Multi-Agent Systems (2)
分散協調ソフトウェア (2)

Topic 2: Introduction

(History of Computer/Agents, Definition of Agents)

Introduction and definitions

Purpose (shown before)

- This lecture focuses on the theory of multi-agent systems, including distributed search, multi-agent planning, multi-agent learning, auction/game theory, and voting.
- Purpose of this lecture is: to provide knowledge of foundation in this research field (Multi-Agent Systems and AI), so that students can read papers published in top-level international conferences and journals.
  - Origin of MAS study was very old in AI, but at the same time, this research field has received attention in recent years in computer science.
  - Because computer was stand-alone with limited communication capability before 90'es and even in 200X, the traditional theory and language were studied to understand programs run on the stand-alone computers for a long time. (so no one think collaboration among multiple programs)
  - But this situation has changed: nowadays computers obviously communicate and they sometimes collaborate.
  - But how to implement programs to collaborate, coordinate or compete with others like human. These are relatively new issues (from late 200X).

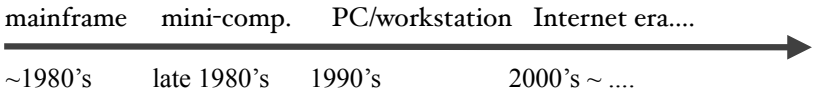
Brief History of Computer

mainframe mini-comp. PC/workstation Internet era....

~1980's late 1980's 1990's 2000's ~ ....

- In 1980s, Computer was stand alone and had only limited functions of communications (it could do only file transfer).
- However, programs gradually became large.
  - One great concern is how efficiently computers execute these large and complicated programs.
  - Parallel computers and distributed systems, --- that is, dividing into smaller pieces.
    - Parallelizing compiler
    - Inter-process communication (via BUS or network).
- Of course, these are still great concerns in computer science because recently we have great progress in this field (cluster, multi-core, etc....)

Brief History of Computer



- Another kind of applications in the Internet.
  - Smaller programs run independently in the Internet.
  - They may be developed by different developers/programmers/organizations.
  - They have different functionality and run on different machines (high- or low-end machines, PDAs, phones, game machines, etc...).
  - Services are created based on these programs.
- Each piece of program is not a part of a large program.
  - The program may work with other different, new (or sometimes unspecified) programs.

Direction of the new type of programs

- Five ongoing (recent) trends have marked the history of computing:
  - *Ubiquity*;
  - *Interconnection*;
  - *Intelligence*;
  - *Delegation*; and
  - *Human-orientation*.

Ubiquity

- The continual reduction in cost of computing capability has made it possible to introduce processing power into places and devices that would have once been uneconomic.
- As processing capability spreads, sophistication (and intelligence) becomes ***ubiquitous***.
- What could benefit from having a processor embedded in it…?
- e.g., IoT (The Internet of Things), CPS (cyber-physical systems)
- Ubiquity = 遍在性

Interconnection

- Computer systems today no longer stand alone, but are networked into large distributed systems
- The internet is an obvious example, but networking is spreading like ever-growing tentacles…
- Since distributed and concurrent systems have become the norm, some researchers are putting forward theoretical models that portray computing as primarily a process of interaction

Intelligence

- The complex tasks that we are capable of automating and delegating to computers has grown steadily.
- This means in our context (that) "intelligence" can automate something instead of doing by human, or it acts as a delegate of human.
- If you don't feel comfortable with this definition of "intelligence", it's probably because you are a human.
- Mechanism of intelligence — a dream.

Delegation

- Computers are doing more for us – without our intervention (if possible).
- We are *giving control* to computers, even in safety critical tasks.
  - One example: fly-by-wire aircraft, where the machine's judgment may be trusted more than an experienced pilot.
  - Next on the agenda: fly-by-wire cars, intelligent braking systems, cruise control that maintains distance from car in front…

Human Orientation

- The movement away from machine-oriented views of programming *toward* concepts and metaphors that more closely reflect the way we ourselves understand the world.
- Programmers (and users) relate to machines differently than they used to.
- Programmers conceptualize and implement software in terms of higher-level – that is, more human-oriented – abstractions.
  - This must be the result of computers becoming intelligent, and assuming human-like decisions and behaviors.

