StarCraft as a Testbed for Engineering Complex Distributed Systems Using Cognitive Agent Technology

Extended Abstract

Vincent J. Koeman, Harm J. Griffioen, Danny C. Plenge, Koen V. Hindriks
Delft University of Technology, The Netherlands
{v.j.koeman,h.j.griffioen,p.c.plenge,k.v.hindriks}@tudelft.nl

ABSTRACT

It has been argued that the evaluation of cognitive agent systems requires richer benchmark problems. We think that real-time strategy (RTS) games can offer such a testbed, as AI for RTS requires the design of complicated strategies for coordinating hundreds of units that need to solve a range of challenges. Therefore, in this paper, we report on the design and development of the first multi-agent connector that provides full access to StarCraft (Brood War). We provide a new interface that is dedicated to a multi-agent approach by connecting each unit in the game to a cognitive agent. Two main challenges are addressed in this work. First, we decide on the right level of abstraction for unit control by means of agents, designing for instance the percepts that are available to units. Second, a sufficient level of performance needs to be ensured in order to allow a large variety of multi-agent implementations to be successful at tackling challenges of RTS AI. The resulting open-source connector readily supports the hundreds of agents that can come and go during the game. Based on the development of the connector and its initial use by over 200 students, we gained valuable insights.

ACM Reference Format:
Vincent J. Koeman, Harm J. Griffioen, Danny C. Plenge, Koen V. Hindriks. 2018. StarCraft as a Testbed for Engineering Complex Distributed Systems Using Cognitive Agent Technology. In *Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), Stockholm, Sweden, July 10–15, 2018,* IFAAMAS, 3 pages.

1 INTRODUCTION

RTS games that deploy large numbers of units can provide an ideal testbed for cognitive agent technologies [3, 14]. The basic idea is that by controlling each unit with a cognitive agent, a program that derives its choice of action from its cognitive state of beliefs and goals represented using some KR technology such as Prolog, we obtain an agent system that needs to address the complicated challenges of coordination and responsiveness. The idea is that a one-to-one mapping between game units and cognitive agents provides a set-up that is ideal for benchmarking and putting cognitive agent systems to the test. Based on this idea, and in accordance with Google (DeepMind) and many other AI researchers [11, 13, 14], we believe that StarCraft is the most suitable RTS game to target as a testbed. Moreover, the several popular competitions for StarCraft AI can serve as a benchmark for implementations [13].

Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), M. Dastani, G. Sukthankar, E. André, S. Koenig (eds.), July 10–15, 2018, Stockholm, Sweden. © 2018 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

Our work thus aims to provide opportunities for demonstrating the added value of and drawing new lessons about our cognitive agent technologies by designing and implementing the first connector for cognitive agents that provides full access to StarCraft (Brood War). Such a connector will bring the aforementioned challenges of RTS games to the field of AOP whilst conversely also opening up a new strategy space for developers of AI for RTS games. From the perspective of cognitive agent technologies, we believe that by facilitating cognitive agents to effectively interface with the prototypical RTS game of StarCraft through our connector, we significantly 'raise the bar' in terms of scalability and related issues. Moreover, we believe that StarCraft provides a very challenging environment to demonstrate the potential and promise of cognitive agent technology for engineering complex distributed systems that simultaneously need to address many AI challenges [6, 12].

Our focus is on the design of a connector that enables and facilitates the use of cognitive agent technology for engineering strategies for StarCraft based on a one-to-one unit-agent mapping. The basic requirement that such a connector should support this unit-agent mapping introduces important challenges in itself:

1. The connector should facilitate an agent system that operates at a level of *abstraction* that is appropriate to cognitive agents.
2. The connector should be sufficiently *performant* in order to support a sufficient variety of viable implementations (i.e., both different approaches to implementing strategies as well as the use of different agent platforms).

In other words, we do not aim to facilitate agent systems that operate at the same level of detail as bots written for StarCraft in C++ or Java, but such systems should in contrast also not consist of a single action '`win`' that will delegate the control to some other subsystem instead. To make optimal use of the reasoning typically employed by cognitive agents [1], low-level details should be left to specific control layers whilst still allowing agents sufficiently fine-grained control in their decision making.

