Agent-Based Programming

Subjects: Computer Science, Artificial Intelligence
Contributor: Rafael C. Cardoso

Intelligent and autonomous agents is a subarea of symbolic artificial intelligence where these agents decide, either reactively or proactively, upon a course of action by reasoning about the information that is available about the world (including the environment, the agent itself, and other agents). It encompasses a multitude of techniques, such as negotiation protocols, agent simulation, multi-agent argumentation, multi-agent planning, and many others. In an agent-based programming language, agents are the building blocks, and programs are obtained by programming their behaviours (how an agent reasons), their goals (what an agent aims to achieve) and their interoperation (how agents collaborate to solve a task).

Keywords: agent-based programming ; multi-agent systems

1. Introduction

Multi-Agent Systems (MASs) [1] are a well established branch of Artificial Intelligence (AI). Even though they are relatively young with respect to more archetypal research areas, MASs have a rich history; in 1995 [2] agent technology was recognised as a rapidly developing research area and one of the fastest growing areas of information technology. Such a statement is still true nowadays, where one can find plenty of research articles, tools, and conferences whose aim is to advance the research in the area. Despite this, MASs are not as widely used as they could be. Considering the agent programming aspect of MASs, according to [3], the key reason is that there is little incentive for developers to switch to current Agent Programming Languages (APLs), as the behaviours that can be easily programmed are sufficiently simple to be implementable in mainstream languages with only a small overhead in coding time. This, amongst the presence of too many unorganised options available, does not help agent-based programming languages and tools to be picked from non-expert users.

An intelligent agent [4] can be generalised as a computerised entity that: is able to reason (rational/cognitive), to make its own decisions independently (autonomous), to collaborate with other agents when necessary (social), to perceive the context in which it operates and react to it appropriately (reactive), and finally, to take action in order to achieve its goals (proactive). An agent-based (or agent-oriented) system is a system where the agents are the main entities, treated as first-class abstractions. From a programming perspective, the same reasoning can be followed. In particular, by using a comparison, we can say that agents are to Agent-Oriented Programming (AOP) languages as objects are to Object-Oriented Programming (OOP) languages. In an agent-based programming language, agents are the building blocks, and programs are obtained by programming their behaviours (how an agent reasons), their goals (what an agent aims to achieve) and their interoperation (how agents collaborate to solve a task).

Agents are well-suited to be used in applications involving distributed or concurrent computation or when communication is required between different components. For this reason, agent technology is useful in applications that reason about messages/objects received over a network. By preserving their processing state and the state of the world around them, agents are also ideally suited to automation applications. Moreover, autonomous agents can operate without user intervention and can be used in applications such as plant/process automation, workflow management, robotics, and others. Another advantage of agent-based programming is that due to the reasoning cycle present in agents, it is also possible to provide explanations about the decisions that an agent has made.

2. History on Agent-Based Programming

In 1993, Agent-Oriented Programming was first introduced [5] as a specialisation of Object-Oriented Programming. Most notably, it discusses the notion of the mental state of an agent, consisting of its information, decisions, and capabilities. This work also describes agent programs in the AGENT-0 interpreter (implemented in the Lisp language) and their communication using speech act theory, the latter is still used to define agent communication in several contemporary agent programming languages. Over the years, many reasoning and cognitive models have been developed for agent-

based programming. In this section, we discuss three particular models that have been fundamental in the design of many agent programming languages in the past and that are still being used in new languages these days: Procedural Reasoning System (PRS), BDI, and Situation Calculus.

The Procedural Reasoning System (PRS) [6] (implemented in Lisp) defines a system capable of reasoning about processes, that is, procedural forms of knowledge. An agent in this system is then able to use these procedures to select intentions for achieving particular goals. Unlike in conventional programming languages, these procedures are not invoked a priori, but they are triggered when they are able to contribute towards some goal or to react to some situation. While sharing some similarities to AI planners of the time, its main difference is that it performs partial hierarchical planning in the sense that it interacts with a dynamic environment during the reasoning process, instead of generating a plan for a static environment.

