Server[3].

While the main goal of Jocular is pedagogical, it is provided as a help to whoever whant to start with GGP (General Game Playing), since it brings an interface which can allow developers to connect players to GGP-Server in an independent way. These players can be programmed in such a way that they make ther moves as programmed, using a random strategy, some kind of preprogrammed strategy or learning algorithms. The player receives all the information and updates by the GGP-Server, which is in charge of keeping the game working and switching the turns.

### 2.3 Integration of RL-glue and GGP through Jocular

Jocular is used as the link between RL-Glue and GGP-Server. To sychronise them, we need to ensure that all the information that Jocular gets from GGP-Server passes to RL-Glue, and that the decisions made by the algorithms in RL-Glue pass to Jocular to finally reach GGP-Server.

This can be seen in figure 2.

[2]It can be downloaded from http://sourceforge.net/apps/trac/ggpserver/wiki/GameController.

[3]It can be downloaded from http://games.stanford.edu/resources/reference/jocular/jocular.html.

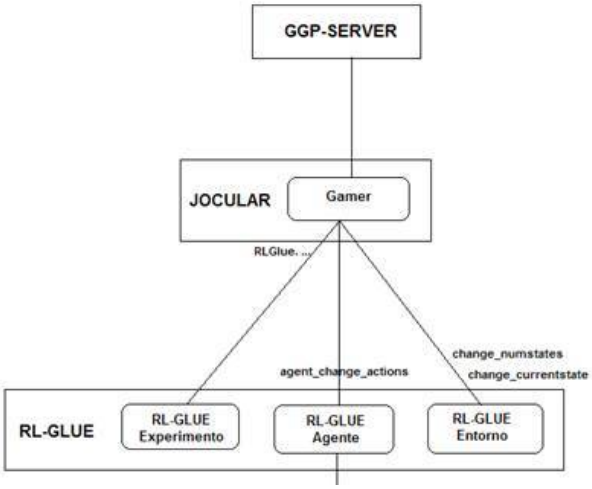


Figure 2: Schematic view of how the RL-Glue is connected to GGP through Jocular.

Since we are only interested in the decision that RL-Glue makes, we have implemented the parts of RL-Glue in the Jocular method which gets the answer to the action which the Jocular player performs (the method `play`). Each time that this method is executed, the states which have been visited are updated and the new actions which are available are considered. Before starting the process, the system checks whether there is a file with states and actions in previous games. This makes it possible to recover the state-action matrix between different games.

Once RL-Glue is ready with the state-action setting, it starts what RL-Glue calls the *experiment*, updates the environment and the RL-Glue agent with the information which has been obtained from Jocular.

When the *experiment* starts, if it is the first call to the method `play` then the agent is initialised (method `RLGlue.RL_agent_start`) as well as the environment RL-Glue (method `RLGlue.RL_env_start`). With this we also get the action that RL-Glue has decided as a start of the game. If it is not the first call to `play` then it just simply gets the action that the agent has decided according to the current state (method `RLGlue.RL_agent_step`).

Next, in case the decision is against the rules (detected by Jocular), then Jocular passes a negative reward to RL-Glue ($-1$ by default). This process is repeated until a valid action is obtained, when the reward is positive (2 by default). The only action that will be taken will be this last valid action. Once tha action is taken, this is sent to the GGP-Server.

The process goes on until the game is finished. At this moment, Jocular checks who is the winner and the total reward (methhod `getPayoffs`). Here, the RL-Glue is called again to make the last update according to the last state and action (method `RLGlue.RL_agent_end`). In the case that the player has lost, all the actions in all the previous states of the game are penalised with a $-2$ reward (this is done through the method `lose` of the RL-Glue agent).

The type and default values for the rewards is then as follows:

- Illegal action: $-1$ reward.
- Legal action: 2 reward.
- Win the game: None.
- Lose the game: $-2$ to all the actions in the game.

This choice of rewards (by default) has been chosen because we want the players to learn the rules as soon as possible (whenever an action is illegal an $-1$ reward is given until a correct action is issued, with a 2 reward).

As we see, all the integration has been done through Jocular. This means that the GGP-Server is kept as it was, and other players for GGP-Server can compete with the algorithms integrated in RL-Glue.

It is important to remark that in order to make Jocular work with RL-Glue, we need to select the "remote" player in the GGP-Server, since we want to use the same ports which the Jocular player has been launched with.

