LightJason, a Highly Scalable and Concurrent Agent Framework: Overview and Application - Demonstration

Malte Aschermann, Sophie Dennisen, Philipp Kraus, Jörg P. Müller

Abstract. Multiagent systems (MAS) provide useful abstractions for modelling and simulating complex socio-technical systems. However, existing open-source platforms suffer from serious issues including limited scalability, inflexible software architecture, and poor support for web deployment. This demo presents an overview and an application of *LightJason*, a highly scalable Java-based platform for agent-oriented programming (AOP) and simulation. We outline the architectural features of *LightJason* and showcase its applicability using an example of a browser-based web application implementing a traffic serious game devised to teach an interdisciplinary student team in MAS and AOP.
Demonstration video: `https://vimeo.com/lightjason/aamas2018`

1. Introduction

The modelling, implementation, and simulation of socio-technical systems (STS) [**Sin13**] is a major challenge for computer science. STS are systems such as Smart Cities in which heterogeneous human and automated entities are embedded in large-scale dynamic environments, form organisations, and act and interact in complex ways. Modelling STS requires appropriate models and mechanisms, such as micro-meso-macro architecture [**SBP12**], normative control, computational mechanism design [**SHR$^+$17**] and social choice [**DM15**], or game-theoretic models. From a computing perspective, dealing with STS imposes hard requirements on underlying platforms and software frameworks in terms of software quality, scalability, flexibility of software architecture, the ability to integrate with third-party software, and to support web deployment and execution on cloud/high-performance computing (HPC) platforms. MAS provide useful abstractions for modelling and simulating complex STS. However, existing open-source platforms like *Jason* [**BHW07, ZRH16**], *GOAL* [**HDBVDHM00**] or *Jade* [**BCG07**] fall short of satisfying the abovementioned requirements: Flaws in algorithmic design slow down performance; inflexible software architectures (e.g. built-in proprietary runtime systems) make integration with other systems (e.g. traffic simulators [**BRM$^+$09, KHRW02**]) hard, and are mostly not geared to support web deployment and web-based visualisation of application, and to use state-of-the-art HPC/cloud platforms. For our findings on algorithmic and architectural comparison, we refer to [**AKMS18, AKM17, AKM16**]. In this demo, we present an overview and an application of *LightJason*, a concurrent Belief-Desire-Intention (BDI) multi-agent framework for creating multi-agent systems with Java. *LightJason* has been inspired by *Jason* and *AgentSpeak(L)* [**Rao96**]; however, it is based

Key words and phrases. Agent-oriented programming and simulation; BDI modelling; agent platform; traffic simulation; scalability.

on a newly developed software architecture and, compared to *Jason*, proposes a new concurrent semantics of the agent cycle and a very efficient implementation of agent perception. Furthermore, the logic programming language *AgentSpeak(L++)* used in *LightJason* provides considerable extensions to *AgentSpeak(L)*, e.g. lambda-expressions, multi-plan and -rule definition, explicit repair actions, multi-variable assignments, parallel execution and thread-safe variables. *LightJason* offers a modular runtime system. By using object-oriented concepts and Java generics, tailored runtime instances can be created for an application, and *LightJason* can be easily integrated into third-party applications.

For further background/technical information and practical examples for multi-agent applications of *LightJason* we refer to [**AKM17**] and the documentation at `http://lightjason.org`.

2. Speeding Game Application

A LightJason-based browser application for teaching MAS. In this demo, we focus on how *LightJason* supports integrating agent technology into state-of-the-art browser-based web applications, which is difficult to handle with existing agent frameworks, which are mostly stand-alone with "hard-wired" integrated development environments (IDEs), runtime (RT), graphical user-interface (GUI) and source editor. In *LightJason*, these components are modular, which makes the framework lightweight, adaptable and easy to integrate.

Scenario. Our use-case model describes a highway traffic scenario which is used to teach students multiagent system (MAS) and AOP. The actors (vehicles, road segments with speed limits, traffic rule enforcement) are modelled as BDI agents. For details on how we modelled the scenario via ASL++ scripts, see `https://git.io/vxh0g`. In this demo, students have the task to write an agent program to steer a specific vehicle through a fixed-length road (four lanes with two directions) with changing speed limits and a black-box traffic rule enforcement (e.g., speed cameras that will be activated using some enforcement strategy unknown to the students).

Vehicle agents can perceive information about *segments*, *current speed*, *allowed speed* and an estimation of expected *penalties*. If and when penalties are imposed is also modelled by means of agents, governing *areas* of varying traffic limits and the *environment*. A well-performing vehicle driving strategy should strive after the user optimum, where the utility function reflects the goals to minimise travel time *and* penalty for speeding, while avoiding collisions with other vehicles.

