Linked Data MASE - A Maze-Based Multi-Agent Systems Environment for Testing and Visualizing Hypermedia Agents

Stefan Meier[0009−0006−4304−876X], Jérémy Lemée[0000−0003−0828−1188], Danai Vachtsevanou[0000−0002−6697−0427], and Simon Mayer[0000−0001−6367−3454]

University of St. Gallen, St. Gallen, Switzerland
stefan.meier@student.unisg.ch,
{jeremy.lemee,danai.vachtsevanou,simon.mayer}@unisg.ch

Abstract. As autonomous agents increasingly operate on the Web, researchers need reusable testbeds for controlled experimentation. Linked Data (LD) environments support uniform hypermedia interaction and have been used for the development of LD-based maze scenarios, suitable for studying diverse agent architectures. We present MASE, a lightweight LD-based maze environment customizable through SPARQL queries to support diverse evaluation requirements across heterogeneous agent architectures and experimental setups. MASE remedies limitations of existing implementations by enforcing embodiment- and interaction-related constraints, attributing each state change to the request and rules that caused it, and providing an event-driven visualization to support development and reduce entry barriers.

Keywords: Agent Environments · Hypermedia Multi-Agent Systems · Linked Data

1 Introduction

Autonomous agents are increasingly joining, navigating, and acting on the Web, exploiting its resource-rich, long-lived, world-scale environment enabled by mature Web standards. While designing agents that access Web tools and information sources offers benefits, *Hypermedia Multi-Agent Systems (hMAS)* [10] take a step further: hMAS are designed to inherit the openness, longevity, and scalability of the Web by aligning with its architectural principles and leveraging W3C standards such as the Linked Data Platform [23] and the Web of Things (WoT) Architecture [16]. For example, hMAS subsume MAS whose environments are represented as named RDF graphs and manipulated through well-defined operations, allowing for navigation based on the Linked Data (LD) principles (e.g., [4,9,11]).

To test and evaluate such Web-based MAS without the complexity and limited accessibility of real-world deployments, LD environments are frequently used as testbeds [5]. These prominently include *maze-like grid worlds* [1,22] that provide lightweight, controllable settings for prototyping and comparing diverse

agent architectures. For example, they have been used to study agents integrating elements from rule-based [22,21], BDI [3,19,20], predictive AI [3], stigmergy-based [22,19] and Large Language Model (LLM)-based architectures [21]. This diversity arises both at the level of MAS situated by heterogeneous agents, and at the level of individual agents, which combine elements from multiple architectural paradigms, therefore, highlighting the relevance of LD-based testbeds for both hybrid MAS and hybrid agent architectures, while the maze analogy offers a simple yet appropriate setting for Web navigation. Such testbeds also enable the investigation of hybrid agent architectures that combine the flexibility of LLM agents with well-formalized cognitive design patterns (e.g., [14,15]), and can mitigate the cost, controllability, and uncertainty issues of LLM-centric environments [6,7,18].

However, current implementations for LD testbeds have limitations that prevent their broader applicability in this context. This includes insufficient consideration of agent embodiment, therefore, preventing studies on access control constraints, physical constraints, agent observability, and exploitation of environmental interaction guidance through HATEOAS (cf. [13]).[1] Additionally, they do not track action causality, complicating debugging; and they offer only limited environment customizability, restricting experiment design.

To remedy these limitations, we present *MASE, a LD Multi-Agent Systems Maze Environment*[2], which builds on existing LD testbed approaches and extends them with (i) an extensible environment model that supports the design of custom types of interaction, (ii) constraints that enforce embodiment of agents in the maze, and (iii) an event-driven visualization of changes in the maze state.

2 Related Work

Maze scenarios have proven to be accessible across heterogeneous agent types, including reactive agents, language agents [21], BDI agents [20], humans [1]. In the context of LD environments, the Web can be viewed as a graph of resources (named RDF graphs) connected through links and manipulated through HTTP operations [9]. Maze environments instantiate this abstraction by modeling traversable connections between cells as RDF links, analogous to hyperlinks, and by exposing navigation and interaction opportunities as hypermedia affordances [2,4].

Building on this analogy, [21] implements a LD testbed by integrating the Graph Store and API infrastructure from the *Building on Linked Data* (BOLD) framework [17] with the scenario of the maze ontology introduced in [1]. The resulting environment simulates 2D mazes as collections of cells, where each cell is a resource described as a named RDF graph. Agents navigate by issuing HTTP `GET` requests to exploit hypermedia affordances that represent connections between adjacent cells following cardinal directions [1]. The shortest path from start to exit is signified by RDF triples that annotate the relevant connections

[1] For further relevant aspects to consider with respect to agent embodiment, cf. [8].
[2] Available at `https://github.com/stefanmhsg/mase`

as“green”. In addition, connections are controlled through key-lock mechanisms representing state-changing affordances: RDF triples that model keys may be discovered in a cell's graph and can be added to a locked cell's graph through HTTP `POST` requests; this unlocks the lock and reveals a connection to a neighboring cell (cf. [21]). Condition-action rules are executed in a continuous loop to materialize changes after a matching key was added to a locked cell.