2 RELATED WORK

Connectors that support connecting cognitive agent technology to games have been made available for other games [2]. So far, however, most connectors have remained rather simple. The most complex multi-agent connectors that have been made available so far, are connectors for Unreal Tournament [7]. The design of such a connector involves similar issues related to the facilitated level of abstraction and the resulting performance as in this work. However, the resulting implementation as reported on in [7] does not support running more than 10 agents, whereas for a StarCraft interface we need to connect hundreds of cognitive agents to control the hundreds of units in game. Moreover, corresponding agent systems for Unreal Tournament generally offer only a very restricted set of actions that agents can perform , limiting the complexity of decision making that is required compared to StarCraft.

An API for StarCraft (Brood War) has been thoroughly developed for several years: BWAPI [5]. BWAPI reveals the visible parts of the game state to AI implementations, facilitating the development of competitive (non-cheating) bots. Several dozens of such bots have been created with this API, mostly written in C++ or Java, aimed at participating in one of the tournaments that are being held for StarCraft AI implementations. However, none of this work facilitates employing cognitive agent technologies.

A first attempt at a cognitive agent interface for StarCraft by using BWAPI was performed by Jensen et al. [8]. In this work, a working proof-of-concept that ties in-game units to cognitive agents was introduced. However, it does not address the major challenges such an implementation faces concerning the level of abstraction and corresponding performance, as we do in this work.

3 DESIGN AND IMPLEMENTATION

The core of a connector for cognitive agents exists of three components: (i) the *entities* that are provided for agents to connect to (i.e., units in StarCraft), (ii) the outputs that are generated by each entity (and thus which *percepts* a corresponding agent receives), and (iii) the inputs that are available for each entity (and thus which *actions* an agent controlling the entity can perform).

As our approach is to provide an entity (i.e., to which an agent can connect) for each unit, and the available actions for each unit are mainly defined by the (interface to) the game itself, the main challenge when balancing the level of abstraction with the resulting performance is in determining the percepts that are available. Combining the (finite set of) information that is available through the BWAPI interface with a developed set of guidelines lead to a set of about 25 percepts[1]. By default, each entity will receive percepts that are relevant to the state of the corresponding StarCraft unit. Such percepts include status information (e.g., the unit's position and its remaining health) and, depending on the type of unit, specific situational information (e.g., the units that are loaded into the current unit if it is loadable). This facilitates decentralized reasoning to be employed by agents about their own state.

Because of the tight performance requirements on StarCraft AI, we have found it useful to provide specific mechanisms to a developer to fine-tune the delivery of percepts containing general information about the match to specific agents. Therefore, through the connector's initialization settings, a list of desired 'global information' (i.e., names of percepts) can be given ("subscribed to") for each unit type. In addition, for a complex game as StarCraft, developers of cognitive solutions will need more advanced support to structure their agents. To this end, we have designed our connector to support special kind of entities, '*managers*', that do not match with unique in-game units but but do have the ability to receive desired percepts through the aforementioned global percepts. Such managers are especially useful to reason about groups of units.

[1]For the full set of percepts and actions available in the environment, see https://github.com/eishub/Starcraft/blob/master/doc/Resources/StarCraftEnvironmentManual.pdf.

4 EVALUATION

In the evaluation of our connector, we have focused first on the performance, as high performance is critical for any MAS approach that uses many agents to deal with the challenges of AI for RTS. We also evaluated our requirement that the connector should not restrict the strategy space in any essential way by looking at a considerable set of implementations that make use of our connector. In three iterations, the connector was refined to its current state. Initially, a pilot was held with around 100 masters students that worked in groups on creating a StarCraft bot using this connector. This helped us track down and resolve initial issues that were present in the connector prototype. More recently, over 200 first year bachelors students did the same with an improved version of the connector, being the *largest StarCraft AI project so far*. Students successfully created full-fledged bots within just 8 weeks, of which some even joined the Student StarCraft AI Tournament (SSCAIT) [4]. In addition, a masters student has developed a StarCraft AI implementation using this connector that is ranked in SSCAIT at around the 30th place of 90 active bots (which are mostly written in C++ or Java) at the time of writing.

5 CONCLUSIONS AND FUTURE WORK

Our cognitive agent connector for StarCraft readily supports the hundreds of agents that can come and go during the game, each of which has to deal with major challenges such as uncertainty and long-term collaborative goals. The unit-to-agent approach of our connector is different from most StarCraft AI implementations. Its viability is demonstrated by multiple large-scale practical uses of the connector, resulting in a varied set of competitive AIs. We believe that our connector provides opportunities to the agent community to show the potential of cognitive agent technologies for addressing the challenges such environments provide. Ensuring a sufficient level of performance of the connector was a significant challenge that had to be addressed in particular in order to ensure a unit-agent mapping approach is viable. In our evaluations, we determined the baseline performance of the connector in a worst-case scenario, which shows that on average there remains sufficient CPU time for strategic reasoning in the cognitive agents. Even though the performance of an agent system depends largely on the used cognitive technology itself, we believe that our connector can be effectively used in practice.

