Layered Disclosure: Why is the agent doing what it's doing?

Extended Abstract: full version available at www.cs.cmu.edu/~pstone/RoboCup/CMUnited99-sim.html

Peter Stone
AT&T Labs — Research
180 Park Ave., room A273
Florham Park, NJ 07932
pstone@research.att.com

Patrick Riley
Computer Science Dept.
Carnegie Mellon University
Pittsburgh, PA 15213
pfr@cs.cmu.edu

Manuela Veloso
Computer Science Dept.
Carnegie Mellon University
Pittsburgh, PA 15213
veloso@cs.cmu.edu

1. INTRODUCTION

A perennial challenge in creating and using complex autonomous agents is following their choices of actions as the world changes dynamically, and understanding why they act as they do. Our work focuses on environments in which agents have complex, possibly noisy, sensors and actuators. In such scenarios, even the human who develops an agent is often unable to identify what exactly caused the agent to act as it did in a given situation. This paper reports on our work to support human developers and observers to better follow and understand the actions of autonomous agents.

To this end, we introduce the concept of *layered disclosure* by which autonomous agents include in their architecture the foundations necessary to allow them to disclose to a person upon request the specific reasons for their actions. The person may request information at any level of detail, and either retroactively or while the agent is acting. We say that the person is *probing* for information, while the agent is explicitly *disclosing* the information.

A key component of layered disclosure is that the relevant agent information is organized in *layers*. In general, there is far too much information available to display all of it at all times. The imposed hierarchy allows the user to select at which level of detail he or she would like to probe into the agent in question.

Layered disclosure has two main uses: (i) as a debugging tool for agent development in complex environments, and (ii) as a vehicle for interactive agent control.

When an agent does something unexpected or undesirable, it is particularly useful to be able to isolate precisely *why* it took such an action. Using layered disclosure, a developer can probe inside the agent at any level of detail to determine precisely what needs to be altered in order to attain the desired agent behavior.

The fact that layered disclosure also works in real time means that a user can monitor the internals of an agent as it acts. When coupled with an interface for influencing agent behaviors, layered disclosure could be used to allow the interleaving of autonomous and manual control of agents.

We implemented layered disclosure in the simulated robotic soccer domain. It played an important role in our successful development of the 1999 RoboCup simulator-league champion CMUnited-99 team [1].

2. LAYERED DISCLOSURE

In developing layered disclosure, we began with the assumption that agents' actions and the actual states of the world over time are generally observable. On the other hand, several agent characteristics are generally unobservable, including the agents' sensory *perceptions*; the agents' *internal states* (current role in team, current task assignment, etc.); the agents' *perceived current world states*; and the agents' *reasoning processes*. The goal of layered disclosure is to make these unobservable characteristics observable, either retroactively, or in real-time as an agent is acting. Furthermore, to avoid being overwhelmed with data, the observer must be able to probe into the agent at an arbitrary level of detail or abstraction.

There are four main steps to realizing this goal.

1. The developer must organize the agent's perception-cognition-action process in different levels of detail.
2. The agent must store a log of all information from its internal state, world model, and reasoning process.
3. This log must be synchronized with a recording of the observable world (or generated in real-time).
4. An interface is needed to allow the developer to probe a given agent's internal reasoning at any time and any level of detail.

2.1 Layered Organization

The first step to implementing layered disclosure is the generation of a layered organizational structure of information to be stored. This organization could include an operator hierarchy such as that used in hierarchical task network (HTN) planning [2]. However, it should also include information beyond the agent's reasoning process such as its perceptions, its internal state, and anything else that might influence its decision-making.

In particular, not all relevant information is necessarily present in an agent's action trace. For example, if an agent executes action with precondition $x > 45$ at time t, then one can conclude that at time t the variable x had a value greater than 45. However, one might want to know exactly what the value of x was at time t and how it got to be so. $x = 46$ may be a qualitatively different symptom from $x = 10,000$. Layered disclosure allows one to probe into an agent to determine the value of x at time t, and, if probing more deeply, the agent's exact sensory perceptions and/or past states which caused x to be set to its actual value at time t.