In contrast, BOLD provides virtual time increments that trigger environment updates since it supports benchmark runs over a reproducible sequence of discrete dataset states [17]. Both approaches (i.e., [21] as well as [17]) decouple environment state changes from individual agent requests, although request-triggered state changes are characteristic of real Web environments (and specifically of WoT environments) and important for studying agent interaction and navigation under genuine asynchronous conditions. In addition, navigation via HTTP `GET` does not allow changes in the state of the environment [12]. Therefore, such approaches cannot enforce embodiment-related constraints, which are central in the MAS field [8]. Finally, the potential of maze scenarios to visualize interaction and navigation effects—thereby supporting inspection and debugging for iterative Web agent development—has not been realized yet.

3 MASE: Linked Data Maze for Hypermedia MAS

The *Linked Data Multi-Agent Systems Maze Environment (MASE)* retains the core idea and principles of the LD maze abstraction and existing implementations (see Section 2). It loads a set of named RDF graphs into a mutable repository and exposes HTTP operations for creating, accessing, modifying, and deleting these graphs. Each graph corresponds to a cell in a 2D maze with traversable connections between adjacent cells represented in RDF. From [21], MASE retains the key-lock mechanism as described in Section 2 as well as the underlying maze datasets with signifiers annotating the ideal route. MASE provides an additional feature for introducing persistent stigmergic markers into the environment that denote the number of times a connection has been used (cf. [24]). After initialization of the environment and during processing of agent requests, MASE executes SPARQL Update queries to evolve the environment based on predefined rules. Additional SPARQL queries can be defined to extend the environment dynamics, for instance, allowing the environment to evolve in response to the effects of agents' operations. For example, Listing 1.1 in Appendix A shows the rule for adding stigmergic markers. Following this model, the maze can be further extended to accommodate scenarios from different research communities.

To interact with MASE, agents issue an HTTP `GET` request to perceive the entry-point cell and discover navigation options. MASE enforces embodiment-related constraints on the server side by validating HTTP requests from agents: This is done by embodying agents as RDF triples in the graph of the cell that corresponds to their current location. Agents can then perform HTTP `GET` and `POST` operations on resources corresponding to their current location, with the only requirement that agents identify themselves by providing their name via

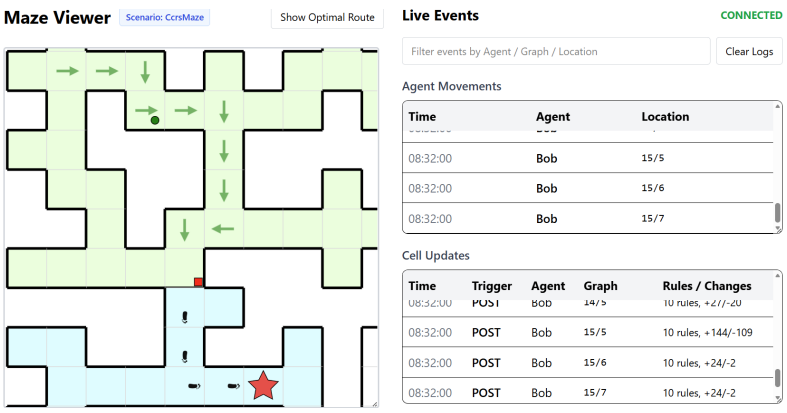

Fig. 1. MASE Viewer with current maze state (left) and event log (right).

the authorization header. To navigate, agents `POST` an RDF triple to their target cell—such access requests are validated by the MASE access control mechanism by checking whether the claimed source cell matches the agent's current location and whether a traversable connection exists in the maze. If the request is accepted, MASE updates the agent's location during rule execution and the agent becomes embodied in the target cell; the HTTP call returns either success or failure to the agent. In contrast to time-driven or loop-based simulations, MASE executes environment updates as part of request processing, after a request has been validated and before a response is returned; the execution model hence corresponds to typical Web service behavior, while state changes are immediately perceivable also to agents operating with very short reasoning cycles. State changes in the environment can be causally attributed to agent interactions, which enables precise analysis of agent behavior and interaction effects in multi-agent settings.

