Jazzbot: A non-monotonically reasoning bot in a simulated 3D environment

Peter Novák, David Mainzer, Michael Köster, Bernd Fuhrmann

Department of Computer Science
Clausthal University of Technology
Julius-Albert-Str. 4, D-38678 Clausthal-Zellerfeld, Germany
peter.novak@tu-clausthal.de

ABSTRACT

In our previous research we designed *Jazzyk*, a modular programming language for development of cognitive agent systems. *Jazzyk* obeys two basic design principles: 1) it allows for an easy integration of heterogeneous knowledge representation technologies, and 2) draws a strict distinction between modeling agent's knowledge and reasoning vs. its behaviours.

To further drive the development of *Jazzyk*, we implemented *Jazzbot*, a softbot embodied in a simulated 3D environment of a computer game *Nexuiz*. The core of *Jazzbot*'s belief base is implemented as a logic program interpreted in the semantics of *Answer Set Programming*, thus exploiting the power of non-monotonic reasoning. It is complemented by a *Ruby* language module for representing the bot's topological knowledge about the environment. *Jazzbot* thus demonstrates the synergistic effect of using heterogeneous, in this case declarative and object-oriented, KR technologies in a single agent system.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Intelligent Agents*; I.2.11 [**Artificial Intelligence**]: Miscelaneous—*Virtual Agents*

Keywords

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

1. INTRODUCTION

In order to exploit a synergy of various heterogeneous KR approaches in single agent systems, in [2] we proposed a modular agent programming framework of *Behavioural State Machines* (*BSM*). As a proof of the concept we subsequently implemented *Jazzyk*, a full featured interpreter[1] for *BSM*.

To further drive the development of *Jazzyk* and study its applications, we implemented *Jazzbot*, a virtual agent embodied in a simulated 3D environment of a first-person shooter computer game *Nexuiz*[2].

This paper describes the architecture of and technologies employed in *Jazzbot* implementation. Section 2, briefly sketches the underlying framework of *Behavioural State Machines*. Without going into details, in Section 3, we subsequently provide an overview of the *Jazzbot's* design, the KR technologies employed and the agent's interaction with the environment. Section 4 concludes this report by a brief discussion of the related work and outlooks for future development of *Jazzyk* and its applications.

2. BEHAVIOURAL STATE MACHINES

The theoretical framework of *Behavioural State Machines* is a current evolution of *Modular BDI Architecture* framework, we proposed in [3]. *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 ruleOur project follows the spirit of [1], where Laird and van Lent argue that approaches for programming intelligent agents should be tested in realistic and sophisticated environments of modern computer games. s. 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 (KB) of the agent. Updates and rules form basic *mental state transformers* (mst's), *Jazzyk*'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 *BSM* framework and in turn *Jazzyk* language, 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. 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. Transi-

[1]The first version of *Jazzyk* interpreter together with a software development kit for *Jazzyk* modules was published under GNU GPL license at `http://jazzyk.sourceforge.net/`.

[2]`http://www.alientrap.org/nexuiz`

tions 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. THE BOT IMPLEMENTATION

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 module. The interface to the environment is facilitated by a *Nexuiz* game client module.

Answer Set Programming module is realized by a *Jazzyk* module which integrates an ASP solver *Smodels* [5]. Hence the syntax and the semantics of logic programs the module handles, i.e. query/update formulae, is that of *Smodels*. Query formulae query the answer sets (stable models) of the actual logic program in the knowledge base. The ASP KR module implements only a naive LP update mechanism.

As we note in [2], different KR tasks require different KR technologies. Therefore we chose an interpreted object-oriented programming language *Ruby*[3] for representation of topological knowledge about the environment. The *Ruby* module features a simple query/update interface allowing evaluation of arbitrary *Ruby* expressions.

