MASPY: A Python Framework for Developing BDI Agents with Reinforcement Learning

Demonstration Track

Alexandre L. L. Mellado
Federal University of Technology - Paraná (UTFPR)
Ponta Grossa, Brazil
mellado@alunos.utfpr.edu.br

Gabriel G. Neres
Federal University of Technology - Paraná (UTFPR)
Ponta Grossa, Brazil
neres@alunos.utfpr.edu.br

André P. Borges
Federal University of Technology - Paraná (UTFPR)
Ponta Grossa, Brazil
apborges@utfpr.edu.br

Rafael C. Cardoso
University of Aberdeen
Aberdeen, United Kingdom
rafael.cardoso@abdn.ac.uk

Gleifer V. Alves
Federal University of Technology - Paraná (UTFPR)
Ponta Grossa, Brazil
gleifer@utfpr.edu.br

ABSTRACT

MASPY is a Python framework for developing Belief-Desire-Intention (BDI) multi-agent systems that combines an AgentSpeak-like declarative language with Python's ecosystem. It supports communication, environmental interaction, and plan execution through a BDI reasoning cycle, and includes a reinforcement learning module that enables agents to adapt to trained environments.

KEYWORDS

BDI Agents, Multi-Agent Systems, Reinforcement Learning

ACM Reference Format:
Alexandre L. L. Mellado, Gabriel G. Neres, André P. Borges, Rafael C. Cardoso, and Gleifer V. Alves. 2026. MASPY: A Python Framework for Developing BDI Agents with Reinforcement Learning: Demonstration Track. In *Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), Paphos, Cyprus, May 25 – 29, 2026*, IFAAMAS, 3 pages. https://doi.org/10.65109/YIGW5980

1 INTRODUCTION AND RELATED WORK

Modern Multi-Agent System (MAS) frameworks streamline development by offering components, methods, and communication protocols for defining agent logic, enabling developers to focus on designing high-level behaviours and interactions. Integrating the Belief-Desire-Intention (BDI) model [4] remains impactful, offering a cognitive approach for modelling rational agency and decision-making in autonomous systems [1, 6, 15].

MASPY (Multi-Agent System for Python) [11] is a framework designed to streamline MAS development using the BDI model with Machine Learning (ML) capabilities. Its objective is to facilitate intuitive and flexible agent programming within a transparent and explainable reasoning paradigm. MASPY is implemented as a declarative language, adopting an AgentSpeak style in Python to design and manage general-purpose BDI-MAS.

Many frameworks support agent-based programming [6], such as Jason [2], Jadex [14], GOAL [10], 2APL [7], and Gwendolen [8], each emphasising different aspects of MAS development, such as scalability, verifiability, and adaptability. However, these languages still lack an intuitive way of integrating learning into their agents. The closest work is Jason-RL [3], which further enhances Jason's BDI reasoning with Reinforcement Learning (RL). Our work differs from theirs in that it uses Python, enabling broader integration with machine learning tools and libraries.

Frameworks in Python, such as SPADE [13], PADE [12], and Mesa [16], support distributed, communicative, and visual simulations, while BDIPython [5] and PROFETA [9] explore BDI agents. These frameworks either provide MAS infrastructure without BDI (e.g., SPADE, MESA), or BDI reasoning without a complete MAS execution environment (e.g., BDIPython, PROFETA and SPADE's plugin). MASPY integrates both, leveraging Python's ecosystem and accessibility for combining symbolic reasoning with machine learning tools.

This demonstration paper showcases how agents reason, communicate, and interact with the environment using learned behaviour, with system monitoring provided via a graphical user interface.

2 THE MASPY FRAMEWORK

The MASPY framework is a Python framework designed to make BDI-MAS development accessible and compatible with modern machine learning tools. MASPY is open-source and includes documentation and runnable examples.[1]

The framework consists of five classes: *Admin*, responsible for system configuration, execution, and logging; *Agent*, which manages beliefs, goals, and plan execution in a BDI reasoning cycle; *Environment*, which models the context in which agents perceive and act; *Communication*, enabling message exchange through communication channels; and *Learning*, integrating RL to provide agents with learnable actions in trained environments. Figure 1 shows, for

Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), C. Amato, L. Dennis, V. Mascardi, J. Thangarajah (eds.), May 25 – 29, 2026, Paphos, Cyprus. © 2026 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). https://doi.org/10.65109/YIGW5980

[1] MASPY WIKI: https://github.com/laca-is/MASPY/wiki/Navigation-on-2D-Grid/ (Accessed: 26/02/2026)

example, how these classes in a system with three agents, two environments, communication channels, and learning classes interact with each other.

MASPY includes an accompanying graphical interface that displays information about the system under execution. The interface shows agent intentions, exchanged messages, perceptions, and actions performed in the environment at each reasoning cycle. This helps users monitor the system in real-time and debug code.

