Adaptive Modular Agent Architecture for Hybrid Two-Level Reasoning

Dmitry Gnatyshak[1,2] [0000−0001−6779−6283], Sergio Álvarez-Napagao[1] [0000−0001−9946−9703], Julian Padget[3] [0000−0003−1314−2094], and Ulises Cortés[1] [0000−0003−0192−3096]

[1] Universitat Politècnica de Catalunya, Barcelona, Spain
[2] Keysight Technologies, Barcelona, Spain
[3] University of Bath, Bath, UK

Abstract. Despite the significant advances in subsymbolic artificial intelligence over the last decade, including novel Large Language Model (LLM) based methods, this technology comes with high, even prohibitive, development and application costs. Recent research suggests that instead of blindly increasing the model sizes and deploying larger numbers of better accelerators, there are benefits from a focus on the model structures and methods employed instead. An approach that has gained in popularity lately that addresses this is neurosymbolic reasoning. By focusing on various ways of combining subsymbolic, mostly neural network-based, computations with classical symbolic reasoning, it promises to alleviate the computation demands of pure deep learning approaches by guiding the learning process with symbolic knowledge. In this paper, we introduce the Modular Hybrid Agent Architecture (MHAgentA), a cognitive agent architecture, along with a Python framework that implements it. This architecture is designed to facilitate the prototyping and deployment of neurosymbolic agents that follow Kahneman's System 1, System 2 model. We provide a breakdown of a high-level view of the architecture and outline the technical details of its implementation. Moreover, the framework produces containerised multiprocessing-based agents, reducing the inherent complexity of deployment that similar systems usually face.

1 Introduction

The concept of an *agent* as an autonomous software program capable of pursuing goals on behalf of its owner is far from new. Envisioned in the era of symbolic artificial intelligence [30], agents have been extensively explored, with a number of classifications and even standards developed [26] formalising their potential capacities, structures, models, etc. Many *cognitive architectures* were developed, such as Soar [18], formalising potential building blocks of an intelligent agent and their internal structures facilitating the pursuit of goals. It was soon discovered, however, that the high computational complexity, as well as issues in translating noisy and imprecise real-world sensor data into symbolic formats, were prohibitive for its application in complex domains [36, 37].

Subsymbolic approaches have now become dominant in AI, trading off inherent explainability for generalisation capacity and, eventually, computational efficiency. As they evolved, the concept of agents was rediscovered, and new agent-based approaches

were established. For instance, with advancements in the field of (deep) reinforcement learning (RL, DRL), many complex problems were tackled with computational agents, with particular focus on various games of high complexity [21, 31, 32, 34, 23].

Furthermore, the recent advances with large language models (LLM) show that they can be sufficient to approximate intelligent agents [25] and even imitate symbolic reasoners [24]. However, the price for these advancements is not negligible with the inherent opacity of deep models and gargantuan time, financial, and environmental costs of training complex neural architectures [7].

Neurosymbolic approaches [17], striving to combine the explainability of symbolic methods with the generalisation of neural networks, hugely vary in combination ratios of these components [8]. With the neurosymbolic approach followed in this paper, we aim to address the high computational demands of both subsymbolic and symbolic approaches when applied to complex use cases by utilising adaptive learning during the agent runtime, as well as value-based reasoning.

To facilitate efficient experimentation while pursuing these research goals, we determined a need for a computationally lightweight, highly customisable modular architecture (and corresponding agent development framework) that can be adapted and modified quickly in accordance with evolving research objectives. This work resulted in MHAgentA — Modular Hybrid Agent Architecture — a modern cognitive architecture and a Python framework designed to facilitate the prototyping of neurosymbolic agents, which is the subject of this paper.

The rest of the paper is organised as follows. In section 2, we briefly explore related research. Section 3 follows with the details on the MHAgentA. Finally, in section 4, we conclude with a discussion of the scope of the work, its goals, and next steps.

