Quantitative Operational Monitoring for BDI Agents

Extended Abstract

Marie Farrell
The University of Manchester
Manchester, United Kingdom
marie.farrell@manchester.ac.uk

Angelo Ferrando
University of Modena and Reggio Emilia
Modena, Italy
angelo.ferrando@unimore.it

Mengwei Xu
Newcastle University
Newcastle, United Kingdom
mengwei.xu@newcastle.ac.uk

ABSTRACT

Belief-Desire-Intention (BDI) architecture is a popular framework for designing autonomous systems. As these systems make independent decisions and execute actions independent from humans, ensuring their safety and reliability becomes a major concern. Traditional verification methods often fail to give run-time operational insights into an agent's behaviours, especially with quantitative assessments under uncertain conditions, such as imprecise actuating. Meanwhile, BDI agents, which rely on context-sensitive subtask expansion, act as they go *e.g.* selecting plans at run time. To address this, we have developed a monitoring method that combines real-time operational data with probabilistic verification. This approach allows us to quantitatively analyse the decisions of BDI agents as they occur to understand the impact of each decision as it happens.

KEYWORDS

BDI Agents; Quantitative Monitoring

ACM Reference Format:
Marie Farrell, Angelo Ferrando, and Mengwei Xu. 2025. Quantitative Operational Monitoring for BDI Agents: Extended Abstract. In *Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Detroit, Michigan, USA, May 19 – 23, 2025*, IFAAMAS, 3 pages.

1 INTRODUCTION

The Belief-Desire-Intention (BDI) [8] framework is a well-studied rational agent framework, forming the basis of, among others, AgentSpeak [21], 3APL [15], 2APL [10], Jason [7], and Conceptual Agent Notation (Can) [22, 23]. In a BDI agent, the *(B)eliefs* represent what the agent knows, the *(D)esires* what the agent wants to bring about, and the *(I)ntentions* those desires the agent has chosen to act upon. BDI agents have been very successful in many areas such as business [6] and healthcare [9].

However, designing BDI-based agents can be complex *e.g.* with a mix of declarative specification (a description of the state sought), procedural specification (a set of instructions to perform), and inherently interleaved concurrent behaviours (*e.g.* multi-tasking). Crucially, this complexity raises concerns about the safety and trustworthiness of the deployment of these agents. As such, there is a growing demand for verification techniques to analyse the behaviours of these BDI agents, exemplified by works [1–3, 11, 13].

Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Y. Vorobeychik, S. Das, A. Nowé (eds.), May 19 – 23, 2025, Detroit, Michigan, USA. © 2025 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

Most current verification methods analyse BDI agents before deployment from some fixed starting states. However, BDI agents make decisions during operation, and we often need to know, for example, the probability of completing a task after doing actions X and Y or selecting plans A and B. Manually updating the new initial state and re-verifying it is possible. Unfortunately, the process for such manual adjustments and subsequent re-verification each time introduces both a high risk of errors (compromising the safety assurance) and a substantial amount of inefficiency for the system analysts. As such, we need an error-resistant and efficient approach to analysing BDI agents that, by nature, make decisions at run time.

To achieve this, we adopt a monitor-based verification methodology *i.e.* Run-time Verification (RV), a non-intrusive verification technique [4] to analyse systems during the operation. However, existing RV approaches often lack supporting probabilistic behaviours [5, 18] or use non-probabilistic counterparts *e.g.* [12, 20, 25]. This qualitative focus does not need the probabilistic nature of BDI-based systems *e.g.* imprecise actuating. As such, we propose our quantitative operational monitoring methodology which not only retains the strengths of traditional RV (*e.g.* robust performance with minimised system impact) but also enriches it by quantitatively analysing the probabilistic aspects of BDI-based system behaviours. It achieves this by directly verifying a BDI agent at run time, providing a quantitative analysis of system behaviour (*e.g.* what is the probability of achieving the given task after selecting plan A) instead of traditional boolean ones. This quantitative approach allows finer responses, facilitating proactive actions in near-failure scenarios, which is essential for safety-critical autonomous systems.

