Multiagent Systems Introduction

José M Vidal

Department of Computer Science and Engineering
University of South Carolina

August 30, 2005

Abstract

An introduction to multiagent notation based on "Introduction to MultiAgent Systems" by Michael Wooldridge (2002) Chapters 1-2.

Trends

- Ubiquity.
- Interconnection
- Agent intelligence.
- Delegation.
- Human-orientation.
- Social computing.

Multiagent Systems

- An agent is capable of independent action.
- When many of them interact you have a multiagent system.
- Agents are everywhere, they talk to each other, they are not too stupid, they do things for us.

Open Questions

- How do we build agents so that the system operates as we want?
- Selfish agents represent better the human's selfish interest but, can they cooperate? How do we determine what are the proper incentives to give them?
- How do they communicate with each other?
- How do they negotiate, bargain, resolve conflicts, etc?
- How do we make it easy for software engineers (largely ignorant about this field) to build sophisticated systems?

Open Questions

- How do we build agents so that the system operates as we want?
- Selfish agents represent better the human's selfish interest but, can they cooperate? How do we determine what are the proper incentives to give them?
- How do they communicate with each other?
- How do they negotiate, bargain, resolve conflicts, etc?
- How do we make it easy for software engineers (largely ignorant about this field) to build sophisticated systems?

Open Questions

- How do we build agents so that the system operates as we want?
- Selfish agents represent better the human's selfish interest but, can they cooperate? How do we determine what are the proper incentives to give them?
- How do they communicate with each other?
- How do they negotiate, bargain, resolve conflicts, etc?
- How do we make it easy for software engineers (largely ignorant about this field) to build sophisticated systems?

Open Questions

- How do we build agents so that the system operates as we want?
- Selfish agents represent better the human's selfish interest but, can they cooperate? How do we determine what are the proper incentives to give them?
- How do they communicate with each other?
- How do they negotiate, bargain, resolve conflicts, etc?
- How do we make it easy for software engineers (largely ignorant about this field) to build sophisticated systems?

Open Questions

- How do we build agents so that the system operates as we want?
- Selfish agents represent better the human's selfish interest but, can they cooperate? How do we determine what are the proper incentives to give them?
- How do they communicate with each other?
- How do they negotiate, bargain, resolve conflicts, etc?
- How do we make it easy for software engineers (largely ignorant about this field) to build sophisticated systems?

Applications

- The Semantic Web; next generation of the web. Web services are just the start.
- Automated manufacturing. [Baker et al., 1999] [Huhns et al., 2002]
- Distributed sensor fusion. [Vidal, 2003] [Scerri et al., 2002] [Ortiz and Rauenbusch, 2002]
- Dynamic resource allocation (who does what). [Mailler et al., 2003]
- Supply-chain automation. [Singh and Huhns, 1999]
- Coordination of robots: Mars colonization. [Pirjanian et al., 2000]
- *Any application where independent localized units must coordinate to achieve some desired system behavior.*

Applications

- The Semantic Web; next generation of the web. Web services are just the start.
- Automated manufacturing. [Baker et al., 1999] [Huhns et al., 2002]
- Distributed sensor fusion. [Vidal, 2003] [Scerri et al., 2002] [Ortiz and Rauenbusch, 2002]
- Dynamic resource allocation (who does what). [Mailler et al., 2003]
- Supply-chain automation. [Singh and Huhns, 1999]
- Coordination of robots: Mars colonization. [Pirjanian et al., 2000]
- *Any application where independent localized units must coordinate to achieve some desired system behavior.*

Different Aspects of Multiagent Systems

Software Engineering

Concerned with technologies like XML, RDF, OWL, FIPA, SOAP, Web Services, Semantic Web, BPEL, WSDL, ontologies, etc.

