International Journal of Scientific Engineering and Research (IJSER)
www.ijser.in
ISSN (Online): 2347-3878, Impact Factor (2015): 3.791

Verification of Autonomous Cleaning Agents

K V Krishnam Raju[1], Sai Keerthi Kallam[2]

[1]CSE Department, SRKR Engineering College,
Bhimavaram, India

[2]CSE Department, SRKR Engineering College,
Bhimavaram, India

Abstract: ***Verification of Autonomous cleaning agents is committed to verify an autonomous cleaning agent, for instance area cleaning robots. It hovers over a range of steps from considering an initial example multi agent system to modeling it and verifying it both logically and functionally with respect to the world coordinates. We strived to study a simple autonomous cleaning system and tried to test it in ways possible and suggest the changes that could have been made to the existing system, so that it works better in world coordinates. This paper is all about verification of the existing system and a keen study of its functionality in detailed perspective regarding the languages used to simulate the AI engine and the justification of utility of Multi Agent systems.***

Keywords: Autonomous System, Agent Speak, BDI Approach, Verification, Multi Agent System

1. Introduction

The term Autonomous system is mostly found in the domain of Artificial Intelligence. So, any Analysis of an autonomous system starts with the clear description of Artificial intelligence and agent system [1] [2] [3] with categorization of the systems. The theory and development of computer systems able to perform tasks normally requiring human intelligence, such as visual perception, speech recognition, decision-making, and translation between languages is called artificial intelligence. Depending on the kind of input and the output rendered, general systems are classified as functional, reactive, behavioral systems. Our major concern is about the reactive systems.

Reactive systems are the systems that cannot adequately be described by the relational or functional views. The relational view regards programs as functions from an initial state to terminal state. Typically the main role of reactive systems is to maintain an interaction with their environment and therefore must be described in terms of their on-going behavior. Every concurrent system must be studied by behavioral means. Any artificial intelligence system is a subset of reactive systems. A complex class of systems which are subset of reactive system is an AGENT. An agent is a reactive system that exhibits some degree of independency. Here the system itself determines how best to achieve a task from the available choices and decides without human intervention.

The following are characteristics of agents which work under simulated environments, and also have a BDI [5] approach behavior. An agent should be capable of sensing the environment and have a base of possible actions that they can perform in order to modify their environment. An environment may be physical or software environment. A rational agent should have the properties of Autonomy, Proactive, Reactivity, Social ability.

2. Agent Speak and BDI Approach

Agent speak[4] is an agent oriented programming language. It is based on logic programming and the BDI [6] architecture for autonomous agents. The language was originally called agent speak language [8] [9], but became more popular as agent speak.

Agent speak was an abstract agent programming language aimed to help the understanding of the relation between practical implementations of the BDI [7] architecture such as procedural reasoning system. A procedural reasoning system (PRS) is a framework for constructing real time reasoning systems that can perform complex tasks in dynamic environments. It is based on the notion of intelligent agent using the belief-desire-intention software model. A user application is predominantly defined, and provided to a PRS system is a set of knowledge areas. Each knowledge area is a piece of procedural knowledge that specifies how to do something.

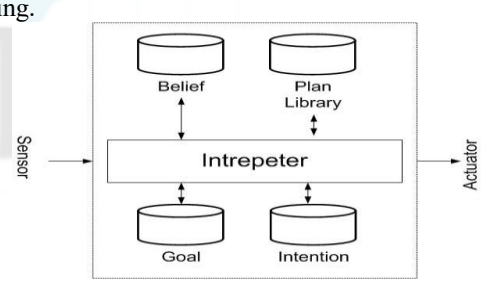

Figure 1: procedural reasoning system

The interpreter is responsible for maintaining beliefs about the world state, choosing which goals to attempt to achieve next and choosing which knowledge area to apply in the current situation. Unlike traditional AI planning systems that generate a complete plan at the beginning and re plan if unexpected things happen. PRS interleaves planning and doing actions in the world. PRS is based on the BDI or belief-desire-intention framework for intelligent agents.