Direction of the new type of programs

- Five ongoing (recent) trends have marked the history of computing:
  - *Ubiquity*;
  - *Interconnection*;
  - *Intelligence*;
  - *Delegation*; and
  - *Human-orientation*.
- Note that all programs do not have all of these trend.
  - What are the common features that the recent programs (elements of applications) have.

Where does it bring us?

- Delegation and Intelligence imply the need to build computer systems that can act effectively on our behalf.
- This implies:
  - The ability of computer systems to act *independently*.
  - The ability of computer systems to act in a way that *represents our best interests* while interacting with other humans or systems.

Where does it bring us?

- Interconnection and Distribution have become core motifs in Computer Science.
- But Interconnection and Distribution, coupled with the need for systems to represent our best interests, implies systems that can *cooperate* and *reach agreements* (or even *compete*) with other systems that have different interests (much as we do with other people)
- Argumentation, Persuasion, Negotiation

Where does it bring us?

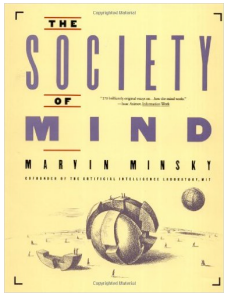
- For Japanese Students;
  - Cooperation: 協調
  - *Reaching agreements:* 合意形成
  - Argumentation : 議論
  - Persuasion: 説得
  - Negotiation: 交渉、ネゴシエーション

Core and Common Feature

- *Independence* (and so *autonomy*) is likely to be the common and basic features of these programs.
  - Cooperate to reach agreement (and often compete).
  - Work based on the best interest of human as delegations.
  - Furthermore, programs are independently developed by different developers.
- However, programs that have the features were not studied until recently.
  - This is the main difference from the studies of (conventional) distributed/concurrent systems.

Core and Common Feature

- *Intelligence* is the second common features all of these programs must have (to realize 'autonomy').
  - Intelligence is traditional research topic in AI but
  - Intelligence in a society is quite new topic.
    - Learning and doing their best actions by all agents in a society often produces frequent conflicts, and negotiations, resulting in an inefficient society.
    - Social rules (norms/conventions)
  - See Minsky's famous book : *Society of Mind.*

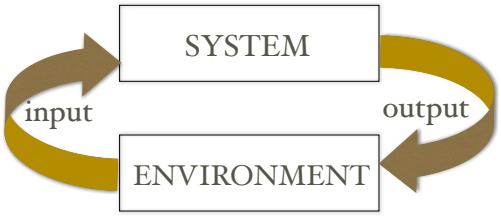


Core and Common Feature

- For Japanese Students
  - *Autonomy;* 自律性 *(autonomous,* 自律的な*)*
  - *Norms*: 規範、ノルム （数学用語と間違えない）
  - *Conventions*: 習慣

Agent, A definition

- The main point about agents is they are *autonomous*: capable of acting independently, exhibiting control over their internal state.
- Thus: an *agent* is a computer system capable of autonomous action in some environment in order to meet its design objectives.



Agent, Another definition

- More delegation-centered definition:
  - An agent is a computer system that is capable of *independent* action on behalf of its user or owner. (figuring out what needs to be done to satisfy design objectives, rather than constantly being told)

- Flavor of agents is autonomy (that is, independent, autonomous actions).
  - Not a part of a large program.
    - If it is a part of a large program, it is hard-encoded what should do next, or directly given "instructions to do next" by the main control components.

Autonomy of Agents

- The main point about agents is they are *autonomous*: capable of acting independently, exhibiting control over their internal states.

This is the meaning of autonomy of programs.

