COMP90054 Software Agents
Planning and Reasoning for Autonomy
A (slightly extended) introduction

Adrian Pearce, Nir Lipovetzky (Lecturers) &
Toby Davies (Tutor)

Outline

Outline

Software Agents: theory and practice

Autonomous agents: concerns automated reasoning and planning from the perspective of (possibly multiple) agents in the context of open, concurrent, non-deterministic environments

- Applications include logistics, mining, robotics, air traffic control, simulation and software agents on the Internet.

How do agents differ from objects?

Software Agents: theory and practice

Knowledge representation and reasoning: concerns the representation of knowledge and actions from the perspective of *agents* through interaction with their environment.

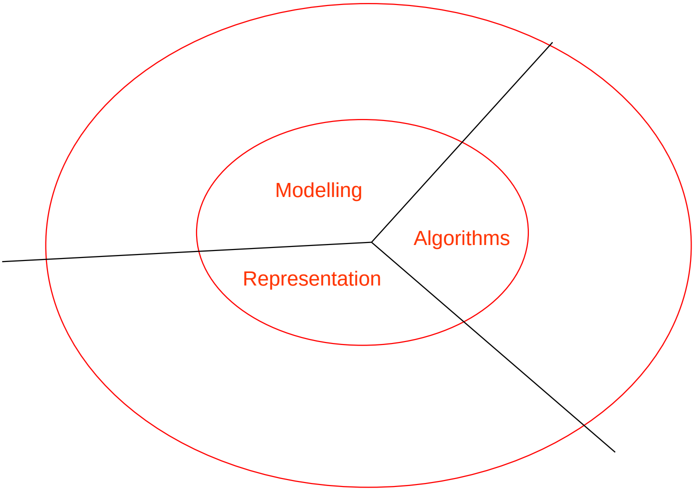
- Handling non-determinism, either through the occurrence of random events in the environment or by the actions of other agents that an agent has no direct control of, such as in human-robot interaction (HRI) for *autonomy*.

What is epistemic reasoning?

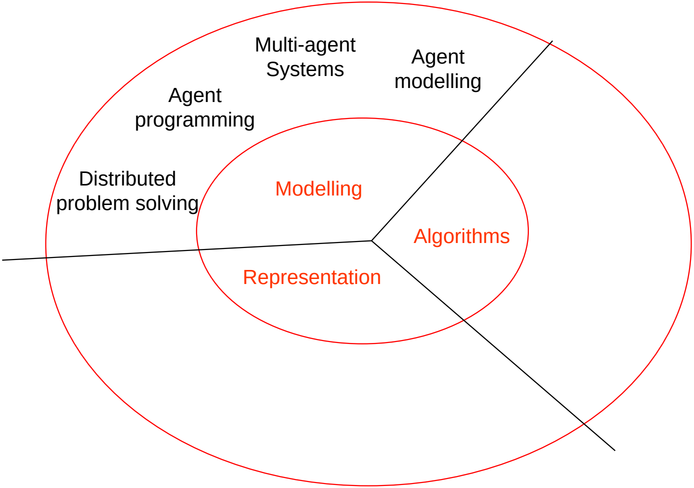
Software Agents: theory and practice

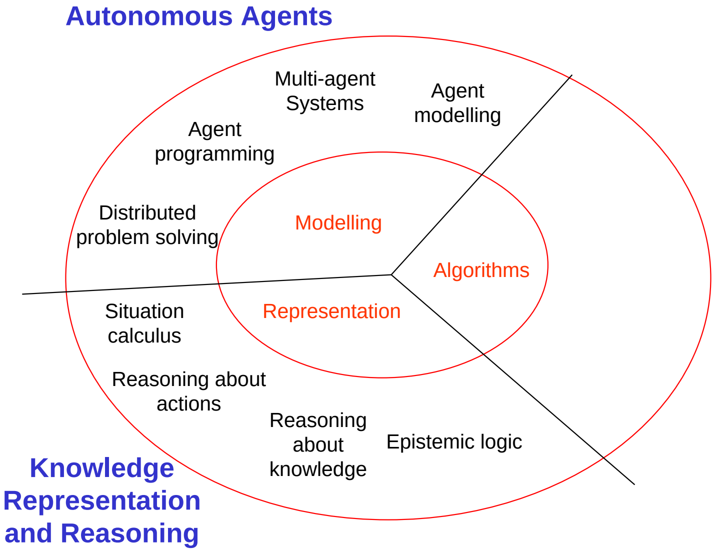
(Advanced) Artificial Intelligence: concerns the foundations, including novel data structures and algorithms for efficient and robust planning and reasoning about agents and their representations (subsumes both autonomous agents and knowledge representation and reasoning).