The architectural components of a BDI system are:

Beliefs: Beliefs represent the informational state of the agent, in other words its beliefs about the world (including itself, other agents and environment). Beliefs can also include

inference rules, allowing forward chaining to lead to new beliefs. Using the term "belief" rather than "knowledge" recognizes that what an agent believes may not necessarily be true. Beliefs are stored in database.

Desires: Desires represent the motivational state of the agent. They represent objectives or situations that the agent would like to accomplish or bring out.

Goals: A goal is a desire that has been adopted for active pursuit by the agent. Usage of the term goals adds further restriction that the set of active desires must be consistent.

Intentions: Intentions represent the deliberative state of the agent what the agent has chosen to do. Intentions are desires to which the agent has to some extent committed. In implemented systems, this means the agent has begun executing a plan.

Plans: Plans are sequence of actions that the agent can perform to achieve one or more of its intentions. Plans may include other plans: my plan to go for a drive may include a plan to find my car keys.

Events: These are triggers for reactive activity by the agent. An event may update beliefs, trigger plans or modify goals. Events may be generated externally and received by sensors or integrated systems. Additionally, events may be generated internally to trigger decoupled updates or plans of activity.

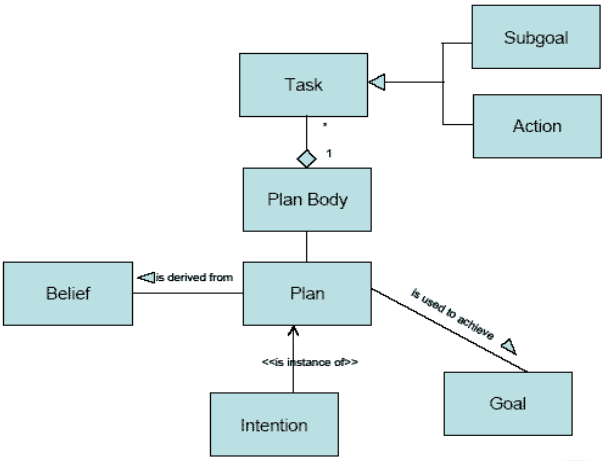

Figure 2: Relationship between Tasks, Plans, Beliefs, Intentions and Goals (in BDI)

3. Proposed Method

The work in this paper is committed to verify the Autonomous cleaning agents. It detects the garbage at a particular point of area in our environment. We have predefined precepts. For the by far existing application of a grid based cleaning environment.

Whenever garbage is found at a place or grid the bot automatically triggers the percept stored in its knowledge base to execute a particular action. The verification process will involve the logic verification and the functional verification. Before verifying any bot review system we need to execute the existing application on an available agent speak interpreter namely Jason and then we should have to look over for the different execution modes in different infrastructure modules.

The following are the basic percepts for the cleaning agents:
next(slot): The bot checks for slots to move to next. It moves to the next slots available, when prompted. It assumes that the space available is physical and when no other slots are available, the bot stands still for further percepts.

pick(garb): When a garb value is found at slot by the bot, it will trigger this percepts from the knowledge and then picks the garbage at the slot. The knowledge slot is all about verifying the bot is picking up the garbage or not.

drop(garb): After a bot picks up the garbage it triggers this percept to go the centralized garbage collector and drops the garbage.

burn(garb): It is defined on the centralized garbage collector agent to direct towards the disposal of the garbage and also towards the ideal mapping from the testing environment to the real world coordinates.

For the sake of logic verification we use the agent code interpreter and first create the following files for any application. mas1j file, which indicates the root of the whole project. asl files one for each of the agents involved in agent speak. java files which indicate the environment testing for the agent speak code.

All our logic verification takes place by linking the agent code and the environment code into the centralized project root file which will in turn synchronize all the agents and environment files into one loop.

In mas1j file we have options for the infrastructure which specifies the type of output verification we are opting for and also the jade and JacMO infrastructure for the sniffing activities of back ground internal actions taking place in the environment.