- Program that decides its actions as it is told by others (via network communications, inter-process communications, etc.....) is not autonomous.
- Program that decides its actions according to the contents of the internal memory.
  - Even if it may receive a message that requests doing a certain action, this information is just stored in its memory. Then it decides the best action to do next by looking at all contents in its memory. So the requested action may not be selected soon.
- This feature is the main flavor of agent.

Self-Interested Agents

- Discussion in the previous slide leads to the concept of self-interested agents.
  - What is "action to do next." What is the criteria of selecting actions.
- Self-interested Agents --- Characterizing agents.
  - Agent that decides its actions according to the local utility (preference, payoff values, etc.).
    - Basically, actions leading to its goal have higher utilities, and those which hinder the goal have lower (or minus) utilities.
    - If a number of agents have subgoals that are required to achieve more larger common goal, the achievement of local goal is important but, agent should not do actions making one of other subgoal unachievable (cooperative agents). So helping other agents may be one of interested actions.
    - How to define the utility is one of key issues.

Rational Agents

- Self-interested agent is *rational* if it acts to maximize its local utility value.
- However any agent can know only limited information.
  - To know everything is costly and computationally impossible, so agent cannot always take the best action.
  - The rationality of agent is often called *bounded rationality* due to this limitation.

Rational Agents

- For Japanese Students
  - self-interested: 利己的
    - ここでの訳としては良くない

      利己的 = selfish, egoistic
  - *rational:* 合理的、*rationality:* 合理性
  - *Bounded rationality* ：限定合理性

Multi-Agent System, A definition

- A multi-agent system is one that consists of a number of *loosely-coupled* agents, which *interact* with one another.
  - In the most general case, agents will be acting on behalf of users with different goals and motivations.
  - To successfully interact, they will require the ability to (of course, communicate) *cooperate*, *coordinate*, and *negotiate* with each other, much as people do.
  - The set of agents that interact with, so affect each other is called "*society of agents*."

Multi-Agent System, A definition

- For Japanese students
  - Coordinate: 調整
    - 日本語ではこれを協調と訳すこともあるが、CoordinationとCooperationは違うことに注意
    - Collaborationというのもある。

Multi-Agent Systems

- In Multi-agent Systems, we address questions such as:
  - How can cooperation emerge in societies of self-interested agents?
  - What kinds of languages can agents use to communicate?
  - How can self-interested agents recognize conflict, and how can they (nevertheless) reach agreement?
  - How can autonomous agents coordinate their activities so as to cooperatively achieve goals?

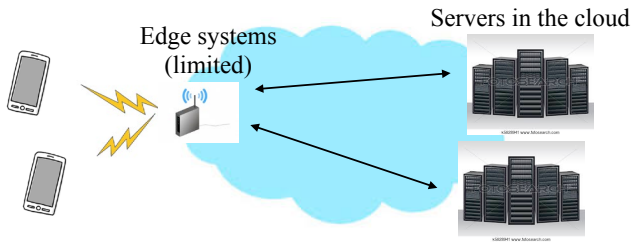
Multi-Agent Systems

- Cooperation, coordination, negotiation, agreement, and society, ...
  - While these questions are all addressed in part by other disciplines (notably economics and social sciences), what makes the multi-agent systems field unique is that it emphasizes that the agents in question are *computational, information processing* entities (that is, programs).
  - So they must be realized as algorithms, protocols or programs.

Multi-Agent System is Interdisciplinary

- The field of Multi-agent Systems (MAS) is influenced and inspired by many other fields:
  - Economics
  - Philosophy (not only MAS but also computer science)
  - Game Theory
  - Logic (AI and Database, etc)
  - Ecology
  - Social Sciences
- This can be both a strength (infusing well-founded methodologies into the field) and a weakness (there are many different views as to what the field is about).
- This has analogies with artificial intelligence itself.