The Belief-Desire-Intention (BDI) model [7][8] consists of a reasoning process that aids the decision-making of selecting an appropriate action towards the achievement of some goal. Its three mental attitudes are: belief—knowledge that the agent believes about its environment, itself, and other agents; desire—the desired states that the agent wants to achieve; and intention—a sequence of steps towards the achievement of a desire. These mental attitudes respectively represent the information, motivational, and deliberative states of the agent. The workflow in a generic BDI system is shown in Figure 1 and works as such: a belief revision function receives input information from the environment (e.g., sensors), and it is responsible for updating the belief base. This update can generate more options that can become current desires based on the belief base and the intentions base. A filter is responsible for updating the intentions base, taking into account its previous state and the current belief base and desire base. Finally, an intention is chosen to be carried out as an action by the agent. BDI is the most popular model of agency, it has been and continues to be used in many agent programming languages. AgentSpeak(L) [9] is a language that serves as an abstraction of implemented BDI systems that can be used to interpret agent programs as horn-clause logic programs. The theory behind this language has been implemented as a basis for many APLs.

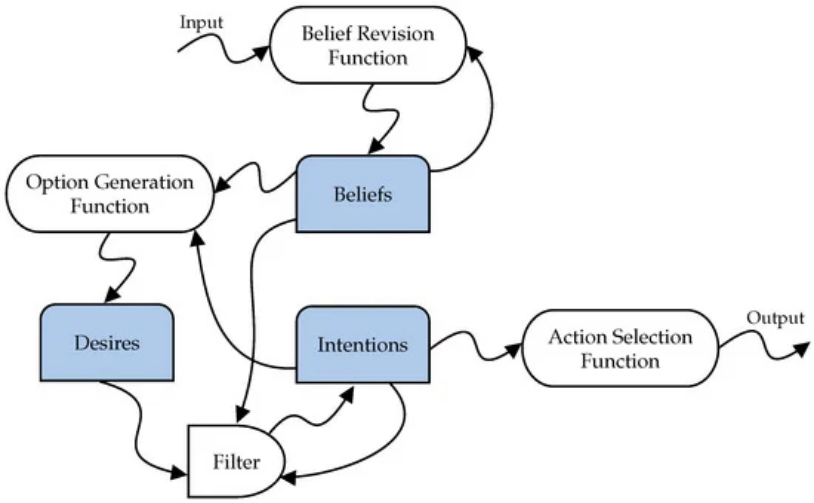

Figure 1. The BDI model.

Situation Calculus [10] is a first order language designed to represent changes in dynamic environments. A situation is a first order term representing a sequence of actions. An initial situation is when no actions have occurred yet. The function do(a,s) results in a successor situation to *s* after executing the action *a*, similar to state transition systems. Dynamic environments play an important role in agent-based programming, and as such, Situation Calculus has been used to model how the world changes as a result of executing actions.

References

1. Wooldridge, M. An Introduction to MultiAgent Systems, 2nd ed.; John Wiley and Sons: Hoboken, NJ, USA, 2009; ISBN 047149691X.
2. Wooldridge, M.J.; Jennings, N.R. Intelligent agents: Theory and practice. Knowl. Eng. Rev. 1995, 10, 115–152.
3. Logan, B. An agent programming manifesto. Int. J. Agent-Oriented Softw. Eng. 2018, 6, 187–210.
4. Russell, S.J.; Norvig, P. Artificial Intelligence: A Modern Approach, 3rd ed.; Prentice Hall: Upper Saddle River, NJ, USA, 2010.

5. Shoham, Y. Agent-oriented Programming. Artif. Intell. 1993, 60, 51–92.
6. Georgeff, M.; Lansky, A. Procedural Knowledge. Proc. IEEE (Spec. Issue Knowl. Represent.) 1986, 74, 1383–1398.
7. Bratman, M.E. Intentions, Plans, and Practical Reason; Center for the Study of Language and Information: Stanford, CA, USA, 1999.
8. Rao, A.S.; Georgeff, M. BDI Agents: From Theory to Practice. In Proceedings of the First International Conference on Multiagent Systems (ICMAS), San Francisco, CA, USA, 12–14 June 1995; pp. 312–319.
9. Rao, A.S. AgentSpeak(L): BDI Agents Speak Out in a Logical Computable Language. In Agents Breaking Away, Proceedings of the 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, Eindhoven, The Netherlands, 22–25 January 1996; Lecture Notes in Computer Science; de Velde, W.V., Perram, J.W., Eds.; Springer: Berlin/Heidelberg, Germany, 1996; Volume 1038, pp. 42–55.
10. McCarthy, J.; Hayes, P.J. Some Philosophical Problems from the Standpoint of Artificial Intelligence. In Machine Intelligence 4; Meltzer, B., Michie, D., Eds.; Edinburgh University Press: Edinburgh, UK, 1969; pp. 463–502.