2 Related work

A significant number of cognitive architectures have been proposed over the last 40 years, many inspired by the works of Allen Newell [22] and Marvin Minsky [20]. These works outlined an intelligent agent as a system of various components defining its capacities and behaviour, such as knowledge, language, perception, decision-making, etc.

One of the most famous cognitive architectures, Soar [18], is a classical example of a symbolic agent architecture. Other notable cognitive architectures include ACT-R [5] and LIDA [6], grounded in cognitive neuroscience and modelling their reasoning processes after human cognition, and a large set of architectures inspired by the belief-desire-intention reasoning model [27], such as PRS [14], JACK [10], dMARS [12], JADEX [9], and others. However, most of them were developed a long time ago, meaning that they lack easy inherent support for more recent optimisation and deployment techniques, as well as leaning more towards symbolic agents while posing technical challenges for extensive incorporation of modern deep learning-based subsystems.

On the subsymbolic side, there is a lot of research on DRL methods [21, 19, 29, 28]. Given enough computational and time resources (prohibitive amounts in complex real-world scenarios), they can find near-optimal solutions for many agent-based environments. Nevertheless, these costs and the inherent opacity of DRL agents' behavioural policies hinder their adoption in many areas.

The explainable reinforcement learning field aims to address the latter issue, but it faces its own set of problems, such as a lack of standardisation and high ambiguity in the definition of what is a (good) explanation with a promising research direction lying in extracting symbolic explanations from subsymbolic behavioural policies [13].

Large language models also seem to benefit from following the classical cognitive reasoning approaches. For instance, it was shown that a classical intelligent agent structure with its components imitated by LLMs could simulate high-level behaviour [25]. Moreover, it was shown that forcing LLM architectures to undergo internal "reasoning" cycles could vastly improve the output quality [24, 11].

Finally, there exist many works on neurosymbolic methods, with various types of approaches identified in this field, from using symbols to the benefit of the otherwise purely subsymbolic systems and vice versa to attempts to create true hybrid solutions [8, 4, 35]. The last is generally regarded as the most promising way forward [17] and is the approach we follow with MHAgentA.

3 MHAgentA: Modular Hybrid Agent Architecture

3.1 Requirements

As stated above, MHAgentA is a modular and customisable cognitive architecture inspired by Kahneman's System 1 – System 2 model of human mind [16, 15]. The separation into two systems implies, to a first approximation, a two-part architecture capturing a coarse division between fast and slow thinking. We also argue that the learning subsystem and the intrinsic value model are crucial for efficient *adaptive* agents capable of dynamically distributing the decision workload between the two systems.

The framework's main requirements were computational efficiency and ease of agent development and deployment. We aimed to achieve the former with multi-processing (especially given the current high availability of high-performance computing) and the latter by using a popular programming language with a low entry threshold, such as Python, and with containerisation.

3.2 Cognitive architecture

Like many other cognitive architectures, MHAgentA takes inspiration from the perception of the human mind, a complex system running many parallel processes. As stated above, we view it as two interacting systems: the faster, resource-efficient and reactive System 1, and the slower, more sophisticated, *conscious* System 2, which can generally take over control from System 1 as needed. Moreover, the separation of responsibilities between these two is not fixed: Kahneman states that while System 2 initially handles novel situations, it gradually relinquishes control to System 1 as the situation becomes more familiar. This ability to learn to adapt the workload separation between these two systems is the key feature we set as the main learn-term goal for this ongoing work.