Example: Game Theory (Economics)



- Accessing servers in the cloud induces many packets and causes some delay.
- Realtime applications (using the associated data) should be processed near users for interactivity and quick responses (and privacy).
  - Ex. Navigation, Recommendation, etc.
- Edge systems are near users but have the limited CPUs and storages.
- Other example: TCP/IP
- “Tragedy of the commons” （共有地の悲劇）

Some viewpoints

- Viewpoints of autonomous agents and multi-agent systems.
  - Micro-view (developing intelligent behaviors)
    - Relatively complex planning agents. Interaction with a few agents. (robots, vehicles, airplanes, mechanics, etc...)
    - Autonomous decision: How internally identify the best action
  - Macro-view (interference among agents):
    - How does local decision based on ‘self-interested’ness affect the entire (social) performance/efficiency
    - How do they reach common rules (norms/conventions) via experience to improve the social efficiency.
    - Key point: ‘Incentive to follow rules’ or ‘enabling agents to understand by learning that following rules is beneficial for each agent’ (trustiness, confidence)

History (appendix)

History: When did it start?

- The origin of the studies on multi-agent systems is The *Hearsay II Speech Understanding System.*
  - Developed at Carnegie-Mellon in the mid-1970's by L. Erman, F. Hayes-Roth, V. Lesser and D. Reddy.
  - Goal was to reliably interpret connected speech involving a large vocabulary
  - First example of the blackboard architecture, "a problem-solving organization that can effectively exploit a *multi-processor system*." (Fennel and Lesser, 1976)

Motivation of Hearsay II

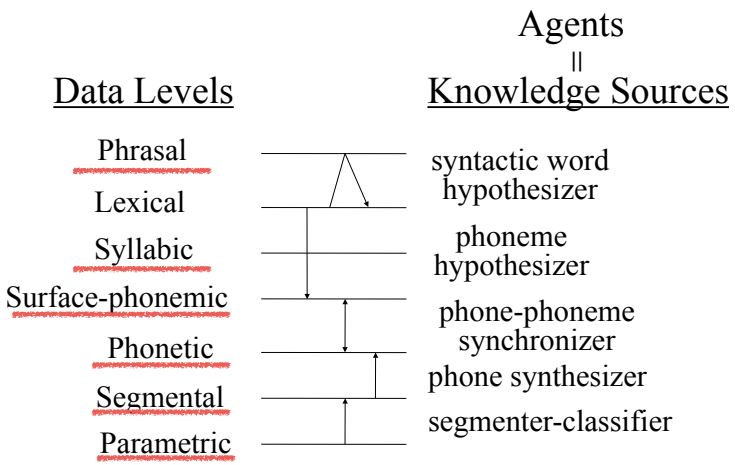
- Real-time speech understanding required more processor power than could be expected of typical machines in 1975 (between 10 and 100 mips); parallelism offered a way of achieving that power.
- There are always problems beyond the reach of current computer power—parallelism offers us hope of solving them now.
- The complicated structure of the problem (i.e., speech understanding) motivated the search for new ways of organizing problem solving knowledge in computer programs (Multi-processor machine in Carnegie-Mellon in the mid-1970's).
- But, *how to program??* (no language; no compiler; no techniques to develop/write programs for multi-processor machine;)

Result Sharing in Blackboard Systems

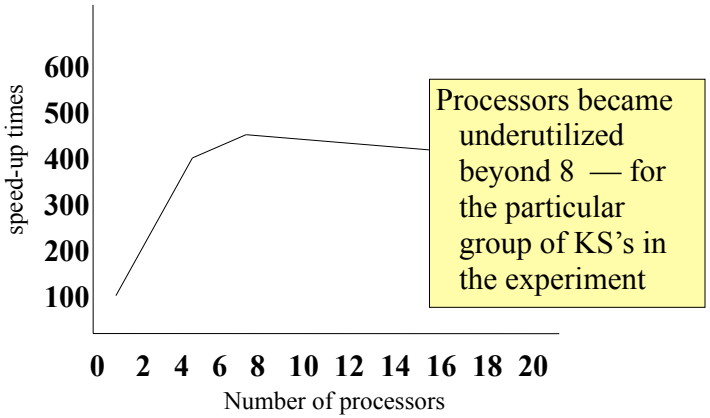
- The first scheme for cooperative problem solving: the blackboard (BB) system (a kind of shared memory).
- Multiple agents (KSs) can read and write to BB.
- Results shared via shared data structure.
- Agents write partial solutions to BB.
- BB may be structured into hierarchy.
- Mutual exclusion over BB required ⇒ bottleneck.