Technical realisation. We chose a browser-based application (Web-UI) with a web server for the simulation backend and an interface for visualisation. In *LightJason* BDI logic is programmed as scripts using a code editor, and translated into Java objects which are executed by the RT. For visualising the current state of the environment and agents, we used the *Phaser*[1] game engine. Figure 1 illustrates the architecture of the resulting web application.

The *Speeding Game* application consists of a *JAR-bundle* of programs with all necessary components and a web-based GUI for modelling the agents and accessing lecture material (see Figure 1).[2] The RT executes the agents concurrently and in a scalable manner. To give feedback to the users regarding the performance of their strategies, penalty charts (Figure 2b) are shown via *Chart.js*[3]. *Ajax* and *WebSockets* are used for the communication between browser frontend and Java backend. The web server component uses *Jetty* with a *Servlet* and *WebSocket* endpoint for the web server and *Bootstrap* for the browser interface.

---

[1] `https://phaser.io/`

[2] For details and source code, see `https://git.io/vATPU` and `https://git.io/vATPq`.

[3] `http://www.chartjs.org/`

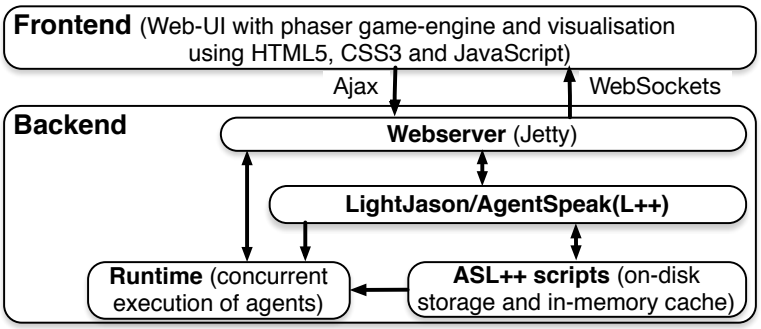


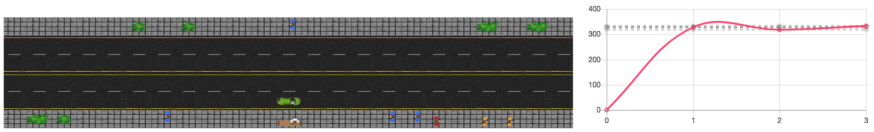
FIGURE 1. Architecture of the *Speeding Game* application

3. Evaluation and conclusions

A first evaluation of the *Speeding Game* application was carried out in a small-scale teaching study. In summer 2017, the Ph.D. students of the interdisciplinary Research Training Group *SocialCars*[4] attended a series of two-day simulation labs as part of their initial qualification programme. The students' background was very interdisciplinary, covering the areas of psychology, informatics, traffic engineering, operations research, geoinformatics, and communication technology. The purpose of the simulation labs was to increase the mutual understanding of research fields. Each lab contained a mix of theoretical lectures and practical hands-on workshops. In this context, we had the opportunity to present and evaluate the *Speeding Game* application in an AOP lab for the abovementioned scenario. The objectives of the lab were twofold: *1) Give an overview of MAS with a practical introduction to MAS models and frameworks, while 2) taking various degrees of initial knowledge and skills regarding programming in general and AOP in particular into account.* The main goal for the participants was to learn AOP for traffic simulations by completing three main assignments:

(1) Get familiar with the *LightJason* programming language *AgentSpeak(L++)*
(2) Understand the execution behaviour of agents by analysing and trying out a given baseline strategy based on an extended Nagel-Schreckenberg model [**NS92**]
(3) Create a new and competitive driving strategy

At the end of the lab, students submitted their results, the best strategies were determined in a contest, and prizes were awarded. The didactic concept of the lab followed the flipped-classroom (FC) [**Tuc12**] method, such that the PhD students could put their theoretical knowledge of accompanying lectures into practice. Lectures and exercises alternated over the two-day workshop.



(A) Scenario with road segments (B) Penalty statistics

FIGURE 2. Excerpts of simulation web interface

[4] `https://socialcars.org/`

While completing their assignments, students were using the *Speeding Game* web application to create, evaluate, and iteratively optimise vehicle driving strategies. At the end of each simulation run, key performance statistics (driving time, penalty) was shown and the students could further improve their strategy. The chart displayed in Figure 2b shows the variation of the collected penalty value with the 0.25 and 0.75 quantile values over all runs. Also, the theoretical course materials, and the reference documentation of the *AgentSpeak(L++)* programming language are integrated into the *Speeding Game* application. This allows students to easily access required resources at any time while working on their assignments. In order to evaluate satisfaction and learning success of the lab, participants were asked to fill in a short survey[5]. The results indicate that the large majority of the participants has benefited from the course and that the objective of teaching the interdisciplinary set of students the main concepts of the AOP has been achieved well (given the expectations one can have to a two-day course). Over 50% thought that *MAS modelling can be integrated in their own research* and 100% agreed to the statement that *it is useful for building complex simulations*. From a technical point of view, the idea to build a web application with a browser interface was a good choice. Participants can use the tool without any installation process. The browser interface was intuitive and easy-to-use. Using the built-in IDE, modifications to the BDI agent scripts can be made in the web application and compiled on-the-fly.