MASE provides a front end for visualizing the maze with event-driven updates received via WebSocket. It displays the positions of agents and other items in the maze. MASE introduces a custom user interface vocabulary to define the visual appearance of items in RDF. Style information is provided as RDF triples within the dataset and can be changed during rule execution as well, enabling new items to be displayed without the need to adjust the front end or event handling. An event log lists all transactions performed on the dataset with an expandable view to see which RDF triples have been added or removed based on which HTTP request or SPARQL query. The visualization of MASE is shown in Figure 1. We think this form of visualization enables proximal use of the environment, for example in teaching settings, where agent behavior and interaction effects can be observed directly. Finally, we note that MASE provides an environment without predefined performance metrics; hence, it is not a benchmark by itself. Nevertheless, its request-based execution model and event log enable the measurement of a variety of evaluation metrics, including success/failure of reaching the goal cell, number of requests, successful/failed moves, number of unlocked cells, path length, and elapsed time.

References

1. Amundsen, M.: Building Hypermedia APIs with HTML5 and Node. O'Reilly Media, Inc., CA USA (2011)
2. Bandini, S., Manzoni, S., Vizzari, G.: Web Sites as Agents' Environments: General Framework and Applications. In: Weyns, D., Van Dyke Parunak, H., Michel, F. (eds.) Environments for Multi-Agent Systems II. pp. 235–250. Springer, Berlin, Heidelberg (2006). https://doi.org/10.1007/11678809_14
3. Beaumont, K., O'Neill, E., Bermeo, N.V., Collier, R.W.: Collaborative route finding in semantic mazes. In: ATAC@ ISWC. pp. 13–18 (2021)
4. Bizer, C., Heath, T., Berners-Lee, T.: Linked Data - The Story So Far. In: Linking the World's Information: Essays on Tim Berners-Lee's Invention of the World Wide Web, pp. 115–143. Association for Computing Machinery, New York, NY, USA (2023), https://doi.org/10.1145/3591366.3591378
5. Boissier, O., Ciortea, A., Harth, A., Ricci, A., Vachtsevanou, D.: Agents on the Web (Dagstuhl Seminar 23081). Dagstuhl Reports **13**(2), 71–162 (2023). https://doi.org/10.4230/DagRep.13.2.71
6. Botti, V.: Agentic AI and Multiagentic: Are We Reinventing the Wheel? (2025). https://doi.org/10.48550/arXiv.2506.01463
7. Bădică, C., Bădică, A., Ganzha, M., Ivanović, M., Paprzycki, M., Selişteanu, D., Wrona, Z.: Contemporary Agent Technology: LLM-Driven Advancements vs Classic Multi-Agent Systems (2025). https://doi.org/10.48550/arXiv.2509.02515
8. Castellucci, M., Burattini, S., Ciortea, A., Lemée, J., Vachtsevanou, D., Ricci, A., Mayer, S.: Towards agents' embodiment in hypermedia multi-agent systems. In: European Conference on Multi-Agent Systems. pp. 361–381. Springer (2024). https://doi.org/10.1007/978-3-031-93930-3_21
9. Charpenay, V., Käfer, T., Harth, A.: A Unifying Framework for Agency in Hypermedia Environments. In: Alechina, N., Baldoni, M., Logan, B. (eds.) Engineering Multi-Agent Systems. pp. 42–61. Springer International Publishing, Cham (2022). https://doi.org/10.1007/978-3-030-97457-2_3
10. Ciortea, A., Boissier, O., Ricci, A.: Engineering world-wide multi-agent systems with hypermedia. In: Weyns, D., Mascardi, V., Ricci, A. (eds.) Engineering Multi-Agent Systems. EMAS 2018. Lecture Notes in Computer Science. vol. 11375, pp. 285–301. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-25693-7_15
11. Ciortea, A., Mayer, S., Gandon, F., Boissier, O., Ricci, A., Zimmermann, A.: A Decade in Hindsight: The Missing Bridge Between Multi-Agent Systems and the World Wide Web. In: Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems. pp. 1659–1663. AAMAS '19, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC (2019)
12. Fielding, R., Nottingham, M., Reschke, J.: HTTP Semantics. RFC 9110 (Proposed Standard) (2022), https://www.rfc-editor.org/rfc/rfc9110
13. Fielding, R.T., Taylor, R.N.: Principled design of the modern web architecture. ACM Trans. Internet Technol. **2**(2), 115–150 (2002). https://doi.org/10.1145/514183.514185
14. Gatti, A., Mascardi, V., Ferrando, A.: ChatBDI: Think BDI, talk LLM. In: Vorobeychik, Y., Das, S., Nowé, A. (eds.) 24th international conference on autonomous agents and multiagent systems (AAMAS 2025). International Foundation for Autonomous Agents and Multiagent Systems (2025)

