Lecture 7: Outline

- Multi-agent systems
- Agent roles, scope, and goals
- Adaptable systems
- Example: a multi-robotic system

Multi-agent systems

- Computer-based systems composed of multiple interacting intelligent/autonomous agents in some environment
- Agents: (typically) software agents, could be also robots or humans
- Essential characteristic: autonomy. Agents are (at least partially) independent, self-aware, autonomous
- Agents have their own local view (local knowledge, local perception) on the environment and base their decisions on this view

Multi-agent systems (cont.)

- Agents – decentralised and distributed entities, cooperating to achieve their individual goals
- Agents (typically) communicate asynchronously
- The characteristics of such systems:
    - have mobile elements (code, devices, data, services, users);
    - need to be context-aware;
    - are open (i.e., components can appear and disappear)

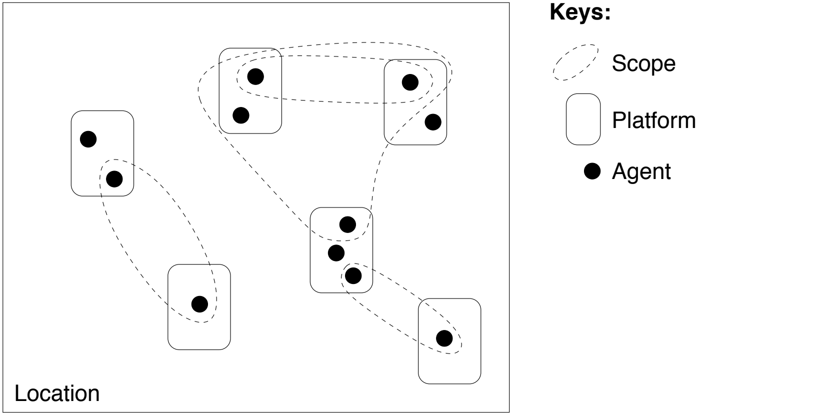
Multi-agent systems: agent roles

- Agent role: structuring unit of the agent functionality
- An agent role determines what it can do, in what joint activities it can participate, etc.
- Each agent has one or more roles associated with it
- May be used as a prerequisite before being involved in various agent cooperation activities, thus ensuring agent compatibility and interoperability
- Examples: *Buyer*, *Seller*, *Manager* (implicit) from the Auction system

Multi-agent systems: agent scope

- Scope – agent cooperation space (typically hosted at some location/server)
- Scope structures activity of agents at a specific location (physical or virtual)
- Scope provides isolation of several communicating agents, thus supporting error confinement and localised error recovery
- Examples: Moodle space for a specific course, “rooms” in a game server, teleconferencing, distant lecturing

Agent abstractions



The same device/component (platform) can host the same agent in different roles

Agent in a particular role collaborates/coordinates with other agents in a scope (if the scope conditions are satisfied)

Multi-agent systems: the scope entry and starting conditions

- Usually, an agent can join the scope activities if it satisfies the pre-defined conditions (e.g., it is in a specific role and the values of some its other attributes are sufficient)
- The scope itself has the conditions indicating when it becomes active. For instance, there should at least one agent in the role "Lecturer" and two agents of the role "Student" to activate the scope "Lecture"
- Since, in general, agents may join and leave a scope at will, the conditions may become broken and the scope activities are stopped/frozen

Multi-agent systems: perceptions and local knowledge

- Agents need to be self-aware of their environment/surroundings, including a general status of other collaborating agents
- Agent perceptions about their environment constitute their stored local knowledge
- Based on this (often incomplete) knowledge, agent make decisions about their next actions. Special rules for "calculating" the best possible action from current agent perceptions
- Crossing into the artificial intelligence (AI) territory here ...

Example: foraging ants as a scientific model

- Humans often blatantly copy/steal best solutions from the nature (naturally, the nature have had plenty time to really test them :))
- One scientific study of self-organising agents: foraging ants
- Ants rely on their perceptions (smells) to approximate distance from the nest, different food sources, and other ants
- Also, ants may leave pheromone traces to "point" other ants to the most promising direction
- We can formulate the rules that determine the next ant actions (which way to go) based on four perceptions and test the likelihood of finding the food or surviving

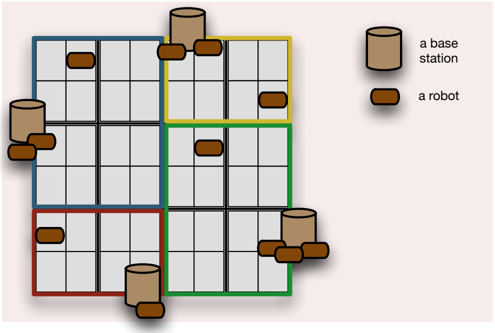
Multi-agent systems: goals

- Sometimes, the purpose of a multi-agent system can be expressed in terms of goals/mission that agent must try to accomplish
- Goals can be typically split into subgoals, forming goal hierarchies
- One of desired properties to be verified of such a multi-agent system is goal reachability