Although this work is focused on StarCraft Brood War, the brand new 'raw API' of StarCraft 2 is reported to be similar to BWPAI in Vinyals et al. [15], and the work in this paper should therefore be relatively straightforwardly applicable and/or portable to StarCraft 2 (and possibly other RTS games) in future work.

Finally, a challenge that was not addressed is the fact that debugging (cf. Koeman et al. [10]) becomes increasingly difficult with increasing numbers of agents. As debugging concurrent programs is a hard problem in general, more work is required in this area. In addition, in order to better support automated testing (cf. Koeman et al. [9]), it may be beneficial to develop a mechanism that automatically saves the state of an agent system when a save game is created in StarCraft. This can be used to immediately initialize the system to the desired state when executing a test with a specific save game (i.e., representing a scenario).

REFERENCES

[1] Rafael H. Bordini, Mehdi Dastani, Jürgen Dix, and Amal El Fallah Seghrouchni. 2009. *Multi-Agent Programming*. Springer.

[2] Frank Dignum. 2012. Agents for games and simulations. *Autonomous Agents and Multi-Agent Systems* 24, 2 (March 2012), 217–220.

[3] Frank Dignum, Joost Westra, Willem A van Doesburg, and Maaike Harbers. 2009. Games and agents: Designing intelligent gameplay. *International Journal of Computer Games Technology* 2009 (2009).

[4] Michal Čertický, Paul Paradies, Marek Šuppa, Björn Persson Mattsson, Tomáš Vajda, Rafał Poniatowski, and Sören Klett. 2011. Student StarCraft AI Tournament. https://sscaitournament.com. (2011). Accessed: 2017-11-14.

[5] Adam Heinermann. 2008. Brood War API. https://github.com/bwapi/bwapi. (2008). Accessed: 2017-11-14.

[6] Koen V. Hindriks. 2014. *The Shaping of the Agent-Oriented Mindset*. Springer International Publishing, 1–14.

[7] Koen V. Hindriks, Birna van Riemsdijk, Tristan Behrens, Rien Korstanje, Nick Kraayenbrink, Wouter Pasman, and Lennard de Rijk. 2011. Unreal GOAL Bots. In *Agents for Games and Simulations II: Trends in Techniques, Concepts and Design*, Frank Dignum (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 1–18.

[8] Andreas Schmidt Jensen, Christian Kaysø-Rørdam, and Jørgen Villadsen. 2015. Interfacing Agents to Real-Time Strategy Games. In *SCAI*. 68–77.

[9] Vincent J. Koeman, Koen V. Hindriks, and Catholijn M. Jonker. 2016. Automating Failure Detection in Cognitive Agent Programs. In *Proceedings of the 2016 International Conference on Autonomous Agents &#38; Multiagent Systems (AAMAS '16)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 1237–1246.

[10] Vincent J. Koeman, Koen V. Hindriks, and Catholijn M. Jonker. 2017. Designing a source-level debugger for cognitive agent programs. *Autonomous Agents and Multi-Agent Systems* 31, 5 (Sept. 2017), 941–970.

[11] R. Lara-Cabrera, C. Cotta, and A.J. Fernández-Leiva. 2013. A review of computational intelligence in RTS games. In *2013 IEEE Symposium on Foundations of Computational Intelligence (FOCI)*. 114–121.

[12] Brian Logan. 2015. *A Future for Agent Programming*. Springer International Publishing, Cham, 3–17.

[13] S. Ontañón, G. Synnaeve, A. Uriarte, F. Richoux, D. Churchill, and M. Preuss. 2013. A Survey of Real-Time Strategy Game AI Research and Competition in StarCraft. *IEEE Transactions on Computational Intelligence and AI in Games* 5, 4 (Dec. 2013), 293–311.

[14] Glen Robertson and Ian Watson. 2014. A review of real-time strategy game AI. *AI Magazine* 35, 4 (2014), 75–104.

[15] Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John Agapiou, Julian Schrittwieser, et al. 2017. StarCraft II: A New Challenge for Reinforcement Learning. *arXiv preprint arXiv:1708.04782* (Aug. 2017).