Cognitive agents with non-monotonic reasoning

Peter Novák
peter.novak@tu-clausthal.de

Department of Computer Science
Clausthal University of Technology
Julius-Albert-Str. 4, D-38678 Clausthal-Zellerfeld, Germany

ABSTRACT

This extended abstract provides an overview of my research towards a dissertation thesis in the context of *programming cognitive agents with non-monotonic reasoning capabilities.*

Categories and Subject Descriptors

D.3.1 [**Programming Languages**]: Formal Definitions and Theory; F.3.2 [**Logics and Meanings of Programs**]: Semantics of Programming Languages; I.2.5 [**Artificial Intelligence**]: Programming Languages and Software; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Intelligent Agents*

General Terms

Languages, Theory, Design, Experimentation

Keywords

hybrid agent architectures; agent programming languages; reactive vs. deliberative; virtual agents; open-source software tools; Jazzyk; Jazzbot

1. MOTIVATION

An important aspect of research in programming intelligent agent systems is dealing with highly dynamic and unstructured environments. Embodied agents often have to interact with a world under incomplete information and uncertainty, which can even be intrinsic to the nature of the environment in question. Several approaches to formally model and process information under the open world assumption were proposed in the area of non-monotonic reasoning. In my thesis I focus on *applications of Answer Set Programming (ASP) [1] in the domain of cognitive agents*, i.e. such with explicit representation of their mental attitudes such as beliefs, goals, obligations and alike. I especially focus on implementation of agent's belief base as an ASP knowledge base (KB).

An agent using an ASP module in its belief base and at the same time embodied in an environment, requires a programming framework which 1) allows an easy integration of heterogeneous knowledge bases treated on a par, i.e. no knowledge representation (KR) approach is preferred over another, and 2) provides a flexible programming language for encoding of agent's behaviours. Even though the landscape of agent programming approaches is thriving (see e.g. [2]), none of them provides enough flexibility to allow an easy integration of heterogeneous knowledge representation techniques.

First, I motivate and develop a theoretical framework allowing a straightforward integration of heterogeneous knowledge bases into an agent system. Subsequently, I provide its implementation in a form of an agent programming language with an associated interpreter, and finally I present two case study applications demonstrating a synergistic effect of using an ASP belief base, together with other types of KBs in a single agent performing non-trivial behaviour.

Cite as: Extended Thesis Abstract for Doctoral Mentoring Program, P. Novák, Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008), Padgham, Parkes, Müller and Parsons (eds.), May, 12-16., 2008, Estoril, Portugal, pp. 1746-1747.

2. BEHAVIOURAL STATE MACHINES

The theoretical framework of *Behavioural State Machines* (*BSM*) [9] is the current evolution of *Modular BDI Architecture* [11] and our subsequent studies on modularity of agent programming languages [12]. *BSM* draw a strict distinction between the *knowledge representational* and *behavioural* layer of an agent program. An agent consists of a set of *KR modules*, each providing a set of query and update interface routines, and an *agent program* encoding the agent's behaviours in terms of nested reactive rules. The basic rules consist of two parts: a *query* and an *update*. Queries are expressions accessing the underlying modules via their provided interface routines and if evaluated to true, the execution of the right hand update part is enabled. A primitive update is again an invocation of a KR module interface routine, modifying the underlying partial knowledge base of the agent. Updates and rules form basic *mental state transformers* (mst 's), higher level syntactic constructs allowing source code modularization of an agent program, which in turn is a mst as well. The main focus of the *BSM* framework is thus the highest level of control of an agent: its *behaviours*.

The underlying semantic abstraction is that of a transition system over a set of agent's mental states and a set of transitions between them. An agent's mental state is a collection of partial states of its KBs represented by the agent's KR modules. As the interaction with the environment is facilitated by specialized KR modules as well, the state of the environment is included in the agent's mental state. Transitions are induced by updates of components of mental states. An agent system semantics is then a set of all enabled paths within the transition system, which the agent can traverse during its lifetime. Alternatively, the computational model

of *BSM* provides a functional view on an agent program, specifying a set of enabled transitions/updates, the agent can execute in a situation it happens to be in.

3. CASE STUDIES

To allow implementation of software agents based on the *Behavioural State Machines* framework, I developed *Jazzyk* [8, 10], a programming language closely implementing the *BSM* framework. Subsequently I implemented an interpreter for this language[1].

Following the spirit of [6], where Laird and van Lent argue that approaches for programming intelligent agents should be tested in realistic and sophisticated environments of modern computer games, we work on two case studies demonstrating applicability of *BSM*: *Jazzbot* virtual agent, and an agent team for the *Multi-Agent Programming Contest* scenario.

We designed and implemented *Jazzbot* [13], a virtual agent embodied in a simulated 3D environment of a first-person shooter computer game *Nexuiz*[2]. *Jazzbot* provides a test-bed for investigation of applications of non-monotonic reasoning techniques, ASP in particular, on a realistic, yet affordable agent system.