15. Ichida, A.Y., Meneguzzi, F., Cardoso, R.C.: BDI agents in natural language environments. In: Proceedings of the 23rd international conference on autonomous agents and multiagent systems. pp. 880–888. AAMAS 2024, IFAAMAS, Auckland, New Zealand (2024)
16. Kovatsch, M., Matsukura, R., Lagally, M., Kawaguchi, T., Toumura, K., Kajimoto, K.: Web of Things (WoT) Architecture (2020), https://www.w3.org/TR/wot-architecture10/
17. Käfer, T., Charpenay, V., Harth, A.: BOLD: A Benchmark for Linked Data User Agents and a Simulation Framework for Dynamic Linked Data Environments (2023). https://doi.org/10.48550/arXiv.2307.09114
18. La Malfa, E., La Malfa, G., Marro, S., Zhang, J.M., Black, E., Luck, M., Torr, P., Wooldridge, M.: Large language models miss the multi-agent mark. arXiv preprint arXiv:2505.21298 (2025)
19. Lemée, J., Vachtsevanou, D., Mayer, S., Ciortea, A.: Signifiers for conveying and exploiting affordances: From Human-Computer Interaction to Multi-Agent Systems. Annals of Mathematics and Artificial Intelligence **92**, 815–835 (2024). https://doi.org/10.1007/s10472-024-09938-6
20. Saffaf, N., Charpenay, V.: Crawl into the Dungeon with Hypermedia Agents. In: Käfer, T., Harth, A., Ciortea, A., Charpenay, V. (eds.) Proceedings of the All the Agents Challenge (ATAC 2021). CEUR Workshop Proceedings, vol. 3111, pp. 19–24. CEUR, Virtual Event, New York (Oct 2021), https://ceur-ws.org/Vol-3111/#short3
21. Schmid, S., Freund, M., Harth, A.: Adaptive Planning on the Web: Using LLMs and Affordances for Web Agents. In: Tiwari, S., Villazón-Terrazas, B., Ortiz-Rodríguez, F., Sahri, S. (eds.) Knowledge Graphs and Semantic Web. pp. 93–108. Springer Nature Switzerland, Cham (2025). https://doi.org/10.1007/978-3-031-81221-7_7
22. Schmid, S., Schraudner, D., Harth, A.: MOSAIK: An Agent-Based Decentralized Control System with Stigmergy for a Transportation Scenario. In: Pesquita, C., Jimenez-Ruiz, E., McCusker, J., Faria, D., Dragoni, M., Dimou, A., Troncy, R., Hertling, S. (eds.) The Semantic Web. pp. 697–714. Springer Nature Switzerland, Cham (2023). https://doi.org/10.1007/978-3-031-33455-9_41
23. Speicher, S., Arwe, J., Malhotra, A.: Linked Data Platform 1.0 (2015), https://www.w3.org/TR/ldp/
24. Spieldenner, T., Chelli, M.: Linked Data as Medium for Stigmergy-based Optimization and Coordination. In: Fill, H.G., van Sinderen, M., Maciaszek, L.A. (eds.) Software Technologies. pp. 1–23. Springer International Publishing, Cham (2022). https://doi.org/10.1007/978-3-031-11513-4_1

A Appendix

```
PREFIX dyn: <https://paul.ti.rw.fau.de/~am52etar/dynmaze/dynmaze#>
PREFIX maze: <https://kaefer3000.github.io/2021-02-dagstuhl/vocab#>
PREFIX sm:   <https://example.org/stigmark#>

INSERT {
    # Origin Graph: Stigmergy marker creation
    GRAPH ?oldGraph {
        ?oldGraph sm:hasMarker ?newMarker .
        ?newMarker a sm:Marker .
        ?newMarker sm:contains _:node .
        _:node sm:refersTo ?targetGraph .
        _:node sm:quantitative 1 .
    }
}
WHERE {
    # the move event
    GRAPH ?targetGraph {
        ?agent dyn:entersFrom ?oldGraph .
    }

    # find the cell the agent is currently in
    GRAPH ?oldGraph {
        ?oldGraph maze:contains ?agent .

        # ensure marker does NOT already exist
        FILTER NOT EXISTS { ?oldGraph sm:hasMarker ?anyMarker }
    }

    # create new marker IRI
    BIND(IRI(CONCAT(STR(?oldGraph), "#marker")) AS ?newMarker)
}
```

Listing 1.1. Example of a SPARQL Query that inserts an initial stigmergic marker when the first agent leaves a cell.