Adaptable systems

- Often, a multi-agent system is required to be flexible with respect to the changing environment $\Rightarrow$ belong the class of adaptable systems
- A system must adapt to both negative (like agent failures or leaving) and positive (e.g., additional available resources) changes in the environment
- As a result, special system fault tolerance and dynamic reconfiguration mechanisms can be installed. Here, a system configuration can be understood as a particular distribution of agents in specific roles and environment resources
- Essentially, it is pre-defined agent cooperation scenarios to collectively recover from a bad situation or take full advantage of a good one

Example: a multi-robotic system



Cleaning an area by robots (workers), which are supervised by base stations. Robots are mobile, while base stations are stationary

Both types of agents are unreliable. In the case of failure, the system should dynamically reconfigure itself, if possible

Multi-robotic system: requirements

1. The main goal of the multi-robotic system is to clean a given area
2. The area to be cleaned is divided into a number of non-intersecting zones, which in turn are divided into a number of non-intersecting sectors. The number of sectors within different zones is the same
3. The area is considered as cleaned when each its zone is cleaned. A zone is considered as cleaned when each its sector is cleaned
4. The system involves two types of agents, base stations and robots, to achieve its main goal
5. Each active (i.e., non-failed) base station is associated with a number of the zones that it has a responsibility of cleaning
6. Each active base station is also associated with a number of robots that it is supervising

Multi-robotic system: requirements (cont.)

7. Each robot may be associated with a single sector that it is assigned to clean at the moment
8. The base stations are able keep track of the cleaning process by storing the cleaning status of area sectors in their memory
9. A base station is able to communicate with other base stations and robots by sending messages
10. Communication between base stations can be used to 1) broadcast changes of the cleaning status of zone sectors, and 2) redistribute attached robots among the base stations
11. Communication from base stations to robots can be used to 1) assign cleaning tasks to the attached robots, and 2) change robot attachment to a different base station

Multi-robotic system: requirements (cont.)

12. Communication from robots to their base station can be used to report successful completion of an assignment
13. A base station may unrecoverably fail at any moment. An active base station has an ability to detect a failure of another base station
14. Once a base station fails, all its zones are considered as non-associated and all its robots as unattached
15. A robot may unrecoverably fail at any moment. A supervising base station has an ability to detect a failure of an attached robot
16. Only a robot, attached to some base station, can be given a cleaning assignment by this base station

Multi-robotic system: requirements (cont.)

17. Only non-cleaned sector can be assigned to a robot. A sector assigned to be cleaned cannot be already assigned to another robot
18. A cleaning assignment, communicated to a robot, should only concern cleaning of a sector belonging to one of the zones associated with the supervising base station
19. A robot cannot be given more than one assignment
20. A base station regularly updates its knowledge about cleaned sectors (after notifications from robots or broadcasts from other base stations)
21. A base station regularly communicates its local knowledge to the other base stations

Multi-robotic system: requirements (cont.)

22. If a base station has cleaned all its zones, its active robots may be reallocated under control of another base station
23. If a base station has no more active robots but still has sectors to clean, its zones and sectors may be reallocated under control of another base station
24. If a base station fails, its unassociated zones and unattached robots may be reallocated under control of another (active) base station

Multi-robotic system: context

CONTEXT

Robots_ctx

SETS

BSTATION

ROBOT

CONSTANTS

zone_number, sector_number, Zones, Sectors, responsible_init, attached_init

AXIOMS

$zone\_number \in \mathbb{N}1 \;\wedge\; sector\_number \in \mathbb{N}1$

$Zones \in \mathbb{P}(\mathbb{N}1) \;\wedge\; Sectors \in \mathbb{P}(\mathbb{N}1)$

$Zones = 1..zone\_number$

$Sectors = 1..sector\_number$

$responsible\_init \in Zones \twoheadrightarrow BSTATION$

$attached\_init \in ROBOT \twoheadrightarrow BSTATION$

END

Multi-robotic system: notes

- Here *zone_number* and *sector_number* are the pre-defined numbers of zones and sectors. Essentially, they are global system parameters (any positive integer numbers)
- The types for all zones (*Zones*) and sectors (*Sectors*) are then constructed as the corresponding intervals of natural numbers. They are defined as constant sets
- The functions *responsible_init* and *attached_init* represent the initial values for which zones are supervised by which base stations and which robots belong to which base stations (i.e., initial system configuration)

Multi-robotic system: machine

MACHINE

Robots_mch

SEES

Robots_ctx

VARIABLES

$responsible, attached, assigned\_zone, assigned\_sector,$
$local\_map, active\_bs, active\_rb$

INVARIANT

$responsible \in Zones \twoheadrightarrow BSTATION$

$attached \in ROBOT \twoheadrightarrow BSTATION$

$assigned\_zone \in ROBOT \twoheadrightarrow Zones$

$dom(assigned\_zone) \subseteq dom(attached)$

$ran(assigned\_zone) \subseteq dom(responsible)$

$\forall r \cdot r \in dom(assigned\_zone) \Rightarrow$
$\quad responsible(assigned\_zone(r)) = attached(r)$

...