MHAgentA's modularity and customisability allow for the creation of agents with varying capabilities implemented by a number of symbolic and subsymbolic techniques depending on the use case. Essentially, each of the provisioned module types is optional

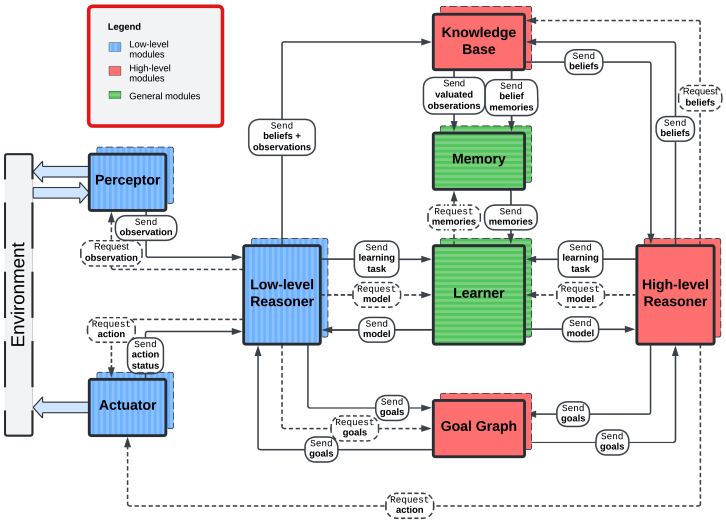

Fig. 1. MHAgentA modules and their interaction channels.

and represents a set of potential capabilities. However, for this section we will assume that the full set of module types defined below is used. The complete diagram of MHAgentA's module types and their interactions appears in Figure 1.

Each of these modules represents a *separate semi-autonomous process* running in parallel with other module processes and interacting with them to facilitate information processing and decision-making. Depending on the strictness of the agent and multi-agent system definitions and specifics of module implementations, a MHAgentA agent can be considered a multi-agent system itself.

The architecture does not specify the exact details of each module's implementation; instead, it focuses on the communication scheme (i.e., defining the internal information flow) between them. To this end, each module can be as complex or simple as needed. Moreover, it is possible to define several modules of the same type but with potentially different purposes and capabilities.

The following paragraphs describe the architecture's modules in more detail.

Perceptors: should be used as sensors, gathering data from the environment and forwarding it to a low-level reasoner for processing and (symbolic) belief extraction. This process can be automatic and continuous or triggered as a response to a request from a low-level reasoner. In the case of several observation types or modalities, although it is possible to implement a single perceptor to handle them all, it might be more efficient to have a set of specialised perceptors instead.

Actuators: these are agents' means to act upon the environment. Certain actions can be performed autonomously and continuously; however, actions performed in response to requests from reasoners are more common. Although actuators have the capacity to provide basic feedback on the action execution (e.g., failure or success of its execution from the internal agent perspective), the full consequences of the performed action should be observed by agent perceptors.

Low-level reasoners: analogues of System 1, they are conceptualised as fast subsymbolic reactive decision-makers. Each has easy access to raw observations via its connection to perceptors and actions via actuators. It receives its current goals from the goal graph module and works towards achieving or maintaining them. A low-level reasoner's additional responsibility crucial to the neurosymbolic reasoning capacity is processing the raw observations into sets of beliefs (alternatively performed inside the knowledge base, depending on the use case requirements and design choices).
Knowledge bases: represent collections of the agent's inherent and acquired knowledge of the world. They can include the use-case-specific ontology, sets of rules, current sets of beliefs, etc. We argue for the importance of including a value model of the agent, even as simple as a three-level partial order lattice of beliefs (e.g., like-neutral-dislike). This allows the agent to automatically assign numeric values to the observations, which in turn can function as rewards for reinforcement learning policies trained during the agent's execution. Finally, as mentioned above, the knowledge base can also process raw observations into their symbolic counterparts.
High-level reasoners: analogues of System 2, with the role of classical, sophisticated symbolic planners making strategic decisions on behalf of their agents while leaving the minute details of the plans' execution to their low-level counterparts. However, they do have a certain level of control over an agent's actions via goals and direct access to actuators. Finally, the intended symbolic implementation of high-level reasoners makes the strategic level of agents' decision-making inherently transparent, contributing to establishing trust and ensuring a level of explainability.
Goal graphs: used to store the plans, both active and inactive, and to facilitate the exchange of their execution status between the low and high-level reasoners. They are essentially storage buffers but with the capacity to utilise their semi-autonomy to process and combine existing plans, as well as automatically process partial stages of plan execution without taking computational resources away from either of the reasoners.
Memory: another independent storage module type managing the information that keeps track of past information. By default, this information is used primarily for learning, e.g., for a subsymbolic RL policy or a symbolic rule-based system.
Learners: general-purpose modules responsible for learning new behaviours based on the agents' experience at the runtime. Both high and low-level reasoners can communicate with them, allowing the agent to learn both subsymbolic and symbolic behaviours. The knowledge and the memory modules play a critical role here by projecting agent values onto beliefs and observations and then storing them for learning.