- Compare: LINDA tuple spaces, JAVASPACES.

Hearsay II Blackboard Organization



Effective Parallelism According to Processor Utilization



Lesson and Multi-Agent Systems

- The multi-processor implementation of Hearsay-II, with explicit synchronization techniques to maintain data integrity, achieved a speed-up factor of six — but the need for any synchronization techniques is a bad idea for a true distributed interpretation architecture...
- Two approach for efficiency: Pursue
  - Tightly coupling CPUs in a machine to reduce the cost of communications and access to memory. Parallel computer/ Multi-processor/ Multi-core.
  - True distribution like people.

Lesson and Multi-Agent Systems

- So for true distribution
  - Only a high-level communications (like human).
    - Loosely coupling
  - No (or less) synchronization (lock/unlock).
  - Independent and autonomous decision.
  - These suggested the multi-agent systems. (around 1978).
    - Victor Lesser started this research area.

End of Topic 2,
and Assignment No.1

宿題 1

- ある広い家を２人 (robot)で手分けして掃除をすることを考える。
  - 家の見取図は持っている。それぞれの部屋の広さ、家具やテーブルなど掃除の障害になりそうなものの配置は書いてあるが、形状などはわからない（形状が複雑だと掃除が面倒）。一つの部屋の複数人で清掃することも可能である。
  - 掃除は完全に終わらせないといけない。全体が完了して仕事の拘束から解放される。ただ、自分の作業量は少ない方がよいと考えている。

A. 問題１：人２人で作業を行うには、どのような協力方法があるか検討せよ。３人、４人と増えたときはどうなるか？

B. 問題２： 問題1の方法で数回作業し、各自は自分の経験のみから、汚れやすい箇所（あまり汚れないところは手抜きができる）や、障害物の形状による困難さを知る。この情報を持ち寄って、新たに共同作業する方法を考えて見よ。なお、経験の無い箇所の情報は、他人から聞くしか無い。しかし、かならずしも正しい情報を提供されるとは限らない。

C. 問題３：上記はプログラムで書けるでしょうか。理由や検討内容を示してください。

Assignment No.1

Consider cleaning a large house by two (or more) people (or robots).

- They have a sketch of the house. The size of each room and the arrangement of furniture, tables, and other objects that may hinder cleaning are described, but their shapes are unknown. (Complex shapes make cleaning cumbersome). It is also possible to clean a room by multiple people.
- The cleaning must be done completely. The whole is completed and freed from work constraints. However, all think that it's better to have less work.

[Q1:] Let's consider how two people can work together in a cooperative way. What happens when a few people increase if your method is applied

[Q2:] You worked several times with your method in Q1, and each person learns from her/his own experience about which places are likely to get dirty (you can cut corners where it is not so dirty) and the difficulty of the shapes of the obstacles. Let's consider new ways to work together using this information. Note that there is no other way to know but to ask others about the information of the place where you had no experience. But it doesn't always provide the right information.

[Q3:] If you have to write the agent program to implement coordinated behavior in Q1 and 2 without no centralized controller, what information is necessary to write this program ?

宿題・Assignment

- A4で１ページ以上、考えられる方法を日本語（英語）で書いてください。図などを利用することも可能です。提出のファイル形式はPDFとします。プログラムを書いて提出することは求めていません。デザインする上での検討項目を書いてください。
  - 提出の締切はMoodleを確認。Waseda Moodleで提出すること。

- Please write your answer in more than 1 page of A4 in Japanese (or English). You can use figures/diagrams. The file format of the submission must be PDF. I am not asking you to write and submit a program. Please write down the items to be considered in designing the program.
  - Deadline is shown in the Waseda Moodle. Upload it via Moodle.