Algorithms

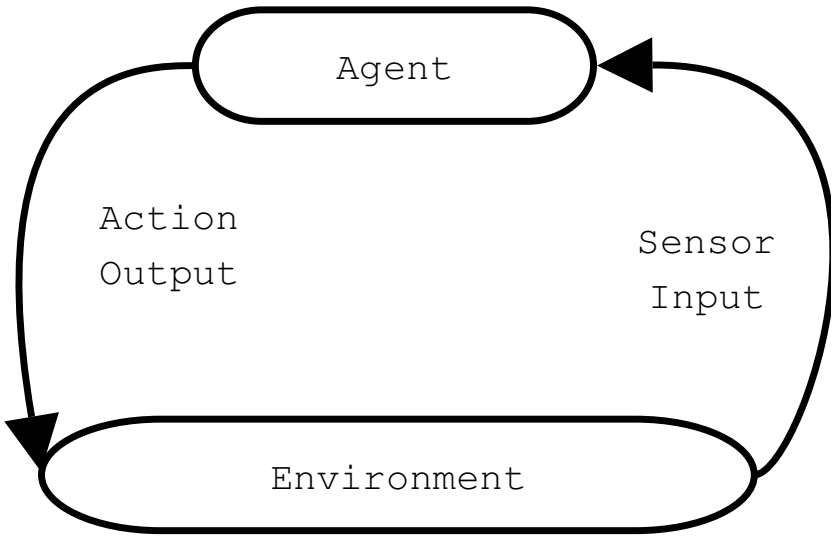
Composed of economists, sociologists, political scientists, game theorists, etc. Study concepts such as Nash equilibrium, tit-for-tat, ESS, stigmergy, dynamics, incentive-compatible design, etc.

- Multiagent systems brings these together.

Two Tracks in This Class

- This class will try to cover both the theory and applications.
- The best way to understand coordination protocols is by seeing them in action.
- We will cover most of the pertinent theoretical results.
- We will use NetLogo as a way to quickly implement and play with prototype multiagent systems.
- We do not cover web services implementation as that is another class.

An Agent



- "An agent is a computer system that is *situated* in some *environment*, and that is capable of *autonomous action* in this environment in order to meet its design objectives."
- Usually the agent only has partial control–actions might not have expected consequences.
- Control systems.
- Software demons.

Environment

- Accessible vs. inaccessible- can the agent "see" everything?
- Deterministic vs. non-deterministic- do actions have guaranteed effect?
- Static vs. dynamic- does the environment change on its own?
- Discrete vs continuous- is the number of actions and percepts finite? (Does it matter?)

Environment

- Accessible vs. inaccessible- can the agent "see" everything?
- Deterministic vs. non-deterministic- do actions have guaranteed effect?
- Static vs. dynamic- does the environment change on its own?
- Discrete vs continuous- is the number of actions and percepts finite? (Does it matter?)

Environment

- Accessible vs. inaccessible- can the agent "see" everything?
- Deterministic vs. non-deterministic- do actions have guaranteed effect?
- Static vs. dynamic- does the environment change on its own?
- Discrete vs continuous- is the number of actions and percepts finite? (Does it matter?)

Environment

- Accessible vs. inaccessible- can the agent "see" everything?
- Deterministic vs. non-deterministic- do actions have guaranteed effect?
- Static vs. dynamic- does the environment change on its own?
- Discrete vs continuous- is the number of actions and percepts finite? (Does it matter?)

Environment Issues

- Robotic environments are almost always inaccessible. Market systems are accessible, if we ignore the other agents' mental states.
- Many environments are deterministic but they are complex enough that it does not matter.
- Static environments are easier to deal with. Unfortunately, if there are other agents then they are likely changing the environment.
- In the end, perceptions and actions are turned into finite binary numbers, so everything is approximated by discrete math.