3.3 Implementation details

In addition to the cognitive architecture design, MHAgentA also comes with a direct implementation as a Python package, `mhagenta`, available through the PyPI repository[4]. It facilitates the implementation, deployment, log collection, and general runtime management of agent execution (via Orchestrator objects). Each agent is structured as a multi-process system of modules (one process per module) wrapped inside a virtual Docker container [1]. The internal module interaction is facilitated by the Rab-

[4] https://pypi.org/project/mhagenta/

bitMQ message-broker software [3] with a simple messaging server instantiated inside each container. Individual module execution uses `subprocess` and `asyncio` Python modules, as well as a custom priority queue for scheduling the execution of internal tasks, such as incoming message processing and ticks of the user-defined internal loop. Basic module coordination (e.g., synchronous start and finish) is done via a root controller sending high-priority commands through a dedicated broadcast channel. Beyond that, there is no internal mechanism ensuring synchronous execution in terms of time steps; it is up to the user to ensure that each function is properly sliced to avoid execution bottlenecks at the individual module level. The module states are saved after the agent stops, with an option to resume the execution from them. Finally, several special variants of the modules are provided, with more pending release. Among those are RabbitMQ-based Perceptor and Actuator classes and an environment template.

4 Discussion

This paper introduces the MHAgentA cognitive architecture and an agent framework developed as part of a PhD project. The final goal of the project is to test the capabilities of the neurosymbolic adaptive learning approach for creating efficient and effective agents for complex environments.

MHAgentA is designed as a tool for rapid prototyping and iterating over possible solutions for this problem. Its Python implementation, using containerisation and multiprocessing, proved to be sufficiently efficient when tested with various reinforcement learning environments of the `gymnasium` library [2], and is expected to scale efficiently with an increase in environment complexity. Furthermore, containerization and the use of RabbitMQ are expected to facilitate efficient MHAgentA agents deployment on HPC clusters, with tests to support this claim scheduled.

On the side of the default module instantiations, we plan to provide code examples of MHAgentA agents, both general and environment-specific, in the near future.

The ultimate goal, and the current stage of the work, is to finish the implementation and testing of a Python MHAgentA agent under the following requirements (as discussed above):

- DRL-based low-level reasoner and a symbolic planner high-level reasoner.
- Knowledge base combining the ontological representation of the environment in question with a value system over potential beliefs.
- Goal graph organised hierarchically with primitive actions at the bottom and complex plans at the top.
- Learner module capable of updating its policies to shift the responsibility border between the low and high-level reasoner higher on the goal graph.

We selected NeuralMMO [33] as the experiment environment and will extensively test the agent's capabilities to solve the list of tasks provided with NeuralMMO.