Our methodology comprises two phases: Design Time and Run Time. At design time, BDI agents are modelled using a suitable probabilistic specification language chosen by the system analysts. All potential behaviours are explored and captured in the transition system of a Discrete-Time Markov Chain (DTMC) [16] before deployment. At run time, a monitor is automatically constructed to observe the system during operation. The observed run-time data is then used to update the DTMC to reflect the agent's actual operational progress. In revising the DTMC, we avoid the re-construction of transition systems, which is typically the most computationally costly verification aspect. Analysing the probabilistic system via an updated DTMC has negligible overhead, ensuring immediate and relevant quantitative insights aligned with monitoring. If verification yields inconclusive results within set bounds, the system proceeds, enabling further analysis with richer operational data for a definitive conclusion. This approach ensures ongoing, precise insights into system behaviour at any operation stage.

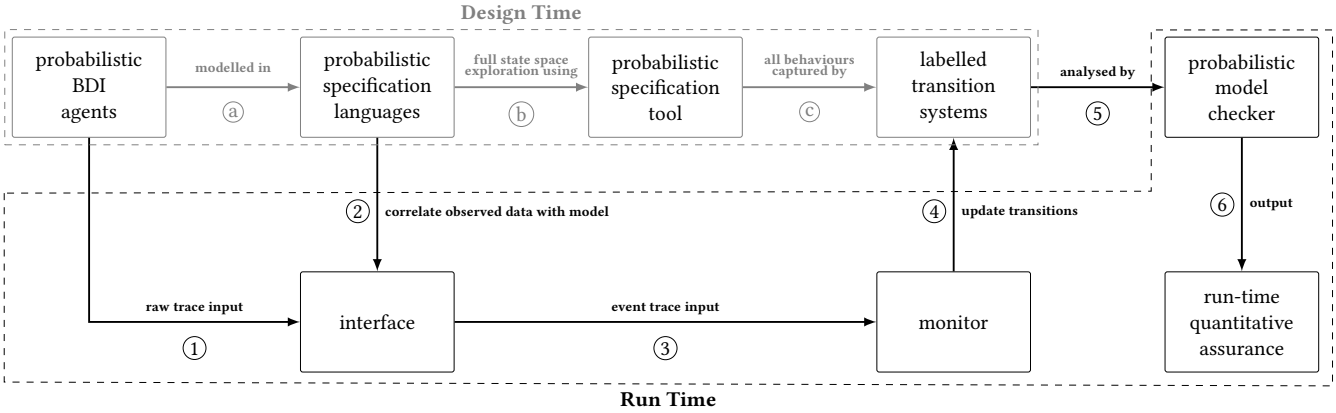

Figure 1: Quantitative Operational Monitoring Methodology for BDI Agents.

2 METHODOLOGY

Our methodology in Fig. 1 consists of two phases: Design Time and Run Time. In the former, on the top-left, we have probabilistic BDI agents (e.g. from [2]) modelled (ⓐ) in a suitable probabilistic specification language. A design-time full-state space exploration (ⓑ) follows by the relevant tool for the chosen probabilistic language. Its output is captured in the transition system of a DTMC, specifying all possible system behaviours (ⓒ). We note that a design-time quantitative analysis could be obtained through step ⑤ immediately, employing model checkers (such as PRISM [17]) to verify properties from a manually specified initial state.

Our objective is to provide a quantitative analysis of the behaviours of BDI agents in operation. This requires collecting and reasoning about run-time operational data from the BDI agents, and, importantly, providing the run-time analysis at any given point during operation. We achieve this in the run-time phase by employing a monitor-based verification approach that does not necessitate interfering with the model itself. Instead, it always observes the system as it operates (①), capturing the latest data (in the form of raw input such as system logs) from the system. To verify the running system, we correlate the observed data with the model (②), obtaining the trace of events (*e.g.* executed path) in the model (③). We update the transition system accordingly to reflect the system's evolution regarding these events (④). We highlight that by directly updating the transition system with events observed in the monitor, we bypass the need to re-construct these transition systems, typically the most computationally costly verification aspect. Meanwhile, inspecting the system via the transition system has extremely low computational overhead [1] (⑤ and ⑥). As such, each quantitative verification is as prompt as the monitoring itself, guaranteeing the quantitative insights are timely and pertinent.

