AGENTS AND THEIR ENVIRONMENTS

Dr S.Srinivasan, Mr Vivek Jaglan, Dheeraj Kumar
HOD Computer Applications, PDM Bhadurgarh, India
dss_dce@yahoo.com
Assistant Professor, PDM Bhadurgarh, India
jaglanvivek@gmail.com
Assistant Professor, PDM Bhadurgarh, India.
Dheerajkumar47@gmail.com

ABSTRACT

The world itself is a very complex place for architecture agents to act in. The real world is partially observable, stochastic, sequential, dynamic and continuous. Most architecture is designed to only deal with fractions of the total possible environmental complexity by acting in particular domains. For example, some architecture assumes that the world is static and that the only things that change in the world are via an agent's actions. Other architectures may operate in dynamic environments but require that world be consistent or predictable. Some require a limited number of clearly defined percepts and actions. In some cases the experience is divided into atomic “episodes” and the choice of action in each episode depends only on episode itself. The links below briefly describe and define some of the possible environmental considerations made when developing cognitive architectures, mostly without direct reference to specific architectures.

Keywords: test bed, closed world assumption, Agent, Sensor, Actuator, Multi-agent system.

1 INTRODUCTION

An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuator [2]. A Vacuum cleaner agent is a reactive agent which means it responds when it senses dirt on the floor. Where as a Taxi Driver agent should behave in a different way since its environment is entirely different. It is a Goal directed agent. [5] It has to keep all the percepts it receives in its knowledge base. The architectures of these two typical examples should certainly differ greatly. Subsumption Architecture, ATLANTIS, Theo, Prodigy, ICARUS, Adaptive, Intelligent Systems(AIS) A Meta-reasoning Architecture for X’ (MAX), Homer, Soar, Teton, RALPH-MEA, Entropy Reduction Engine are twelve architectures we have considered for discussion so far environments are concerned [4]. The features of 12 architectures are given in a Table 2.

2 ENVIRONMENTS

2.1 Static Environments

In this section various environments with which the agents have to work are discussed. A static environment consists of unchanging surroundings in which an agent navigates, manipulates, or perhaps simply problem solves. The agent, then, does not need to adapt to new situations, nor do its designers need to concern themselves with the issue of inconsistencies of the world model within the agent itself. An example of such an environment is a simulated office setting, where the doorways and halls never change. Other static environments include those for simple problem solving like crossword, chess w/clock, poker, backgammon, eight-puzzle in which nothing changes except through the action of the agent.

2.1.2 Dynamic Environments

In a dynamic environment, the next state of environment is completely determined by the current state and the action executed by the agent. If design goal of an architecture is to create an agent that operates in a real world environment, it has to include some mechanisms that allow the agent to operate in a dynamic environment. Certainly there are real world environments that are not dynamic but static; but these are usually controlled situations and thus not representative of the full range of environments in which intelligent agent operates. Furthermore, there may be dynamic simulated environments in which an intelligent agent could be put to good use. Some of the examples are English tutor, part-picking, refinery controller.

Generally planning systems have trouble
dealing with dynamic environments. In particular, issues such as truth maintenance in the agent's symbolic world model and re-planning in response to changes in the environment must be addressed. In the planning-type architecture it self these capabilities have to be incorporated but it will not reactive due the complex sensory data with a world model. One approach to this problem is that the planning component may be dropped altogether as is done in subsumption-type architectures.

2.1.5 Real-World Environments

In order for agents to operate in the real world, they are normally designed to meet different criteria . Agents that operate in the real world require robust perception mechanisms and are often faced with dynamic and unpredictable environments and a higher degree of complexity than they might encounter otherwise. The Agent has to face numerous potential challenges in the real world. The agent's sensors and effectors may be imperfect, it may be required to produce new plans based on updated information very rapidly, and it might have to reason about the temporal aspects of its plans.

To avoid such problems, simulators may be used and this will help researchers to focus on higher-level cognitive functions such as learning and planning. However, it may be that the solutions to these lower-level problems need to arise from within the architecture rather than from outside of it, which would have a profound impact on the ultimate architecture design. If this is indeed the case then ignoring these issues is ultimately a disservice to the potential growth of the architecture.

By choosing to address the issues incumbent in acting in the real world it is also possible to draw insights into their interaction with each other .

2.1.6 Complex Environments

Both real and simulated environments can be very complex. Complexity in this case includes both the enormous amount of information that the environment contains and the enormous amount of input the environment can send to an agent. In both cases, the agent must have a way of managing this complexity. Often such considerations lead to the development of sensing strategies and attentional mechanisms so that the agent may more readily focus its efforts in such rich environments.