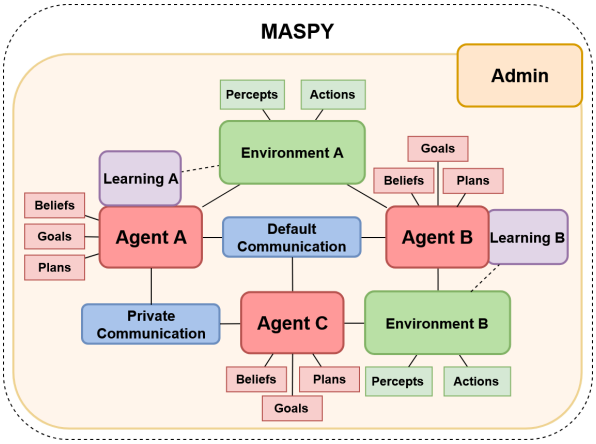

Figure 1: A MASPY system showcasing how classes interact.

3 CASE STUDY

We introduce a navigational scenario to demonstrate how MASPY integrates learning, perception, communication, and BDI reasoning with a unified Python framework. In this scenario, several agents are placed in a 2D grid with randomly generated obstacles and must locate a target position (Figure 2). Agents can move in four directions and perceive only their current position.

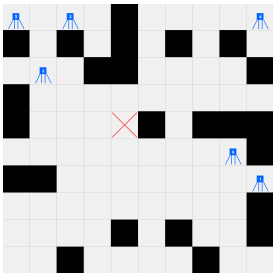

Figure 2: Navigational example interface using Pygame. Blue squares are agents, black squares are obstacles, and the red cross is the target position.

Before execution, each agent trains independently using its learning module to build a Q-table for the given map. This training produces only partial knowledge due to a limited number of training episodes (2000), so no single agent learns the full optimal path. Given unlimited training episodes, agents would eventually learn optimal policies, however, our goal was to show that partial training combined with symbolic reasoning can save training resources and still achieve optimal behaviour. This training takes place only before execution. During execution, the agents select actions using the best prediction from the trained model and update their internal map beliefs based on environmental perceptions, without modifying the trained module.

The agent's behaviour is driven by the two plans in Listing 1 (lines 1 and 13), both triggered by the goal to "move", but in different contexts. The plan `best_move` requires both the target and the agent's position on the map as context (i.e., symbolic, search-based movement), meanwhile `make_move` needs only the agent's position to execute (i.e., learning-based movement). Context is crucial in BDI reasoning, as the same goal can yield different intentions.

The code snippet highlights MASPY's reasoning cycle, environment interaction, use of the learned policies and communication. While agents do not know the target's location, they use the best action to move in the environment, shown in lines 16 and 17. They share information they discover, such as obstacles and updated paths (lines 8 and 19), and the target location (line 23), allowing them to collectively build the map of the grid. After discovering the location of the target, agents use an A* algorithm that accounts for incomplete map knowledge to find the best-known path to it, as shown in lines 5 and 6. Because the agent has incomplete knowledge, if a movement attempt fails while heading to the target, it reruns A* using the updated knowledge, including which paths have already failed.

```
1  @pl(gain, Goal("move"),
2      Belief("target", (Any,Any)) &
3      Belief("agt_pos", (Any,Any), "Map"))
4  def best_move(self, src, target, position):
5      direction = astar_explore(position, target)
6      self.move(direction)
7      self.perceive("Map") # Updates Map Perception
8      self.update_map() # Updates Map and Informs Others
9      if not self.has(Belief("arrived",(Any,Any),"Map")):
10         return False # Retries current event
11     self.stop_cycle()
12
13 @pl(gain,Goal("move"),
14     Belief("agt_pos",(Any,Any),"Map"))
15 def make_move(self, src, position):
16     policy = self.get_best_action("Map",(position,))
17     self.move(policy)
18     self.perceive("Map") # Updates Map Perception
19     self.update_map() # Updates Map and Informs Others
20     target = self.get(Belief("arrived",(Any,Any),"Map"))
21     if target is None:
22         return False # Retries current event
23     self.send(broadcast, tell, Belief("target",target))
```

Code Listing 1: Agent's Movement Plans

4 CONCLUSION AND FUTURE WORK

MASPY is a Python framework that provides declarative agent programming, communication protocol, environment modelling, and RL capabilities, making it easier to design explainable and adaptive agents within the Python ecosystem. The navigational example illustrated how a custom environment can be modelled, trained and combined with symbolic reasoning. For future work, we plan to continue expanding MASPY modules, explore distribution mechanisms, incorporate more complex learning, and add further capabilities to the graphical interface.