Intelligent Agents

- Reactivity- respond in a timely fashion when needed.
- Pro-activeness- satisfy internal goals. Take actions when it seems like they will be useful.
- Social ability- interact with other agents in order to satisfy goals.
- Pro-activeness and reactivity are often hard to trade-off.
- Social ability is exceptionally hard to implement. It requires both sides to cooperate.
- Agents in MASs are not intelligent in the general sense. They have simple specialized decision-making capabilities.

Agents and Objects

- We can and do use objects to implement agents, but they are different.
- Agents have reactive, proactive, and social behavior.
- Agents have their own thread of control.

How to Tell

Objects do it for free; agents do it for money.

Agents and Objects

- We can and do use objects to implement agents, but they are different.
- Agents have reactive, proactive, and social behavior.
- Agents have their own thread of control.

How to Tell

Objects do it for free; agents do it for money.

Abstract Architectures for Agents

- Environment E changes over time $e, e', \ldots$.
- An agent Ag has a set of possible actions $Ac = \alpha, \alpha', \ldots$
- A run r is a sequence of environmental states and actions $e_0 \xrightarrow{\alpha_0} e_1 \xrightarrow{\alpha_1} e_2 \ldots$
- R the set of all such possible runs.
- R^{Ac} be the subset of these that end with an action.
- R^E be the subset that ends with an environment state.
- $\tau : R^{Ac} \longrightarrow 2^E$ is the **state transformer** function which represents the effect of agents' actions.
- An Env is a triple (E, e_0, τ).

Abstract Architectures for Agents

- Environment E changes over time $e, e', \ldots$.
- An agent Ag has a set of possible actions $Ac = \alpha, \alpha', \ldots$
- A run r is a sequence of environmental states and actions $e_0 \xrightarrow{\alpha_0} e_1 \xrightarrow{\alpha_1} e_2 \ldots$
- R the set of all such possible runs.
- R^{A_c} be the subset of these that end with an action.
- R^E be the subset that ends with an environment state.
- $\tau : R^{A_c} \longrightarrow 2^E$ is the **state transformer** function which represents the effect of agents' actions.
- An *Env* is a triple (E, e_0, τ).

Abstract Architectures for Agents

- Environment E changes over time $e, e', \ldots$.
- An agent Ag has a set of possible actions $Ac = \alpha, \alpha', \ldots$
- A run r is a sequence of environmental states and actions $e_0 \xrightarrow{\alpha_0} e_1 \xrightarrow{\alpha_1} e_2 \ldots$
- R the set of all such possible runs.
- R^{A_c} be the subset of these that end with an action.
- R^E be the subset that ends with an environment state.
- $\tau : R^{A_c} \longrightarrow 2^E$ is the **state transformer** function which represents the effect of agents' actions.
- An *Env* is a triple (E, e_0, τ).

Abstract Architectures for Agents

- Environment E changes over time $e, e', \ldots$.
- An agent Ag has a set of possible actions $Ac = \alpha, \alpha', \ldots$
- A run r is a sequence of environmental states and actions $e_0 \xrightarrow{\alpha_0} e_1 \xrightarrow{\alpha_1} e_2 \ldots$
- R the set of all such possible runs.
- R^{Ac} be the subset of these that end with an action.
- R^E be the subset that ends with an environment state.
- $\tau : R^{Ac} \longrightarrow 2^E$ is the **state transformer** function which represents the effect of agents' actions.
- An Env is a triple (E, e_0, τ).

Abstract Architectures for Agents

- Environment E changes over time $e, e', \ldots$.
- An agent Ag has a set of possible actions $Ac = \alpha, \alpha', \ldots$
- A run r is a sequence of environmental states and actions $e_0 \xrightarrow{\alpha_0} e_1 \xrightarrow{\alpha_1} e_2 \ldots$
- R the set of all such possible runs.
- R^{A_c} be the subset of these that end with an action.
- R^E be the subset that ends with an environment state.
- $\tau : R^{A_c} \longrightarrow 2^E$ is the **state transformer** function which represents the effect of agents' actions.
- An Env is a triple (E, e_0, τ).

