Matt Forbes
January 2011
CS 400 - AI Indep. Study
Chapter 10 Discussion

Knowledge Representation - Summary

Two main models of representing knowledge were presented: object-based and event-based. When an agent doesn't need a notion of time and intervals, and object-based representation (situation calculus) can be a more simple choice. In this model, there is a discrete number of situations possible, and the agent moves through them. Each situation has a set of fluents (predicates and value functions) that describe the environment at that time. Events are much more complicated and descriptive; an event can be abstracted to be "composed from aspects of some space-time chunk." An event can be a time interval, a place, a person, a statement, etc. Categorizing of events allows for creation of powerful ontological hierarchies, where a subevent is fully contained by its super (ex. Bellingham is a subevent of Washington, or yesterday is a subevent of last week.)

Correctly categorizing an event or object is very important to any knowledge base. Newly detected information can only be useful when it can be compared and classified according to what an agent already knows. An important decision about representing knowledge is how to deal with unknown information; should it be assumed false in the absence of concrete facts? In first-order logic, that is not the case, an explicit clause declaring it false is needed. On the other hand, most people operate under a closed-world assumption, making reasoning much less verbose.

Applications

Representing knowledge is just a component of an intelligent agent, albeit an important one. Information must be organized in a way that allows an agent to accurately describe and categorize new events and objects that it discovers. This should enable the agent to make informed decisions that will bring it closer to its goal. When its internal representation of the world is malformed, it could take the agent drastically off course.

Unmanned Mars Rover - When we send robots to explore distant planets, they need a way to learn about their environment to keep the machine safe long enough to find valuable data. The goals of a Mars rover agent might possibly be identifying minerals, creating a detailed map of a specified region, etc. Identifying minerals would require an ontology of how types of minerals are related and offer a way to categorize new data. Representing the objects on and the layout of the surface of the planet could be categorized in this ontology as well.

Language-Speaking Agent - An agent that can understand language would need a huge ontology describing now only how to form sentences, but also the subjects of those sentences. This hierarchy of knowledge would be constantly changing as new relationships

between objects, events were added to the knowledge base. If for example it knew about my family, it would know I have a brother Derek, a sister Marissa, mother Sue, and father Eric. Querying this agent for my siblings would require the agent to have a relationship such as brotherOf(Matt, Derek), sisterOf(Matt, Marissa), and also understand that brothers and sisters are considered siblings. So assuming the agent could make the connection that I have these two siblings, then it would need to know how to form sentences about its knowledge. A simple version might look at the "owner" of said relations, and have that be the subject of the sentence its creating. Furthermore, the predicate of the sentence is about Matt, and says that I "have" a brother and "have" a sister. The final sentence might be "Matt has two siblings, he has one brother and one sister."

Implementation

I wasn't sure what kind of example to implement that exemplifies knowledge representation. What I ended up writing is more about planning, but still required a simple knowledge base. GPS (General Problem Solver) is my implementation for this week. This work is not entirely original, after reading the thought process used to arrive at this solution (as well as the code to implement it) I thought it would be a good exercise to work through myself. Original code was by Peter Norvig in his book *Artificial Intelligence Programming*, mine is most likely extremely similar.

GPS uses what's called "means-ends" analysis; essentially works backwards, starting with the action that will satisfy its goal. Once it has picked the action that it needs to eventually make, it tries to get itself in a position (or state) that satisfies the pre-conditions of that action. Doing this in a recursive manor, continually satisfying conditions, eventually leads to the initial state or a dead-end. In the former case, an acceptable path has been found that solves the problem.

The agent's knowledge is the set of functions (actions) that form relations between world conditions. Each function has a set of pre-conditions that must be satisfied to perform it, and also a set of conditions that would be introduced/removed from the world after executing it. This forms the agents knowledge base; world conditions are related by actions that achieve them. This week's reading was more categorical, but the idea remains the same here.