3 RELATED WORK

Formal verification of BDI has been summarised in a survey [19]. However, none of them supports the operational monitoring that was provided in our work. Meanwhile, there is a vast and mature community of run-time verification see e.g. in [4]. However, most of them do not support probabilistic reasoning. As such, there is a growing trend that applies probabilistic verification techniques for the run-time analysis on other systems, which may need to be re-constructed, *e.g.* adaptive software systems. For example, [14] focus on system re-configurations (*e.g.* a new device in the network), using a run-time probabilistic verification that re-uses the previous analysis results when some small changes are made to the model of the system. Our approach, which is on BDI agents, maintains the same high-level governing descriptions of agent behaviours through semantics. That said, given different initial BDI agent configurations (*i.e.* different agent programs), corresponding DTMCs can be generated and monitored by our approach.

Another loosely related approach for run-time probabilistic verification is through statistical model checking [24]. This method, largely proposed to overcome the state space explosion issue inherent in traditional model checking, avoids exhaustive exploration of all system states by sampling system executions and statistically estimating the probability that specific properties hold. As such, it can return a verification result relatively fast to be acceptable for run-time usage. However, it still remains "static" unless initial states in the model are manually updated each time to be in sync with the current state of the actual system, and, importantly, by bypassing exhaustive exploration, it sacrifices the strong correctness guarantees we are seeking for responsible autonomous systems.

4 CONCLUSION

Quantitative operational monitoring, providing run-time insights, enables proactive decision-making and mitigation strategies for BDI agent analysts to react promptly before safety violations even occur. Besides the proposed methodology, we note that significant progress has been made in implementing our methodology, illustrating our approach in action with proven practicality. In addition to the feasibility, we are also in the process of extensively analysing our approach for computational efficiency and scalability, ensuring broader applicability independent of particular BDI designs.

ACKNOWLEDGMENTS

This work is partially supported by a Royal Academy of Engineering Research Fellowship. Authors are listed alphabetically to indicate equal contributions.

REFERENCES

[1] Blair Archibald, Muffy Calder, Michele Sevegnani, and Mengwei Xu. 2022. Modelling and Verifying BDI Agents with Bigraphs. *Science of Computer Programming* 215 (2022), 102760. https://doi.org/10.1016/j.scico.2021.102760

[2] Blair Archibald, Muffy Calder, Michele Sevegnani, and Mengwei Xu. 2023. Quantitative modelling and analysis of BDI agents. *Software and Systems Modeling* (2023).

[3] Blair Archibald, Muffy Calder, Michele Sevegnani, and Mengwei Xu. 2023. Quantitative Verification and Strategy Synthesis for BDI Agents. In *Proceedings of NASA Formal Methods*. Springer Nature Switzerland, 241–259.

[4] Ezio Bartocci, Yliès Falcone, Adrian Francalanza, and Giles Reger. 2018. Introduction to Runtime Verification. In *Lectures on Runtime Verification - Introductory and Advanced Topics (Lecture Notes in Computer Science, Vol. 10457)*. Springer, 1–33.

[5] Andreas Bauer, Martin Leucker, and Christian Schallhart. 2011. Runtime Verification for LTL and TLTL. *ACM Trans. Softw. Eng. Methodol.* 20, 4 (2011), 14:1–14:64.

[6] Steve S Benfield, Jim Hendrickson, and Daniel Galanti. 2006. Making a strong business case for multiagent technology. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent systems*. ACM, 10–15.

[7] R.H. Bordini, J.F. Hübner, and M. Wooldridge. 2007. *Programming multi-agent systems in AgentSpeak using Jason*. Vol. 8. John Wiley & Sons.

[8] Michael Bratman. 1987. *Intention, Plans, and Practical reason*. Harvard University Press.

[9] Lars Braubach, Alexander Pokahr, and Winfried Lamersdorf. 2014. Negotiation-based Patient Scheduling in Hospitals. In *Advanced Intelligent Computational Technologies and Decision Support Systems*. 107–121.