Abstract Architectures for Agents

- Environment E changes over time $e, e', \ldots$.
- An agent Ag has a set of possible actions $Ac = \alpha, \alpha', \ldots$
- A run r is a sequence of environmental states and actions $e_0 \xrightarrow{\alpha_0} e_1 \xrightarrow{\alpha_1} e_2 \ldots$
- R the set of all such possible runs.
- R^{A_c} be the subset of these that end with an action.
- R^E be the subset that ends with an environment state.
- $\tau : R^{A_c} \longrightarrow 2^E$ is the **state transformer** function which represents the effect of agents' actions.
- An Env is a triple (E, e_0, τ).

Abstract Architectures for Agents

- Environment E changes over time $e, e', \ldots$.
- An agent Ag has a set of possible actions $Ac = \alpha, \alpha', \ldots$
- A run r is a sequence of environmental states and actions $e_0 \xrightarrow{\alpha_0} e_1 \xrightarrow{\alpha_1} e_2 \ldots$
- R the set of all such possible runs.
- R^{A_c} be the subset of these that end with an action.
- R^E be the subset that ends with an environment state.
- $\tau : R^{A_c} \rightarrow 2^E$ is the **state transformer** function which represents the effect of agents' actions.
- An Env is a triple (E, e_0, τ).

Abstract Architectures for Agents

- Environment E changes over time $e, e', \ldots$.
- An agent Ag has a set of possible actions $Ac = \alpha, \alpha', \ldots$
- A run r is a sequence of environmental states and actions $e_0 \xrightarrow{\alpha_0} e_1 \xrightarrow{\alpha_1} e_2 \ldots$
- R the set of all such possible runs.
- R^{A_c} be the subset of these that end with an action.
- R^E be the subset that ends with an environment state.
- $\tau : R^{A_c} \longrightarrow 2^E$ is the **state transformer** function which represents the effect of agents' actions.
- An *Env* is a triple (E, e_0, τ).

Agent Model

- An agent is $Ag : R^E \rightarrow Ac$.
- Agents are functions that map from the history of the environment to an action.
- Agents are assumed to be deterministic.
- A system is a pair of an Ag and an Env.
- The set of possible runs in a system is $R(Ag, Env)$.
- So the sequence $(e_0, \alpha_0, e_1, \alpha_1, \ldots)$ is a run of Ag in $Env = (E, e_0, \tau)$.
- Two agents are said to be behaviorally equivalent wrt Env iff $R(Ag1, Env) = R(Ag2, Env)$.

Agent Model

- An agent is $Ag : R^E \rightarrow Ac$.
- Agents are functions that map from the history of the environment to an action.
- Agents are assumed to be deterministic.
- A system is a pair of an Ag and an Env.
- The set of possible runs in a system is $R(Ag, Env)$.
- So the sequence $(e_0, \alpha_0, e_1, \alpha_1, \ldots)$ is a run of Ag in $Env = (E, e_0, \tau)$.
- Two agents are said to be behaviorally equivalent wrt Env iff $R(Ag1, Env) = R(Ag2, Env)$.

Agent Model

- An agent is $Ag : R^E \rightarrow Ac$.
- Agents are functions that map from the history of the environment to an action.
- Agents are assumed to be deterministic.
- A system is a pair of an Ag and an Env.
- The set of possible runs in a system is $R(Ag, Env)$.
- So the sequence $(e_0, \alpha_0, e_1, \alpha_1, \ldots)$ is a run of Ag in $Env = (E, e_0, \tau)$.
- Two agents are said to be behaviorally equivalent wrt Env iff $R(Ag1, Env) = R(Ag2, Env)$.