ACKNOWLEDGMENTS

This work is supported in part by CNPq/MCTI/FNDCT N° 22/2024 grant number 444568/2024-7, "Engineering Neuro-Symbolic Agents".

REFERENCES

[1] Rafael H Bordini, Amal El Fallah Seghrouchni, Koen Hindriks, Brian Logan, and Alessandro Ricci. 2020. Agent programming in the cognitive era. *Autonomous Agents and Multi-Agent Systems* 34 (2020), 1–31. https://doi.org/10.1007/s10458-020-09453-y

[2] Rafael H. Bordini, Jomi Fred Hübner, and Michael Wooldridge. 2007. *Programming Multi-Agent Systems in AgentSpeak Using Jason (Wiley Series in Agent Technology)*. John Wiley & Sons.

[3] Michael Bosello and Alessandro Ricci. 2019. From Programming Agents to Educating Agents - A Jason-Based Framework for Integrating Learning in the Development of Cognitive Agents. In *Engineering Multi-Agent Systems: 7th International Workshop, EMAS 2019, Montreal, QC, Canada, May 13-14, 2019, Revised Selected Papers* (Montreal, QC, Canada). Springer-Verlag, Berlin, Heidelberg, 175–194. https://doi.org/10.1007/978-3-030-51417-4_9

[4] Michael Bratman. 1987. *Intention, Plans, and Practical Reason.* Cambridge: Cambridge, MA: Harvard University Press.

[5] Paul Bremner, Louise A Dennis, Michael Fisher, and Alan F Winfield. 2019. On proactive, transparent, and verifiable ethical reasoning for robots. *Proc. IEEE* 107, 3 (2019), 541–561. https://doi.org/10.1109/JPROC.2019.2898267

[6] Rafael C. Cardoso and Angelo Ferrando. 2021. A Review of Agent-Based Programming for Multi-Agent Systems. *Computers* 10, 2 (Jan 2021), 16. https://doi.org/10.3390/computers10020016

[7] Mehdi Dastani. 2008. 2APL: a practical agent programming language. *Autonomous agents and multi-agent systems* 16 (2008), 214–248. https://doi.org/10.1007/s10458-008-9036-y

[8] Louise A. Dennis and Michael Fisher. 2023. *Verifiable Autonomous Systems: Using Rational Agents to Provide Assurance about Decisions Made by Machines*. Cambridge University Press. https://doi.org/10.1017/9781108755023

[9] Loris Fichera, Fabrizio Messina, Giuseppe Pappalardo, and Corrado Santoro. 2017. A Python framework for programming autonomous robots using a declarative approach. *Science of Computer Programming* 139 (2017), 36–55. https://doi.org/10.1016/j.scico.2017.01.003

[10] Koen V Hindriks. 2009. Programming rational agents in GOAL. In *Multi-agent programming: Languages, tools and applications*. Springer, 119–157. https://doi.org/10.1007/978-0-387-89299-3_4

[11] Alexandre Lizieri Leite Mellado, André Pinz Borges, and Gleifer Vaz Alves. 2025. MASPY: A Python-Based Framework for Developing BDI Multi-agent Systems. In *Advances in Practical Applications of Agents, Multi-Agent Systems, and Computational Social Science: The PAAMS Collection*. Springer Nature Switzerland, Cham, 216–227. https://doi.org/10.1007/978-3-032-07638-0_18

[12] Lucas Silveira Melo, Raimundo Furtado Sampaio, Ruth Pastôra Saraiva Leão, Giovanni Cordeiro Barroso, and José Roberto Bezerra. 2019. Python-based multi-agent platform for application on power grids. *International transactions on electrical energy systems* 29, 6 (2019). https://doi.org/10.1002/2050-7038.12012

[13] Javier Palanca, Andrés Terrasa, Vicente Julian, and Carlos Carrascosa. 2020. SPADE 3: Supporting the new generation of multi-agent systems. *IEEE Access* 8 (2020), 182537–182549. https://doi.org/10.1109/ACCESS.2020.3027357

[14] Alexander Pokahr, Lars Braubach, and Winfried Lamersdorf. 2003. Jadex: Implementing a bdi-infrastructure for jade agents. *EXP-in search of innovation (Special Issue on JADE)* 3, 3 (2003), 76–85.

[15] Lavindra de Silva, Felipe Meneguzzi, and Brian Logan. 2020. BDI Agent Architectures: A Survey. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*. International Joint Conferences on Artificial Intelligence Organization, 4914–4921. https://doi.org/10.24963/ijcai.2020/684

[16] Ewout ter Hoeven, Jan Kwakkel, Vincent Hess, Thomas Pike, Boyu Wang, Rht, and Jackie Kazil. 2025. Mesa 3: Agent-based modeling with Python in 2025. *Journal of Open Source Software* 10, 107 (2025), 7668. https://doi.org/10.21105/joss.07668