Jazzbot is a goal-driven agent. It features a *belief base*, *goal base*, and an interface to its *virtual body* in a *Nexuiz* environment. While the goal base consists of a single KB realized as an ASP logic program, the belief base is composed of two modules: ASP logic programming one and a *Ruby*[3] module. *Ruby* is an interpreted object oriented programming language. Finally, the interface to the environment is facilitated by the *Nexuiz* game client module connected to a remote *Nexuiz* server. *Jazzbot*'s behaviours are implemented as a *Jazzyk* program. It can fulfill e.g. search and deliver tasks in the simulated environment while it avoids obstacles and walls.

Multi-Agent Programming Contest[4] [3, 4, 5] provides a well established test-bed for testing approaches to programming multi-agent systems. Implementation of a team of agents for this scenario in *Jazzyk*, with a technological basis similar to that of *Jazzbot*, will allow me to also touch on coordination and cooperation issues in MAS.

4. OPEN ISSUES

My on-going and future work follows two principal directions. First, I focus on development of techniques for programming agents based on the template of *Jazzbot*, so that I can better understand a methodology for programming such systems. The aim is to design at least a fragmentary formal higher level specification language based on modal logic, which allows a straightforward translation (compilation) into raw *Jazzyk* programs.

The second line of research I am following stems from the observation, that the state-of-the-art techniques (for an overview see e.g. [7]) for knowledge base updating w.r.t. semantics of ASP require storing a complete history of logic program updates processed during agent's lifetime. In order to implement a non-trivial LP update mechanism, first the advanced approaches like e.g. *Dynamic Logic Programming* [7] have to be adapted, and possibly simplified.

[1]The first version of *Jazzyk* interpreter was published under GNU GPL license at `http://jazzyk.sourceforge.net/`.
[2]`http://www.alientrap.org/nexuiz`
[3]`http://www.ruby-lang.org/`
[4]`http://cig.in.tu-clausthal.de/AgentContest2007/`

5. REFERENCES

[1] C. Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press, 2003.

[2] Rafael H. Bordini, Lars Braubach, Mehdi Dastani, Amal El Fallah Seghrouchni, Jorge J. Gomez-Sanz, João Leite, Gregory O'Hare, Alexander Pokahr, and Alessandro Ricci. A survey of programming languages and platforms for multi-agent systems. *Informatica*, 30:33–44, 2006.

[3] Mehdi Dastani, Jürgen Dix, and Peter Novák. The first contest on multi-agent systems based on computational logic. In Francesca Toni and Paolo Torroni, editors, *CLIMA VI*, volume 3900 of *Lecture Notes in Computer Science*, pages 373–384. Springer, 2005.

[4] Mehdi Dastani, Jürgen Dix, and Peter Novák. The second contest on multi-agent systems based on computational logic. In Katsumi Inoue, Ken Satoh, and Francesca Toni, editors, *CLIMA VII*, volume 4371 of *Lecture Notes in Computer Science*, pages 266–283. Springer, 2006.

[5] Mehdi Dastani, Jürgen Dix, and Peter Novák. Agent Contest Competition - 3rd edition. In *Proceedings of Fifth international Workshop on Programming Multi-Agent Systems, ProMAS'07*, volume 4908 of *LNAI*. Springer Verlag, 2007.

[6] John E. Laird and Michael van Lent. Human-level AI's killer application: Interactive computer games. *AI Magazine*, 22(2):15–26, 2001.

[7] João Alexandre Leite. *Evolving Knowledge Bases*, volume 81 of *Frontiers of Artificial Intelligence and Applications*. IOS Press, 2003.

[8] Peter Novák. An open agent architecture: Fundamentals (revised version). Technical Report IfI-07-10, Department of Informatics, Clausthal University of Technology, November 2007.

[9] Peter Novák. Behavioural State Machines: programming modular agents. In *AAAI 2008 Spring Symposium: Architectures for Intelligent Theory-Based Agents, AITA'08*, March 26-28 2008.

[10] Peter Novák. Jazzyk: A programming language for hybrid agents with heterogeneous knowledge representations. Sixth International Workshop on Programming Multi-Agent Systems, 2008.

[11] Peter Novák and Jürgen Dix. Modular BDI architecture. In Hideyuki Nakashima, Michael P. Wellman, Gerhard Weiss, and Peter Stone, editors, *AAMAS*, pages 1009–1015. ACM, 2006.

[12] Peter Novák and Jürgen Dix. Adding structure to agent programming languages. In Mehdi Dastani, Amal El Fallah-Seghrouchni, Alessandro Ricci, and Michael Winikoff, editors, *Proceedings of Fifth international Workshop on Programming Multi-Agent Systems, ProMAS'07*, volume 4908 of *LNAI*. Springer Verlag, 2007.

[13] Peter Novák, David Mainzer, Michael Köster, and Bernd Fuhrmann. Jazzbot: A non-monotonically reasoning bot in a simulated 3D environment. 2008.