[10] Mehdi Dastani. 2008. 2APL: a practical agent programming language. *Autonomous agents and multi-agent systems* 16, 3 (2008), 214–248.

[11] Louise A Dennis, Michael Fisher, Nicholas K Lincoln, Alexei Lisitsa, and Sandor M Veres. 2016. Practical verification of decision-making in agent-based autonomous systems. *Automated Software Engineering* 23 (2016), 305–359.

[12] Angelo Ferrando, Rafael C. Cardoso, Marie Farrell, Matt Luckcuck, Fabio Papacchini, Michael Fisher, and Viviana Mascardi. 2021. Bridging the gap between single- and multi-model predictive runtime verification. *Formal Methods Syst. Des.* 59, 1 (2021), 44–76.

[13] Michael Fisher, Viviana Mascardi, Kristin Yvonne Rozier, Bernd-Holger Schlingloff, Michael Winikoff, and Neil Yorke-Smith. 2021. Towards a framework for certification of reliable autonomous systems. *Autonomous Agents and Multi-Agent Systems* 35 (2021), 1–65.

[14] Vojtech Forejt, Marta Z. Kwiatkowska, David Parker, Hongyang Qu, and Mateusz Ujma. 2012. Incremental Runtime Verification of Probabilistic Systems. In *Runtime Verification, Third International Conference, RV 2012, Istanbul, Turkey, September 25-28, 2012, Revised Selected Papers (Lecture Notes in Computer Science, Vol. 7687)*. Springer, 314–319. https://doi.org/10.1007/978-3-642-35632-2_30

[15] Koen V. Hindriks, Frank S. De Boer, Wiebe Van der Hoek, and John-Jules Ch Meyer. 1999. Agent programming in 3APL. *Autonomous Agents and Multi-Agent Systems* 2, 4 (1999), 357–401.

[16] Marta Kwiatkowska, Gethin Norman, and David Parker. 2010. Advances and challenges of probabilistic model checking. In *2010 48th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 1691–1698.

[17] Marta Z. Kwiatkowska, Gethin Norman, and David Parker. 2011. PRISM 4.0: Verification of Probabilistic Real-Time Systems. In *Computer Aided Verification - 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings (Lecture Notes in Computer Science, Vol. 6806)*. Springer, 585–591.

[18] Martin Leucker and Christian Schallhart. 2009. A brief account of runtime verification. *J. Log. Algebraic Methods Program.* 78, 5 (2009), 293–303.

[19] Matt Luckcuck, Marie Farrell, Louise A Dennis, Clare Dixon, and Michael Fisher. 2019. Formal specification and verification of autonomous robotic systems: A survey. *ACM Computing Surveys (CSUR)* 52, 5 (2019), 1–41.

[20] Stefan Mitsch and André Platzer. 2016. ModelPlex: verified runtime validation of verified cyber-physical system models. *Formal Methods Syst. Des.* 49, 1-2 (2016), 33–74.

[21] Anand S. Rao. 1996. AgentSpeak (L): BDI agents speak out in a logical computable language. In *Proceedings of European Workshop on Modelling Autonomous Agents in a Multi-Agent World*. Springer, 42–55.

[22] Sebastian Sardina and Lin Padgham. 2011. A BDI agent programming language with failure handling, declarative goals, and planning. *Autonomous Agents and Multi-Agent Systems* 23 (2011), 18–70.

[23] Michael Winikoff, Lin Padgham, James Harland, and John Thangarajah. 2002. Declarative & procedural goals in intelligent agent systems. *KR* 2002 (2002), 470–481.

[24] Håkan LS Younes, Marta Kwiatkowska, Gethin Norman, and David Parker. 2004. Numerical vs. statistical probabilistic model checking: An empirical study. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 46–60.

[25] Yuhong Zhao, Simon Oberthür, Martin Kardos, and Franz-Josef Rammig. 2005. Model-based Runtime Verification Framework for Self-optimizing Systems. In *Proceedings of the Fifth Workshop on Runtime Verification, RV@CAV 2005, Edinburgh, UK, July 12, 2005 (Electronic Notes in Theoretical Computer Science, Vol. 144)*. Elsevier, 125–145.