Further technical optimisations, especially regarding third-party tools like Docker and RabbitMQ, are planned as future work, along with an extensive exploration of MHAgentA agents in multi-agent settings.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Docker: Accelerated container application development, https://www.docker.com/, last accessed on 19/02/2025
2. Gymnasium documentation, https://gymnasium.farama.org/, last accessed on 19/02/2025
3. Rabbitmq: One broker to queue them all | rabbitmq, https://www.rabbitmq.com/, last accessed on 19/02/2025
4. Acharya, K., Raza, W., Dourado, C., Velasquez, A., Song, H.H.: Neurosymbolic reinforcement learning and planning: A survey. IEEE Transactions on Artificial Intelligence **5**, 1939–1953 (5 2024). https://doi.org/10.1109/TAI.2023.3311428
5. Anderson, J.R.: Rules of the Mind. Psychology Press (1993)
6. Baars, B.J., Franklin, S.: Consciousness is computational: The lida model of global workspace theory. International Journal of Machine Consciousness **1**, 23–32 (6 2009). https://doi.org/10.1142/S1793843009000050, https://philpapers.org/rec/BERCIC
7. Bender, E.M., Gebru, T., McMillan-Major, A., Shmitchell, S.: On the dangers of stochastic parrots: Can language models be too big? FAccT 2021 - Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency pp. 610–623 (3 2021). https://doi.org/10.1145/3442188.3445922, https://dl.acm.org/doi/10.1145/3442188.3445922
8. Bhuyan, B.P., Ramdane-Cherif, A., Tomar, R., Singh, T.P.: Neuro-symbolic artificial intelligence: a survey. Neural Computing and Applications **36**, 12809–12844 (7 2024). https://doi.org/10.1007/S00521-024-09960-Z/TABLES/12, https://link-springer-com.recursos.biblioteca.upc.edu/article/10.1007/s00521-024-09960-z
9. Braubach, L., Pokahr, A., Lamersdorf, W.: Jadex: A short overview. In: Main Conference Net. ObjectDays. pp. 195–207. AgentExpo (2004)
10. Busetta, P., Ronnquist, R., Hodgson, A., Lucas, A.: Jack intelligent agents - components for intelligent agents in java. AgentLink News Letter **2**, 2–5 (1999)
11. DeepSeek-AI, Guo, D., et al.: Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning (1 2025), https://arxiv.org/abs/2501.12948v1
12. D'Inverno, M., Luck, M., Georgeff, M., Kinny, D., Wooldridge, M.: The dmars architecture: A specification of the distributed multi-agent reasoning system. Autonomous Agents and Multi-Agent Systems **9**, 5–53 (7 2004). https://doi.org/10.1023/B:AGNT.0000019688.11109.19/METRICS, https://link.springer.com/article/10.1023/B:AGNT.0000019688.11109.19
13. Gimenez-Abalos, V., Alvarez-Napagao, S., Tormos, A., Cortés, U., Vázquez-Salceda, J.: Intention-aware policy graphs: answering what, how, and why in opaque agents (9 2024). https://doi.org/10.5281/zenodo.13862643, http://arxiv.org/abs/2409.19038 http://dx.doi.org/10.5281/zenodo.13862643
14. Ingrand, F.F., Georgeff, M.P., Rao, A.S.: An architecture for real-time reasoning and system control. IEEE Expert: Intelligent Systems and Their Applications **7**, 34–44 (12 1992). https://doi.org/10.1109/64.180407, https://dl.acm.org/doi/10.1109/64.180407
15. Kahneman, D.: Maps of bounded rationality: Psychology for behavioral economics. American Economic Review **93**, 1449–1475 (12 2003). https://doi.org/10.1257/000282803322655392
16. Kahneman, D.: Thinking, Fast and Slow. Farrar, Straus and Giroux (2011)
17. Kautz, H.: The third ai summer: Aaai robert s. engelmore memorial lecture. AI Magazine **43**, 105–125 (3 2022). https://doi.org/10.1002/aaai.12036, https://ojs.aaai.org/aimagazine/index.php/aimagazine/article/view/19122
18. Laird, J.E.: The Soar Cognitive Architecture. The MIT Press (8 2019)

19. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning. 4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings (9 2015), https://arxiv.org/abs/1509.02971v6
20. Minsky, M.: Society of mind. Simon and Schuster (1988)
21. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D.: Human-level control through deep reinforcement learning. Nature 2015 518:7540 **518**, 529–533 (2 2015). https://doi.org/10.1038/nature14236, https://www.nature.com/articles/nature14236
22. Newell, A.: Unified theories of cognition. Harvard University Press (1994)
23. OpenAI, Berner, C., et al.: Dota 2 with large scale deep reinforcement learning (12 2019), https://arxiv.org/abs/1912.06680v1
24. OpenAI, Hurst, A., et al.: Gpt-4o system card (10 2024), https://arxiv.org/abs/2410.21276v1
25. Park, J.S., O'Brien, J., Cai, C.J., Morris, M.R., Liang, P., Bernstein, M.S.: Generative agents: Interactive simulacra of human behavior. UIST 2023 - Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology (10 2023). https://doi.org/10.1145/3586183.3606763/SUPPL_FILE/3606763.ZIP, https://dl.acm.org/doi/10.1145/3586183.3606763
26. Poslad, S.: Specifying protocols for multi-agent systems interaction. ACM Transactions on Autonomous and Adaptive Systems (TAAS) **2** (11 2007). https://doi.org/10.1145/1293731.1293735, https://dl.acm.org/doi/10.1145/1293731.1293735
27. Rao, A., Georgeff, M.: Bdi agents: From theory to practice. In: Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95). pp. 312–319 (1995)
28. Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., Lillicrap, T., Silver, D.: Mastering atari, go, chess and shogi by planning with a learned model. Nature 2020 588:7839 **588**, 604–609 (12 2020). https://doi.org/10.1038/s41586-020-03051-4, https://www.nature.com/articles/s41586-020-03051-4
29. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Openai, O.K.: Proximal policy optimization algorithms (7 2017), https://arxiv.org/abs/1707.06347v2
30. Shoham, Y.: Agento: a simple agent language and its interpreter. In: Proceedings of the Ninth National Conference on Artificial Intelligence - Volume 2. pp. 704–709. AAAI Press (1991)
31. Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., Hassabis, D.: A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. Science **362**, 1140–1144 (12 2018), https://www.science.org/doi/10.1126/science.aar6404
32. Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., Driessche, G.V.D., Graepel, T., Hassabis, D.: Mastering the game of go without human knowledge. Nature **550**, 354–359 (2017). https://doi.org/10.1038/nature24270, http://dx.doi.org/10.1038/nature24270
33. Suarez, J., Du, Y., Isola, P., Mordatch, I.: Neural mmo: A massively multiagent game environment for training and evaluating intelligent agents. arXiv preprint arXiv:1903.00784 (2019)
34. Vinyals, O., Babuschkin, I., Czarnecki, W.M., Mathieu, M., Dudzik, A., Chung, J., Choi, D.H., Powell, R., Ewalds, T., Georgiev, P., Oh, J., Horgan, D., Kroiss, M., Danihelka, I., Huang, A., Sifre, L., Cai, T., Agapiou, J.P., Jaderberg, M., Vezhnevets, A.S., Leblond, R., Pohlen, T., Dalibard, V., Budden, D., Sulsky, Y., Molloy, J., Paine, T.L.,

Gulcehre, C., Wang, Z., Pfaff, T., Wu, Y., Ring, R., Yogatama, D., Wünsch, D., McKinney, K., Smith, O., Schaul, T., Lillicrap, T., Kavukcuoglu, K., Hassabis, D., Apps, C., Silver, D.: Grandmaster level in starcraft ii using multi-agent reinforcement learning. Nature 2019 575:7782 **575**, 350–354 (10 2019). https://doi.org/10.1038/s41586-019-1724-z, https://www.nature.com/articles/s41586-019-1724-z

35. Wang, W., Yang, Y., Wu, F.: Towards data-and knowledge-driven ai: A survey on neuro-symbolic computing. IEEE Transactions on Pattern Analysis and Machine Intelligence (2024). https://doi.org/10.1109/TPAMI.2024.3483273
36. Weiss, G.: Multiagent systems: a modern approach to distributed artificial intelligence (1999)
37. Wooldridge, M.: An introduction to multiagent systems. John Wiley & Sons (2009)