4. Results

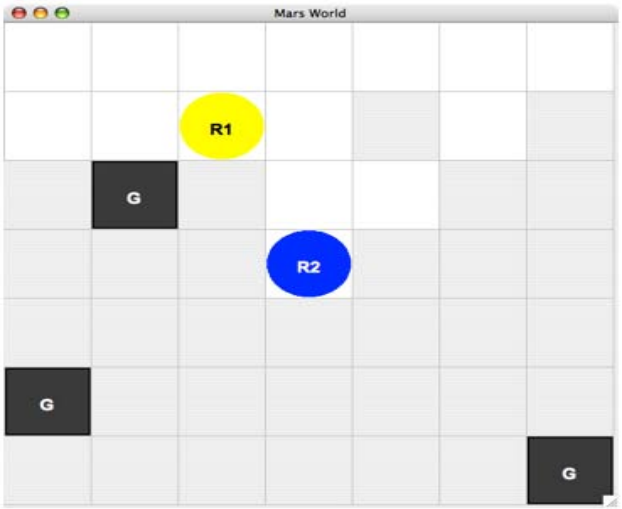

Figure 3: Environment output

This is the environment output which is used to sense the test logic and set the positions of the bots. Along with this we also have an MAS console which shows the test actions wise. Below is the test scenario with MAS console.

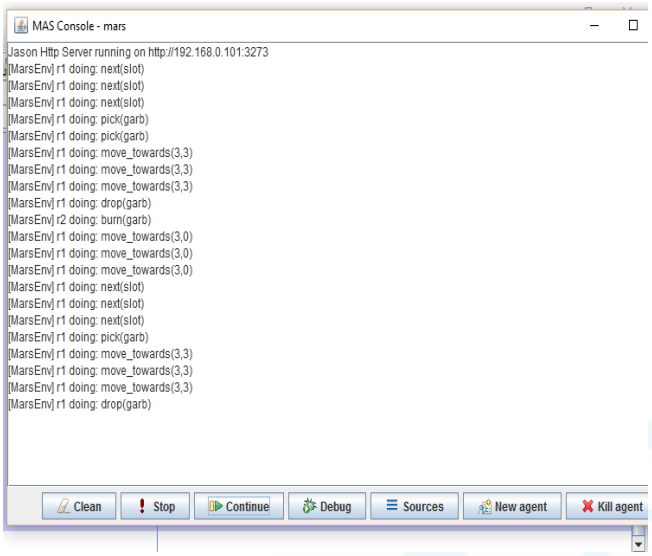

Figure 4: MAS console output

It clearly states the back drop events , which can logically verify the sequence of the agent actions taking place .And can compare whether it is working according to the real world scenario or not. And in addition to that we also have the Jade sniffer facility which is as shown below. we should have to go through specialized settings for to enable the Jade infrastructure settings.

We need to start the sniffer. Go to the plugin options then click on Jason. Followed by this is the lot of options for the sniffer agents and also the infrastructure need to be changed in the .mas1j file as jade.

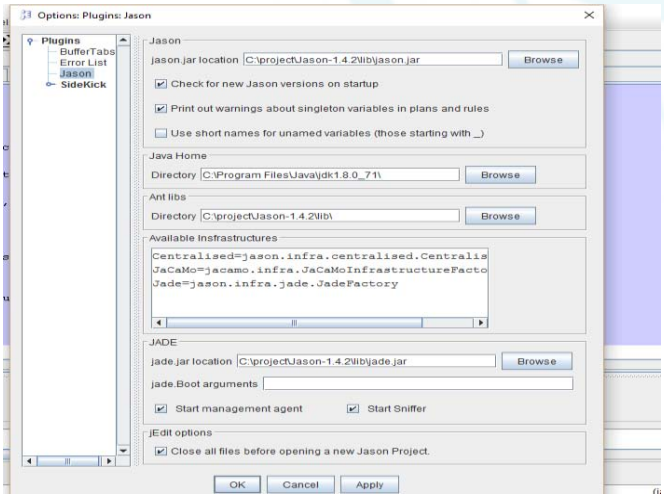

Figure 5: Jade Sniffer

With the jade facility we have time to time end back up of the internal actions that are taking place and also it helps in effective time and functional verification of the given application.

The BDI approach has an extended approach of verification.