Finally, regarding the interface, it is important to recall that the results of the game are shown as (100) when a player wins, (0) when a player loses, and (50) for ties. These values have no other use but indicating who wins and should not be confused with the rewards seen above.

## 3 Installation and final remarks

The RL-GGP system connects RL-Glue with Jocular[5], a platform which creates players for the GGP (General Game Player) project, using the GGP server [4] but it also includes the RL algorithms developed by Hado van Hasselt into RL-glue.

The installation of the whole system can be done from:

`http://users.dsic.upv.es/~flip/RLGGP/`

where the part which just integrates GGP with RL-glue can be easily separated.

Finally, several things can be improved in RLGGP. For instance, we could change the rewards that are assigned to correct or incorrect moves using a configuration file. In addition, some other facilities to change which role each player takes could be simplified, as well as the interface.

## Acknowledgements

We thank the GGP community and the RL-community for developing the tools we integrate in RLGGP. This document has been extracted, translated and summarised from Benacloch's project by José Hernández-Orallo.

## References

[1] D. L. Dowe and A. R. Hajek. A computational extension to the Turing Test. *in Proceedings of the 4th Conference of the Australasian Cognitive Science Society, University of Newcastle, NSW, Australia*, 1997.

[2] D. L. Dowe and A. R. Hajek. A non-behavioural, computational extension to the Turing Test. In *Intl. Conf. on Computational Intelligence & multimedia applications (ICCIMA'98), Gippsland, Australia*, pages 101–106, 1998.

[3] D. L. Dowe and J. Hernandez-Orallo. IQ tests are not for machines, yet. *Intelligence*, 40(2):77–81, 2012.

[4] M. Genesereth, N. Love, and B. Pell. General game playing: Overview of the AAAI competition. *AI Magazine*, 26(2):62, 2005.

[5] D. Haley. Jocular. *http://games.stanford.edu/resources/reference/jocular/jocular.html*.

[6] J. Hernández-Orallo. Beyond the Turing Test. *J. Logic, Language & Information*, 9(4):447–466, 2000.

[7] J. Hernández-Orallo. Constructive reinforcement learning. *International Journal of Intelligent Systems*, 15(3):241–264, 2000.

[8] J. Hernández-Orallo. On the computational measurement of intelligence factors. In A. Meystel, editor, *Performance metrics for intelligent systems workshop*, pages 1–8. National Institute of Standards and Technology, Gaithersburg, MD, U.S.A., 2000.

[9] J. Hernández-Orallo. Thesis: Computational measures of information gain and reinforcement in inference processes. *AI Communications*, 13(1):49–50, 2000.

[10] J. Hernández-Orallo and D. L. Dowe. Measuring universal intelligence: Towards an anytime intelligence test. *Artificial Intelligence*, 174(18):1508 – 1539, 2010.

[11] J. Hernández-Orallo and N. Minaya-Collado. A formal definition of intelligence based on an intensional variant of Kolmogorov complexity. In *Proc. Intl Symposium of Engineering of Intelligent Systems (EIS'98)*, pages 146–163. ICSC Press, 1998.

[12] B. Hibbard. Adversarial sequence prediction. In *Proceeding of the 2008 conference on Artificial General Intelligence 2008: Proceedings of the First AGI Conference*, pages 399–403. IOS Press, 2008.

[13] J. Insa-Cabrera, D. L. Dowe, S. España-Cubillo, M. V. Hernández-Lloreda, and J. Hernández-Orallo. Comparing humans and AI agents. In J. Schmidhuber, K.R. Thórisson, and M. Looks, editors, *Artificial General Intelligence*, volume 6830, pages 122–132. LNAI, Springer, 2011.

[14] J. Insa-Cabrera, D. L. Dowe, and J. Hernandez-Orallo. Evaluating a reinforcement learning algorithm with a general intelligence test. In J.A. Moreno J.A. Lozano, J.A. Gamez, editor, *Current Topics in Artificial Intelligence. CAEPIA 2011*. LNAI Series 7023, Springer, 2011.

[15] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 1998.

[16] B. Tanner and Ad. White. RL-Glue : Language-independent software for reinforcement-learning experiments. *Journal of Machine Learning Research*, 10:2133–2136, September 2009.

[17] S. Whiteson, B. Tanner, and A. White. The Reinforcement Learning Competitions. *The AI magazine*, 31(2):81–94, 2010.