To conclude, in creating and using the *Speeding Game*, we largely benefited from the use of *LightJason* and the architectural design choices described in Section 2 above, i.e. integrating a scenario visualisation and IDE elements for AOP and providing easy access to tasks and information.

Acknowledgements. We thank Ehsan Tatasadi for creating the web interface.

References

[AKM16] Malte Aschermann, Philipp Kraus, and Jörg P. Müller, *Slides - lightjason: A bdi framework inspired by jason*, 2016.

[AKM17] ———, *LightJason: A BDI Framework inspired by Jason*, Multi-Agent Systems and Agreement Technologies: EUMAS 2016 and AT 2016, Lecture Notes in Computer Science, vol. 10207, Springer International Publishing, 2017, pp. 58–66.

[AKMS18] Sebastian Albert, Philipp Kraus, Jörg P. Müller, and Anita Schöbel, *Passenger-induced delay propagation: Agent-based simulation of passengers in rail networks*, Proceedings of the International Workshop on Simulation Science (SimScience 2017), Communications in Computer and Information Science, Springer International Publishing, 2018, Forthcoming.

[BCG07] Fabio Luigi Bellifemine, Giovanni Caire, and Dominic Greenwood, *Developing multi-agent systems with jade*, vol. 7, Wiley & Sons, 2007.

[BHW07] Rafael H. Bordini, Jomi F. Hübner, and Michael Wooldridge, *Programming multi-agent systems in AgentSpeak using Jason*, Wiley & Sons, 2007.

[BRM+09] Michael Balmer, Marcel Rieser, Konrad Meister, David Charypar, Nicolas Lefebvre, and Kai Nagel, *Matsim-t: Architecture and simulation times*, Multi-agent systems for traffic and transportation engineering, IGI Global, 2009, pp. 57–78.

[DM15] Sophie Dennisen and Jörg P. Müller, *Agent-based voting architecture for traffic applications*, Proceedings of the 13th German Conference of Multiagent System Technologies (MATES 2015), Lecture Notes in Artificial Intelligence, vol. 9433, Springer International Publishing, September 2015, pp. 200–217.

[HDBVDHM00] Koen V. Hindriks, Frank S. De Boer, Wiebe Van Der Hoek, and John-Jules Ch. Meyer, *Agent programming with declarative goals*, International Workshop on Agent Theories, Architectures, and Languages, Springer International Publishing, 2000, pp. 228–243.

[KHRW02] Daniel Krajzewicz, Georg Hertkorn, Christian Rössel, and Peter Wagner, *Sumo (simulation of urban mobility) - an open-source traffic simulation*, Proceedings

[5] See `https://goo.gl/forms/fn1mnMcOV3g2QOFE2`

of the 4th middle East Symposium on Simulation and Modelling (MESM2002), 2002, pp. 183–187.

[NS92] Kai Nagel and Michael Schreckenberg, *A cellular automaton model for freeway traffic*, Journal de physique I **2** (1992), no. 12, 2221–2229.

[Rao96] Anand S. Rao, *AgentSpeak(L): BDI agents speak out in a logical computable language*, Proceedings of the 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW 1996), Springer International Publishing, 1996, pp. 42–55.

[SBP12] David Sanderson, Didac Busquets, and Jeremy Pitt, *A micro-meso-macro approach to intelligent transportation systems*, Proceedings of the 6th IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshops (SASOW 2012), IEEE, 2012, pp. 77–82.

[SHR+17] Guni Sharon, Josiah. P. Hanna, Tarun Rambha, Michael W. Levin, Michael Albert, Stephen D. Boyles, and Peter Stone, *Real-time adaptive tolling scheme for optimized social welfare in traffic networks*, Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017), IFAAMAS, 2017, pp. 828–836.

[Sin13] Munindar P. Singh, *Norms as a basis for governing sociotechnical systems*, ACM Transactions on Intelligent Systems and Technology (TIST) **5** (2013), no. 1, 1–23.

[Tuc12] Bill Tucker, *The flipped classroom*, Education next **12** (2012), no. 1.

[ZRH16] Maicon R. Zatelli, Alessandro Ricci, and Jomi F. Hübner, *A concurrent architecture for agent reasoning cycle execution in Jason*, Multi-Agent Systems and Agreement Technologies: EUMAS 2015 and AT 2015, Lecture Notes in Computer Science, vol. 9571, Springer International Publishing, 2016, pp. 425–440.

Technische Universität Clausthal, Institut für Informatik, Clausthal-Zellerfeld, Germany, 38678

Email address: `[malte|philipp|sophie]@lightjason.org, jmue@tu-clausthal.de`