The environment, *Jazzbot* operates in, is provided by a remote *Nexuiz* server. *Nexuiz* is an open-source 3D first-person shooter computer game based on the *Quake DarkPlaces*[4] engine. The *Nexuiz* KR module implements a client functionality and facilitates the bot's interaction with the game server. *Jazzbot* can exploit several virtual sensors: gps, sonar, compass, surface sensor and health status sensor, as well as effectors of its virtual body allowing it to move, jump, turn, use an item, attack, or utter a plain text message. The sensory interface is designed so, that it provides only a strict subset of the information of that a human game player can perceive.

Jazzbot's behaviours are implemented as a *Jazzyk* program. *Jazzbot* can fulfill e.g. search and deliver tasks in the simulated environment while it avoids obstacles and walls. Figure 1 depicts the architecture of *Jazzbot* and features a *Jazzyk* code chunk implementing a simple behaviour of picking up an object by mere walking through it and then keeping notice about it in its ASP belief base.

4. DISCUSSION AND RELATED WORK

Our project follows the spirit of [1], where Laird and van Lent argue that approaches for programming intelligent agents should be tested in realistic and sophisticated environments of modern computer games. Similarly to [6], we put *Jazzyk* language to a test in such a challenging environment. Unlike other agent programming frameworks, *Jazzyk* allows easy integration of heterogeneous KR approaches in *Jazzbot*-like softbots. By replacing ASP KR module, such agents can serve as a test-bed for various other KR approaches in the context of intelligent agents.

[3] `http://www.ruby-lang.org/`

[4] `http://icculus.org/twilight/darkplaces/`

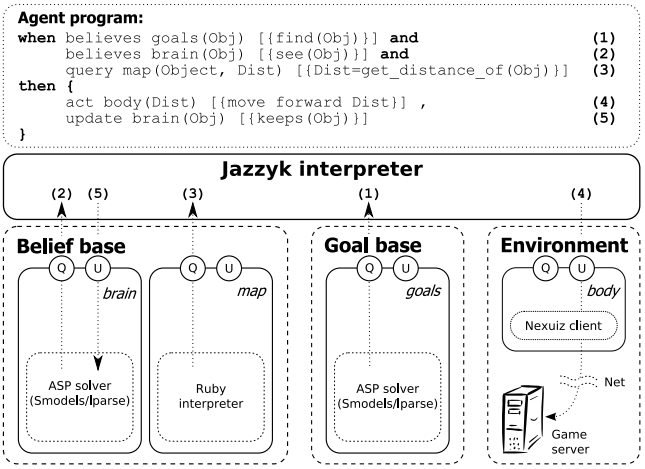

Figure 1: Scheme of *Jazzbot*

In [4], authors describe *Qsmodels* architecture based *Quake* bots implemented in plain ASP/*Smodels*. *Qsmodels* bots use planning as the primary approach to implementation of behaviours. We rather take a position that logic-based reasoning techniques are better suited for reasoning about static aspects of an environment, rather than for steering agents' behaviours. Unlike *Qsmodels* planning bot, *Jazzbot* is a rather reactive agent with strong support for deliberative features.

Currently we are working on a KR module providing inter-agent communication facilities using a standardized agent communication language. That will allow us to investigate implementations of teams of *Jazzyk*/*Jazzbot* agents.

5. REFERENCES

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

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

[3] 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.

[4] Luca Padovani and Alessandro Provetti. Qsmodels: ASP planning in interactive gaming environment. In José Júlio Alferes and João Alexandre Leite, editors, *JELIA*, volume 3229 of *Lecture Notes in Computer Science*, pages 689–692. Springer, 2004.

[5] Tommi Syrjänen and Ilkka Niemelä. The Smodels System. In Thomas Eiter, Wolfgang Faber, and Miroslaw Truszczynski, editors, *LPNMR*, volume 2173 of *Lecture Notes in Computer Science*, pages 434–438. Springer, 2001.

[6] Michael van Lent, John E. Laird, Josh Buckman, Joe Hartford, Steve Houchard, Kurt Steinkraus, and Russ Tedrake. Intelligent agents in computer games. In *AAAI/IAAI*, pages 929–930, 1999.