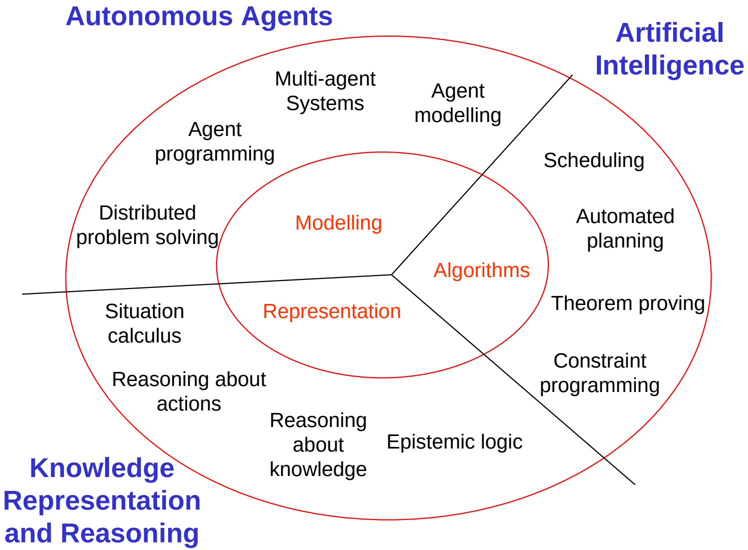
- AI research is increasingly utilised in the development of multi-agent programming and AI planning languages


Modelling
Algorithms
Representation

Autonomous Agents




Autonomous Agents
Multi-agent Systems
Agent modelling
Agent programming
Distributed problem solving
Modelling
Algorithms
Representation
Situation calculus
Reasoning about actions
Reasoning about knowledge
Epistemic logic
Knowledge Representation and Reasoning


Autonomous Agents
Artificial Intelligence
Knowledge Representation and Reasoning
Multi-agent Systems
Agent modelling
Agent programming
Distributed problem solving
Scheduling
Automated planning
Theorem proving
Constraint programming
Situation calculus
Reasoning about actions
Reasoning about knowledge
Epistemic logic
Modelling
Algorithms
Representation

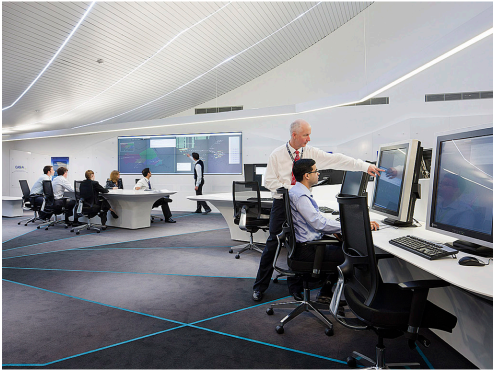
Planning research/application projects within CIS

- Air traffic management: verification of autonomous command & control functions (*Thales Australia*) (Research contract)
- Mining iron ore: optimising autonomous mining & production scheduling (*Rio Tinto Iron Ore* and the *Australian Research Council (ARC)* project)
- Human-agent interaction: foundations of autonomy, spectrum from full tele-operation to complete autonomy (ARC project)

Academics (Teaching & Research): **Adrian Pearce**; Peter Stuckey; Liz Sonenberg; Tim Miller; Chris Leckie; **Nir Lipovetzky**; Michelle Blom; Christina Burt; Paolo Felli & Christian Muise;

PhD Candidates **Toby Davies**; Chris Ewin; & Justin King

Command & Control: Air traffic management



- Thales' systems control nearly 50% of worlds airspace
- Highly distributed systems, typically 8 million lines of code
- Automation is a key issue, importantly, *software verification*

Photos Courtesy of Thales Australia

Application example: Mining iron ore

Photos Courtesy of Rio Tinto Iron Ore

Mines of the future

Context: Robotically operated mine sites, using a combination of autonomous trucks and tele-operation of drilling rigs

- Presently The largest commercial privately funded external robotics initiative in the world today.

Current multi-agent project—Making the Pilbara blend: agile mine scheduling through contingent planning, ARC Linkage Project (The University of Melbourne)

- Aim: Discover what can do with Intelligent Agent Technology for achieving agile mine scheduling
- Technology: integration of multi-agent automated planning techniques with constraint solving techniques.