To build an agent for users of complex data access and management systems for resource and environmental applications. Gathering good examples of this highly specialized and complicated activity is costly and difficult. There is usually only a small set of such good examples available to guide the development of an agent. Consequently, agents are trained, rather than being learned inductively from example sets. One approach is that agents use planning and plan generalization (learning) as their basic mechanism. Plans for yet unseen combinations of goals are created by the merging of plans for individual goals, with the minimum of r e-planning. An example illustrates merging of existing plans, and shows a simple practical solution to the mutual goal clobbering problem. Plans are built from low-granularity agent commands.

2.1.7 Knowledge Rich

Many real and simulated environments are rich in detail and other information. The ability to incrementally add knowledge without significant slowdown is an important functionality for agents in such environments.
The richness and diversity of information can be difficult or impossible to capture during development, so learning is frequently employed to capture domain knowledge as the agent experiences its environment.

2.1.8 Input-Rich Environments
Sometimes a domain presents more perceptual information than an agent can even observe, let alone process intelligently. Additionally, it is important that such a (possibly continual) influx of perceptual data not overwhelm the agent and thus cause a degradation in its reactivity. However, it must respond to relevant information; otherwise, it may behave irrationally. Such considerations have driven the development of architectural mechanisms such as selective attention to manage this environmental complexity.

2.2 Environmental Effects on the Agent

2.2.1 Limited Resources
Generally an agent cannot possess infinite resources. There are some limitations in memory and processing capabilities. The limited computation resources available to the agent directly influence the types of processing it can afford to do. Given these limitations the agent may not be completely rational. However, it is desirable that the agent perform as well as it is capable, obeying some bounded rationality constraints.

2.2.2 Complete Knowledge
Sometimes an agent knows all possibly relevant information about its domain. In this case, learning is not required for domain understanding and the behavior of the system can be pre-coded, dependent on perceptions.

Associated with these environments is the closed world assumption, under which any fact not known to the agent can be taken to be false. This is similar to complete world knowledge, in that the agent knows everything that is true about its domain. This assumption greatly simplifies declarative representation tasks.

2.2.3 Unpredictable Environments
Sometimes dynamic environments may be unpredictable. This means that not only is the world changing but it changes in a way that the agent can not (fully) comprehend. This often occurs when an agent's representation of the world is incomplete (or non-existent). Because of this unpredictability, it may be desirable that the agent's processing be interruptible, to handle unexpected, and urgent contingencies.

A predictable environment is an environment for which an agent has an adequate (or perhaps complete) world model. For example, an agent that had a sophisticated, first-principles model of Newtonian physics could predict with reasonable accuracy the results of throwing an object of known mass and force. However, since such models are computationally prohibitive, most agents consider a dynamic world unpredictable as well.

This assumption does not hold for agent's that behave in simulated, dynamic worlds. Since those worlds can generally be predicted exactly these dynamic environments can be considered to be predictable.

2.2.4 Synchronous Environments
Events in a domain sometimes occur asynchronously with respect to the agent. In such a situation, if the agent does not constantly perceive its world, events may go unnoticed, leading to seemingly irrational behavior. In order to avoid this, architectures often shift to more parallel approach, in terms of the sensing strategies used.

2.2.5 Concurrent Environments
Multiple domain events can occur simultaneously. In such cases, it is important that the agent take actions appropriate to all relevant events. If it can only pay attention to some of the concurrent events, its rationality will suffer. Thus, many architectures use parallel methods in their sensing strategies.

2.2.6 Varying Environments
All of the events occurring in the environment may not demand the same level of attention from the agent. Agent has to pay greatest attention according to priority, so as to maintain salient behavior. For example a taxi driver agent should follow road and signals and not all in its way.

2.2.7 Limited Response Time
An agent rarely has an unbounded amount of time to take actions in response to an environmental event. This limits the amount of processing required before taking an action, and usually also limits the amount of knowledge brought to bear. As a result, much architecture turns to either interruptible processing or situated action. However, the agent must still act as rationally as possible, given the time allowed, according to some bounded rationality constraint.

2.2.8 Multiple Tasks
A domain may require an agent to perform many different types of tasks simultaneously in order to "survive". As a result, many architectures support multiple, simultaneous goals. When multiple tasks and goals may be considered
simultaneously, additional concerns include behavioral coherence and saliency.