A possible coarse breakdown of an agent's perception-cognition-action process is as follows, where layers with greater numbers represent information that is stored at a deeper layer (scale 1-50):

Levels 1–10: The agent's abstract high-level goals.
Levels 11–20: The agent's action decisions, perhaps organized hierarchically.
Levels 21–30: Lower-level details regarding the specifics of these actions including all variable bindings.
Levels 31–40: The agent's internal state.
Levels 41–50: The agent's sensory perceptions.

2.2 The Log File

Each agent's log file is its repository for all useful information that one might want to examine, either as the agent is acting or after the fact. It is generated and stored locally by the agent, or sent directly to the interface for immediate display. Each bit of information is time-stamped and tagged with its corresponding layer indicating its depth within the agents' reasoning process. The interface program can then display this information appropriately.

In general, the log file is of the format
`<time> (<level ind>) <Text>`
where

`<time>`	is the agent's conception of the current time.
`<level ind>`	is the information's layer.
`<Text>`	is arbitrary agent information to be displayed.

Note that in a distributed or asynchronous environment, the agent does not in general have access to the "real world" or global time.

2.3 Synchronization

When doing layered disclosure in real time, the agent simply outputs the requested layer of information relating to the *current* moment. However, when doing layered disclosure after the fact, the agent's log files, which record what would otherwise be unobservable, must be synchronized with a recording of the *observable* world. That is, the human observer needs to be able to identify what point of the log file corresponds to a given point in the recording of the world. The more exact a correspondence can be determined, the more useful the disclosure will be.

In some domains, synchronization may be difficult to achieve. Without synchronization, layered disclosure can still be used retrospectively to understand what an agent was doing given its perception of the world. However, to understand the entire loop, that is to understand whether an agent's action was appropriate to what was really going on, and to access the agent's state from a recording of the world, synchronization is needed.

In order to make such synchronization possible, it is necessary to have a time-stamped recording with time-stamps that correspond to the time-stamps in the log files described above. In environments such that agents have accurate knowledge of the global system or real-world time, the synchronization can be based on this time. Otherwise, the agent must actively transmit its internal time in such a way that it is visible to the recorder, perhaps in conjunction with the agent's actions. It is then the job of the interface to display to the user both the recording and the requested layered disclosure information.

2.4 Interface

When using layered disclosure retrospectively, the interface should include some representation of the agents' actions in the form of a recording. When using layered disclosure in real time, the real world acts as this representation. In addition, the interface includes a method for the user to specify an agent (if there are multiple agents) and an information layer. The interface then displays the specified layer of information for the specified agent at the moment in question. The interface can also visually separate the layers to aid the human observer.

In general, there is far too much information available to display all of it at all times. The imposed hierarchy allows the user to select at which level of detail he or she would like to probe into the agent in question.

3. CONCLUSION

Layered disclosure is potentially applicable in any agent-based domain. While our application is in a multiagent environment, it is inherently implemented on an individual agent: each agent discloses its own internal state. In order to take advantage of layered disclosure, one only needs to provide support for agents to store and disclose their perceptions, internal states, perceived current world states, and reasoning processes; as well as (in the retroactive case) a method for synchronization between the agent's logfile and a recording of the world.

Layered disclosure has proven to be very useful to us in our development of simulated robotic soccer agents. We envision that layered disclosure will continue to be useful in such agent development projects, particularly with complex agents acting in complex, dynamic environments. We also plan to begin using layered disclosure in interactive semi-autonomous agent-control scenarios.

4. REFERENCES

[1] P. Stone, P. Riley, and M. Veloso. The CMUnited-99 champion simulator team. In M. Veloso, E. Pagello, and H. Kitano, editors, *RoboCup-99: Robot Soccer World Cup III*, Berlin, 2000. Springer Verlag.

[2] D. E. Wilkins. Domain-independent planning: Representation and plan generation. *Artificial Intelligence*, 22:269–301, 1984.