Agent = action theory + plan and/or program

Underlying Syllabus:

- An action theory: the agent knows the theory and its consequences (actions effects, frame & qualification problems, sensing, etc.)
- Either a classical plan or a high-level program: specifying the agent tasks/behaviours (nondeterministic & domain actions)

Two aspects:

- Search & Classical Planning
- Foundations & High-level programming

Search & Classical Planning (Nir Lipovetzky)

- Introduction to Automated planning & classical planning as search
- Classical planning as search & other formalisms
- Beyond classical planning: transformations

Foundations & High-level planning/programming (Adrian Pearce)

- (Logical) foundations of (multi) agents & intensionality
- Partial observability & epistemic logic
- Possible world reasoning
- Actions in the situation calculus
- Planning & Golog

High-level programming

High-Level Programming is a promising approach from single-agent systems:

- Primitive actions from the agents world
- Connected by standard programming constructs
- Containing controlled amounts of nondeterminism
- Agent plans a "Legal Execution"
- e.g. GOLOG

Vision: the cooperative execution of a shared high-level program by a team of autonomous agents.

Golog (example operators)

- a - Perform a primitive action
- $\delta_1; \delta_2$ - Perform two programs in sequence
- $\phi?$ - Assert that a condition holds
- $\delta_1 | \delta_2$ - Choose between programs to execute
- $\pi(x,\ \delta(x))$ - Choose suitable bindings for variables
- δ^* - Execute a program zero or more times
- $\delta_1 || \delta_2$ - Execute programs concurrently

Key Point: programs can include nondeterminism

Why High-Level Programming?

- Natural, flexible task specification
- Powerful nondeterminism control
  - order of actions, who does what, ...
- Sophisticated logic of action
  - Concurrent actions, continuous actions, explicit time, ...

Ferrein, Lakemeyer et.al. have successfully controlled a RoboCup team using a Golog variant called "ReadyLog" (Ferrein, Fritz and Lakemeyer 2005).

A Quick Example

Consider a Golog program for getting to university of a morning:

$$ringAlarm; (hitSnooze; ringAlarm)^*; turnOffAlarm;$$
$$\pi(food,\ edible(food)?; eat(food)); (haveShower || brushTeeth);$$
$$(driveToUni \mid trainToUni); (time < 11:00)?$$

There are potentially many ways to execute this program, depending on which actions are possible in the world.

Utilises a theory of action to *plan* a *Legal Execution*:

$$\mathcal{D} \models \exists s, \delta' : Trans^*(\delta, S_0, \delta', s) \wedge Final(\delta', s)$$

Motivating Example: The Cooking Agents

Several robotic chefs inhabit a kitchen, along with various ingredients, appliances and utensils. They must cooperate to produce a meal consisting of several dishes.

$$
\begin{array}{c}
\textbf{proc } \mathit{MakeSalad}(\mathit{bowl}) \\
(\mathit{ChopTypeInto}(\mathit{Lettuce}, \mathit{bowl}) \;||\; \\
\mathit{ChopTypeInto}(\mathit{Carrot}, \mathit{bowl}) \;||\; \\
\mathit{ChopTypeInto}(\mathit{Tomato}, \mathit{bowl})\;)\;; \\
\pi(\mathit{agt}, \mathit{Mix}(\mathit{agt}, \mathit{bowl}, 1)) \\
\textbf{end}
\end{array}
$$

$$
\begin{array}{c}
\textbf{proc } \mathit{ChopTypeInto}(\mathit{type}, \mathit{dest}) \\
\pi((\mathit{agt}, \mathit{obj}), \\
\mathit{IsType}(\mathit{obj}, \mathit{type})?\;; \\
\mathit{Chop}(\mathit{agt}, \mathit{obj})\;; \\
\mathit{PlaceIn}(\mathit{agt}, \mathit{obj}, \mathit{dest})) \\
\textbf{end}
\end{array}
$$

Why Classical Planning?

- Efficient search control (admits fast & tractable solutions)
- Decidable variants frequently possible
- Powerful search control
  - heuristics, landmarks, etc.
- Efficient approach to handling uncertainty
  - Compilation, transformations, etc.

International competitions & toolkits (benchmark problems)

Competitions

- International planning competition (IPC)
- General game playing competition (GDL)
- Berleley Pac-Man

Toolkits

- LAPKT (Lightweight Automated Planning ToolKiT)
- Planning.domains (A collection of tools for working with planning domains)

Project work

Berkeley Pac-Man

- Project 1: get you aquainted with Berkely Pac-Man and deriving heiristics
- Project 2: Pac-man *tournament* where your agents will compete.

Summary