2.2.9 Supervisors

Humans in the environment may be constantly monitoring the agent's performance. Often, they would like explanations of an agent's decisions, either for verification or debugging purposes. More commonly, they are able to add new knowledge directly to the agent's database. To facilitate this type of input, architectures often adopt a uniform, declarative style of knowledge representation. At the furthest extreme, both explanation and knowledge addition can take place in some natural language. [4]

In the Table 1, Rows indicate the environments and the column indicates the particular architecture. Y in the cell means that the architecture corresponding to column uses the environment represented by the row.

Table-2

<table>
<thead>
<tr>
<th>Sl No</th>
<th>Architecture</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Subsumption</td>
<td>Complicated intelligent behavior into simple behavior modules. Organized into layers.</td>
</tr>
<tr>
<td>2</td>
<td>THEO</td>
<td>Plan-Then-Compile. By this integrating learning, planning and knowledge representation.</td>
</tr>
<tr>
<td>4</td>
<td>PRODIGY</td>
<td>Storing the knowledge in a form of first order predicate logic (FOLP) called Prodigy Description language (PDL). Has a modular architecture that stores the knowledge symbolically.</td>
</tr>
<tr>
<td>5</td>
<td>ATLANTIS</td>
<td>Integrating planning and reacting in a</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Adaptive Intelligent System (AIS)</td>
<td>To reason about and interact with other dynamic entities in real time. --- problem solving techniques --- when encountering unexpected situation, decides whether to and how to respond. --- focus attention on the most critical aspects of current situation. --- operating continuously without rebooting. --- able to coordinate with external agent. (more or less similar to human being.)</td>
</tr>
<tr>
<td>7</td>
<td>Meta Reasoning (MAX)</td>
<td>Many ideas in MAX may traced to Prodigy. --- rule-based forward – chaining engine that operates on productions. --- is designed to support to modular agents. --- they are used to respond to a dynamic environment in a timely manner. --- modules are categorized in to Behavior and monitor. --- Some of the modules are: 1.attention focusing 2.multiple problem solving strategies 3.execution monitoring 4.goal-directed exploration</td>
</tr>
<tr>
<td>8</td>
<td>HOMER</td>
<td>--- Is not designed for general intelligence. --- underlying philosophy is to synthesize several key areas of AI to form one complete system. (like planning, learning, natural language understanding, robotic navigation). HOMER answers questions posed by users and carries out instructions given by users. --- is a modular structure. It consists of: 1.memory 2.planner 3.natural language interpreter and generator 4.reflective processes 5.plan executer</td>
</tr>
</tbody>
</table>
|9  | SOAR | --- originally known as STATE OPERATOR AND RESULT. --- main goal is that full range of capabilities to be handled by an intelligent agent from highly routines to extremely difficult open-problems --- the underlying SOAR architecture is the view that symbolic system is necessary and sufficient condition for general intelligence. This is known as Physical Symbolic system Hypothesis (PSSH) --- ultimate aim is to
get general intelligent agent --- is based on a production system i.e. It uses explicit production rules to govern its behaviors .

### 10 Teton
--- is a problem solver --- uses two memory areas
1. Short-Term memory
2. Long-Term memory
--- like human beings , interruption are allowed . --- it has a feature called Execution Cycle which always look for what to do next .

### 11 RALPH-MEA
--- is a multiple execution architecture --- like human being , selecting best one from the environment --- RALPH – MEA uses Execution Architecture (EA) to select from one state to best one . --- it uses the following :
1. Condition action
2. Action utility
3. Goal – based
4. Decision Theoretic

### 12 Entropy Reduction Engine (ERE)
--- focuses on problems that require planning , scheduling and control --- uses many different problem solving methods such as :
1. problem reduction
2. temporal projection
3. rule-based execution

## 3 CONCLUSION

This paper has tackled the question how a developer can choose among the many development options when implementing an agent application. One key aspect here is to understand that agent technology currently offers many problem specific solutions that address only certain types of application domains. We argue that one important foundation for making accurate choices is the availability of well defined and comparable surveys and evaluations of artifacts such as environment and capabilities. Therefore, we have in a tabular form for evaluating many kinds of Architectures with respect to environments . In future work we want to employ the two tables to study Multi-Agent System Technology. The idea is to Integrate state-of-the art AI techniques into intelligent agent designs, using examples from simple, reactive agents to full knowledge-based agents with natural language capabilities and so on. This leads to the study of Multi-Agent systems and its applications. In depth analysis of various Agent architectures is to build a Multi Agent System that will be suitable for our future work on Supply Chain Management.

## 4 REFERENCES