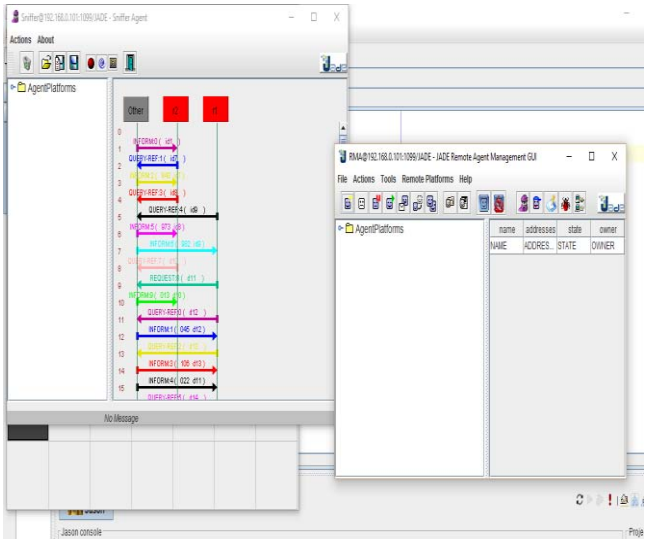

Figure 6: JADE Output

According to the AI engine written application it is only possible for two dimensional directional sense in a bot. If the same application is retrieved to work in a shape less global environment, it may not stick to the basic principles. Here it fails to achieve the reactive behavior with respect to world coordinates environment shape.

5. Conclusion and Future Work

Autonomous systems are that decide for themselves what to do and when to do it. They are currently making large impacts in a variety of applications, including driverless cars, unmanned aircraft and remote monitoring. It works for extended period without human interaction. Modern house hold, business and industrial systems use these systems. These systems are futuristic. These are used in systems that need to react much more quickly than humans can and in remote environments where direct human control is not possible. An autonomous robot perform tasks with high degree of autonomy which is particularly desirable in fields such as space exploration, house hold maintenance and delivering goods and services.

References

[1] Bordini, r., Dastani, m., Dix, J. and el fallah-Seghrouchni, a. eds. Multi-Agent Programming: Languages, Platforms and Applications. Springer, 2005.
[2] Bordini, r., Dastani, m., Dix, J. and el fallah-Seghrouchni, a. eds. Multi-Agent Programming: Languages, Tools and Applications. Springer, 2009.
[3] Bordini, r., fisher, m., Visser, w. and wooldridge, m. Verifying multi-agent programs by model checking. J. Autonomous Agents and Multi-Agent Systems 12, 2 (2006), 239–256.
[4] Bordini, r., hübner, J. and wooldridge, m. Programming Multi-agent Systems in AgentSpeak using Jason. wiley, 2007.
[5] Bratman, m. Intentions, Plans, and Practical Reason. harvard university Press, 1987.
[6] Dennis, l. and farwer, b. Gwendolen: a bDI language for verifiable agents. In Workshop on Logic and the Simulationof Interaction and Reasoning. AISB, 2008

[7] Dennis, l., farwer, b., bordini, r., fisher, m. and wooldridge, m. a common semantic basis for BDI languages. In Proc. 7th Int. Workshop on Programming Multiagent Systems, lNAI 4908 (2008). Springer, 124–139.
[8] Dennis, l., fisher, m., webster, m. and bordini, r. model checking agent programming languages. Automated Software Engineering 19, 1 (2012), 5–63.
[9] Rao, a. AgentSpeak(l): BDI agents speak out in a logical computable language. In Proc. 7th European Workshop on Modeling Autonomous Agents in a Multi-Agent World, LNCS 1038 (1996). Springer, 42–55.

Author Profile

K. V. Krishnam Raju received his B.Tech, M.Tech and PhD in Computer Science and Engineering from JNTU, Hyderabad, India and Nagarjuna University, India and JNTU, Kakinada, India. He is currently working as associate professor in the department of computer science and engineering in SRKR engineering college. His area of research interest includes Data Engineering, Network Security and e-commerce. He is a member of ACM.

Sai Keerthi Kallam received her B.Tech in Computer Science and Engineering from SRKR Engineering College India.