Agent Model

- An agent is $Ag : R^E \rightarrow Ac$.
- Agents are functions that map from the history of the environment to an action.
- Agents are assumed to be deterministic.
- A system is a pair of an Ag and an Env.
- The set of possible runs in a system is $R(Ag, Env)$.
- So the sequence $(e_0, \alpha_0, e_1, \alpha_1, \ldots)$ is a run of Ag in $Env = (E, e_0, \tau)$.
- Two agents are said to be behaviorally equivalent wrt Env iff $R(Ag1, Env) = R(Ag2, Env)$.

Agent Model

- An agent is $Ag : R^E \rightarrow Ac$.
- Agents are functions that map from the history of the environment to an action.
- Agents are assumed to be deterministic.
- A system is a pair of an Ag and an Env.
- The set of possible runs in a system is $R(Ag, Env)$.
- So the sequence $(e_0, \alpha_0, e_1, \alpha_1, \ldots)$ is a run of Ag in $Env = (E, e_0, \tau)$.
- Two agents are said to be behaviorally equivalent wrt Env iff $R(Ag1, Env) = R(Ag2, Env)$.

Agent Model

- An agent is $Ag : R^E \rightarrow Ac$.
- Agents are functions that map from the history of the environment to an action.
- Agents are assumed to be deterministic.
- A system is a pair of an Ag and an Env.
- The set of possible runs in a system is $R(Ag, Env)$.
- So the sequence $(e_0, \alpha_0, e_1, \alpha_1, \ldots)$ is a run of Ag in $Env = (E, e_0, \tau)$.
- Two agents are said to be behaviorally equivalent wrt Env iff $R(Ag1, Env) = R(Ag2, Env)$.

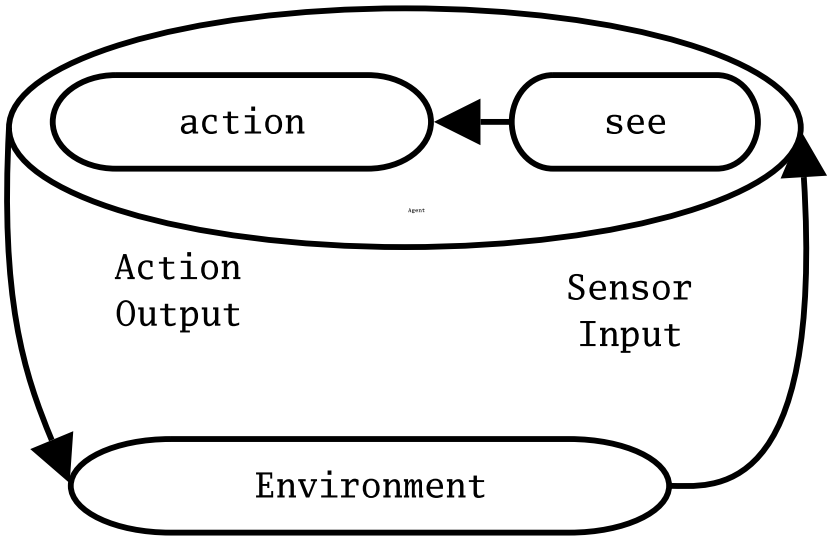
Agent Model

- An agent is $Ag : R^E \rightarrow Ac$.
- Agents are functions that map from the history of the environment to an action.
- Agents are assumed to be deterministic.
- A system is a pair of an Ag and an Env.
- The set of possible runs in a system is $R(Ag, Env)$.
- So the sequence $(e_0, \alpha_0, e_1, \alpha_1, \ldots)$ is a run of Ag in $Env = (E, e_0, \tau)$.
- Two agents are said to be behaviorally equivalent wrt Env iff $R(Ag1, Env) = R(Ag2, Env)$.

