Lecture 1
Introduction and Course Outline

E. Haghverdi

January 8, 2002

The main motivation for the planning and offering of this course came from the recent advances in research at the interface of economics and computer science. This has been a new developement in the already very relevant and active field of multiagent systems. Moreover, the interaction between the fields of computer science and economics goes beyond the complexity analysis for economic mechanisms.

In this course we plan to study mainly the applications of logical techniques in the specification and reasoning about multiagent systems. Towards the end of the course we will have the opportunity to look at more recent developments alluded to above at the interface of CS and economics and will study other mathematical techniques used in such research.

There are three textbooks that we will be using for the major part of the course:

D : Multiagent Systems, A Modern Approach to Distributed Artificial Intelligence, edited by Gerhard Weiss, MIT Press, 1999.

K : Reasoning about Knowledge, R. Fagin, J.Y. Halperin, Y. Moses and M.Y. Vardi, MIT Press, 1995.

G : A Course in Game Theory, M. Osborne, A. Rubinstein, MIT Press, 1994.

Here is a tentative schedule for what we are planning to cover:

- Introduction to multiagent systems at micro and macro levels and formal methods: Chapters 1,2,8 of [D]
- Reasoning about knowledge: Chapters 1,2,4,5,6 of [K]
- Basic concepts in game theory: at least Chapters 1,2,3,5 of [G]
- Either game logics from a course by van Benthem OR Chapter 5 of [D] on distributed rational decision making.

Finally, to get a better perspective on the recent research at the interface of CS and Economics see the following special issues and editorials:

1. Monderer, Tennenholtz and Varian. Economics and AI,*Games and Economic Behaviour*, vol. 35, 2001.

2. Boutilier, Shoham and Wellman. Economic Principles of MAS, *Artificial Intelligence*, vol. 94, 1997.

1 Introduction

Distributed Artificial Intelligence (DAI) was developed starting at mid 1970's and has drawn from and contributed to many fields, including AI, computer science, sociology, economics, philosophy, etc. It can be defined as the study, construction and application of *Multiagent Systems* (MAS). We will make these terms more precise later. The importance of MAS is clear from the nature of computing in today's world: distributed, heterogeneous, open, large, dynamic, highly interactive. Just consider applications such as e-commerce and e-markets, telecommunications network management, Internet, modeling and optimization of of transportation systems to convince yourselves of the complexity of the situation and problems and need for satisfactory design and analysis methodologies for MAS.

The main task of DAI is to answer the question:

> "When and how should which agents interact-cooperate and compete- to successfully meet their design objectives?"

In trying to answer this question one has to address several issues, e.g.,

- Task decomposition and allocation to subagents, solution synthesis;
- Communication languages and protocols between agents;
- Representation of and reasoning about other agents' actions, plans and knowledge;
- Formal specification of systems and interaction between agents.

We will proceed with our study of multiagent systems at the agent level, that is micro level.

2 Micro Level

Definition 2.1 An *agent* is a computational entity that is situated in some environment and is capable of autonomous action in this environment in order to meet its design objectives.

We thus assume that each agent has ways of getting information about its environment, i.e. can *perceive* the environment and is capable of performing some actions. An important issue here is thus the *decision making* procedure of the agent which will let it decide which action to perform from among those that are enabled in order to satisfy its design objective.

In the definition above, by autonomy we mean that the agent can perform the actions on its own without any intervention from other agents or humans. In other words, each agent has control over its own actions.

The environment in which an agent is situated can be classified according to the following categories:

- Deterministic vs non-deterministic
- Accessible vs. inaccessible
- Episodic vs. non-episodic
- Static vs. Dynamic
- Discrete vs. continuous

Examples:

1. Any control system is an example, in particular consider a thermostat situated in a room, i.e. physical environment. Also assume that it has two percepts `cold` and `OK` which refer to temperature being cold and okay respectively. The decision making can be expressed with the following rules:

$$\texttt{cold} \longrightarrow \texttt{heating on}$$
$$\texttt{OK} \longrightarrow \texttt{heating off}$$

2. Software daemons like background processes in UNIX are examples of agents in a software environment. Consider for example the `xbiff` process in X windows under UNIX. It continuously monitors the incoming mail and does an action changing the environment, for example turning the mailbox to black, or beeping, etc. whenever there is an unread mail.

Definition 2.2 An *intelligent agent* in one that is capable of *flexible* autonomous action, where flexibility means:

- **reactivity:** perceiving and responding to changes in the environment

- **pro-activeness:** goal-directed behaviour
- **social ability:** interacting with other agents and humans, e.g. negotiation, cooperation, etc.

It is difficult to build systems that achieve an effective balance between goal-directed and reactive behaviour.

A digression: MAS and object oriented systems clearly have some common points, after all an object is a computational entity that encapsulates some state and performs actions or *methods* on this state and communicates by message passing. However, one can observe the following differences:

- agents have stronger autonomy: an object with a public method must execute that upon being asked to, an agent accepts requests for actions, from other agents and may decide not to do an action.
- intelligence, flexible behaviour is not part of the object oriented programming paradigm.
- agents have their own thread of control, so we naturally have concurrent control threads in a multiagent system.