

## Logical Agents for Language and Action\*

**Martin Magnusson** and **Patrick Doherty**

Department of Computer and Information Science  
Linköping University, 581 83 Linköping, Sweden  
marma@ida.liu.se, patdo@ida.liu.se

### Abstract

Game developers are faced with the difficult task of creating non-player characters with convincing behavior. This commonly involves an exhaustive specification of their actions in every conceivable situation that might arise. The process is laborious and the results do not apply in new and unforeseen circumstances. We present an agent architecture where game characters have a basic understanding of their environment that enables them to figure out what to do by themselves. Knowledge about facts and actions is encoded in a formal logic where automated reasoning technology can apply the knowledge to answer novel questions in dialog and to plan actions in new situations. A number of examples serve to demonstrate how the technology meets the challenges of application in a simple computer game prototype. We envision this technology being used to create more flexible game characters with a deeper understanding of the world they inhabit.

### Introduction

The behavior of most non-player characters (NPCs) in computer games relies on the developers' foresight. Game play programmers meticulously design finite state machines that specify what NPCs should do in every state they can end up in, script writers painstakingly construct dialog trees that completely determine all possible conversational exchanges, and game designers carefully craft quests where the NPCs play out predefined scenarios. But a hallmark of intelligence is the ability to deal with the unexpected. Unforeseen situations pose problems for state machines, novel questions are impossible in scripted dialog, and dynamic scenarios call for the consideration of alternative courses of action.

A few games have successfully attacked one or more of these challenges using artificial intelligence technology. One example is the application of a planning algorithm, reminiscent of the classic STRIPS planning framework, in Monolith's game F.E.A.R. (Orkin 2005). Enemy soldiers plan sequences of actions for taking cover and attacking the player instead of relying on finite state machines. The tight

time constraints enforced by the real-time combat situation necessitated a very limited planning model, but it is nevertheless a demonstration of feasibility.

EA's game series *The Sims* features characters that display a broader range of autonomous action (EA 2007). Part of the game's appeal is the player's ability to modify the environment and see how this causes the Sims to modify their behavior. This behavior is reactive, in response to the characters' needs and desires, rather than being based on a planning framework. The game adds an element of dialog, which is an important behavior that Sims engage in. But the sounds that they utter have no content or meaning, and the player has no means of participating in any conversation.

This is possible to some degree in the game *Creatures* from Millennium Interactive (Grand, Cliff, & Malhotra 1997). The player is able to teach life-like creatures with neural network "brains" limited language skills and to watch them reproduce and evolve over generations through genetic algorithms. *Creatures* successfully manages to build an entertaining game using advanced artificial life technologies.

Academia has approached the challenge of intelligent game characters from at least two different viewpoints. The *Oz* project (Bates 2002) and the Virtual Theater project (Doyle 2001) emphasized *interactive drama*. They attempted to provide programming languages and tools for creating dramatic stories that involve *believable agents* who appear intelligent to the player. This is an author intensive solution where character behaviors are scripted beforehand to ensure a high degree of control over the result.

In contrast, more traditional artificial intelligence research, such as the Soar architecture, emphasize agent autonomy. Soar is based on the use of production rules, which the Soar/Games project coupled to *Quake 2* and *Descent 3* (Laird & van Lent 1999). This allowed the researchers to build reusable rule bases for autonomous agent behavior.

Unlike Soar, which is not primarily a planning framework, *SquadSmart* (Gorniak & Davis 2007) demonstrates how hierarchical task network planning can be used for multi-agent planning in games. An incremental algorithm ensures bounded computation time and an execution module synchronizes actions and copes with failures.

Game contexts also provide excellent opportunities to study natural language understanding problems such as reference resolution in circumscribed environments (Gorniak,

---

\*This work is supported in part by the National Aeronautics Research Program NFFP04 S4203, CENIIT, and the Strategic Research Center MOVIII, funded by the Swedish Foundation for Strategic Research, SSF.  
Copyright © 2008, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.