Reactive Agent

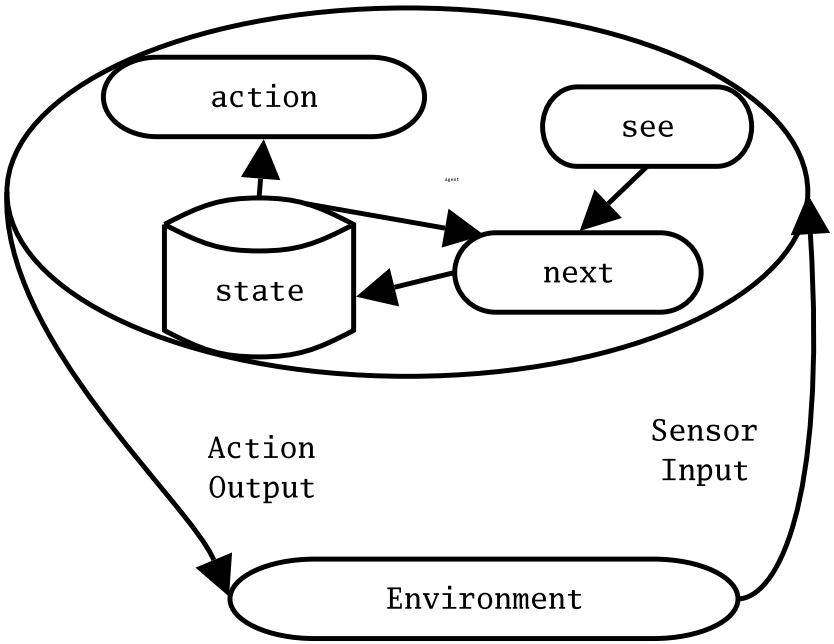
- We can now formally define a reactive agent as $Ag : E \rightarrow Ac$.
- Notice that it only considers one environmental state–the latest one.
- They can implement fewer behaviors than the standard agent.

Perception



- Usually the environment can have a very large number of states and the agent's perceptions can only capture part (inaccessible).
- We define see: $E \rightarrow Per$. Where Per is the set of percepts.
- We define action: $Per^* \rightarrow Ac$.
- What if e and e' get mapped to the same Per?

Agents with State



- Agents can maintain an internal state I and use it when determining what action to take.
- see: $E \rightarrow Per$.
- action: $I \rightarrow Ac$
- next: $I \times Per \rightarrow I$
- They have the same expressive power as standard agents, but they are easier to implement.

References I


Baker, A. D., Parunak, H. V., and Erol, K. (1999).
Agents and the internet: Infrastructure for mass customization.
IEEE Internet Computing, 3(5):62–69.


Huhns, M. N., Stephens, L. M., and Ivezic, N. (2002).
Automating supply-chain management.
In *Proceedings of the First Intenational Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1017–1024, Bologna, Italy. ACM Press, New York, NY.


Mailler, R., Lesser, V., and Horling, B. (2003).
Cooperative negotiation for soft real-time distributed resource allocation.
In *Proceedings of Second International Joint Conference on Autonomous Agents and MultiAgent Systems*, pages 576–583.

References II


Ortiz, C. L. and Rauenbusch, T. (2002).
Dynamic negotiation.
Technical report, SRI.


Pirjanian, P., Huntsberger, T. L., Trebi-Ollennu, A., Aghazarian, H., Das, H., Joshi, S. S., and Schenker, P. S. (2000).
CAMPOUT: a control architecture for multirobot planetary outposts.
In *Proceedings of SPIE*, pages 221–230.

References III


Scerri, P., Modi, P. J., Shen, W., and Tambe, M. (2002).
Applying constraint reasoning to real-world distributed task allocation.
In *Proceedings of Autonomous Agents and Multi-Agent Systems Workshop on Distributed Constraint Reasoning*, pages 134–141.


Singh, M. P. and Huhns, M. P. (1999).
Multiagent systems for workflow.
International Journal of Intelligent Systems in Accounting, Finance and Management.


Vidal, J. M. (2003).
A method for solving distributed service allocation problems.
Web Intelligence and Agent Systems: An International Journal, 1(2):139–146.