Multi-robotic system: machine (cont.)

$$
\begin{array}{l}
assigned\_sector \in ROBOT \nrightarrow Sectors \\
dom(assigned\_zone) = dom(assigned\_sector) \\
active\_bs \in \mathbb{P}(BSTATION) \\
ran(responsible) = active\_bs \\
active\_rb \in \mathbb{P}(ROBOT) \\
dom(attached) \subseteq active\_rb \\
dom(assigned\_zone) \subseteq active\_rb \\
local\_map \in active\_bs \rightarrow \mathbb{P}(Zones \times Sectors)
\end{array}
$$

...

Multi-robotic system: notes (cont.)

- The system keeps the information about the current configuration (*responsible*, *attached*) as well as goal/task assignment (*assigned_zone*, *assigned_sector*)
- Agents are unreliable ⇒ only the currently active agents (*active_bs*, *active_rb*) are taken into account
- *local_map* contains the perception of global situation (how many and which zones/sectors were cleaned) for each base station.
- For a particular base station, its perception about its own zones is more "fresh"/up-to-date than about the other zones

Multi-robotic system: machine (cont.)

```
INITIALISATION
    $responsible, attached := responsible\_init, attached\_init$
    $assigned\_zone, assigned\_sector := \varnothing, \varnothing$
    $local\_map := BSTATION \times \{\varnothing\}$
    $active\_bs, active\_rb := BSTATION, ROBOT$
EVENTS
    $attach\_robot$ = ANY $rb, bs$
        WHERE $rb \in active\_rb \land bs \in active\_bs$
              $rb \notin dom(assigned)$
        THEN
            $attached(rb) := bs$
        END
...
```

Multi-robotic system: machine (cont.)

```
assign = ANY $bs, rb, z, s$
  WHERE $bs \in active\_bs$
        $rb \in dom(attached)$
        $z \in dom(responsible)$
        $s \in Sectors$
        $attached(rb) = bs$
        $responsible(z) = bs$
        $rb \notin dom(assigned\_zone)$
        $(s \mapsto z) \notin local\_map(bs)$
        $\neg(\exists r \cdot r \in dom(assigned\_zone) \wedge$
          $assigned\_zone(r) = z \wedge assigned\_sector(r) = s)$
  THEN
    $assigned\_zone(rb) := z$
    $assigned\_sector(rb) := s$
  END
...
```

Multi-robotic system: notes (cont.)

- A lot of guards reflecting many restrictions for robot assignment, for instance:
  - Both base station and robot should be active;
  - Only a robot attached to a base station can be given assignment by this base station;
  - An assignment sector must be uncleaned;
  - An assigned sector must belong to a zone, for which a base station is responsible;
  - An agent cannot be given more than one assignment;
  - A sector cannot be already assigned to clean.

Multi-robotic system: machine (cont.)

```
$sector\_cleaned \;=\;$ ANY $rb,\; bs$
  WHERE $rb \in dom(attached)$
        $bs \in active\_bs$
        $attached(rb) = bs$
        $rb \in dom(assigned\_zone)$
  THEN
    $local\_map(bs) := local\_map(bs) \cup$
          $\{assigned\_zone(rb) \mapsto assigned\_sector(rb)\}$
    $assigned\_zone \;:=\; \{rb\} \lhd assigned\_zone$
    $assigned\_sector \;:=\; \{rb\} \lhd assigned\_sector$
  END
...
```

Multi-robotic system: machine (cont.)

```
    $robot\_fails \ = \ $ ANY $rb$
      WHERE $rb \in active\_rb$
      THEN
        $active\_rb := active\_rb \ \setminus \{rb\}$
        $attached := \{rb\} \lhd attached$
        $assigned\_zone := \{rb\} \lhd assigned\_zone$
        $assigned\_sector := \{rb\} \lhd assigned\_sector$
      END
...
```

Multi-robotic system: machine (cont.)

```
$bstation\_fails \;=\;$ ANY $bs,\ rbs$
  WHERE $bs \in active\_bs$
        $rbs \subseteq ROBOT$
        $rbs = attached^{\sim}[\{bs\}]$
  THEN
    $active\_bs := active\_bs \setminus \{bs\}$
    $attached := attached \rhd \{bs\}$
    $responsible := responsible \rhd \{bs\}$
    $assigned\_zone := rbs \lhd assigned\_zone$
    $assigned\_sector := rbs \lhd assigned\_sector$
    $local\_map := \{bs\} \lhd local\_map$
  END
...
```

Multi-robotic system: notes (cont.)

- As a result of a base station failure, both zones and robots related to this base station become unassociated/"orphaned". Also, the information about assignments and the local map of the base station should be deleted
- How to calculate all the robots attached to a specific base station? $rbs = attached^{\sim}[\{bs\}]$
- These calculated robots (stored in a local variable *rbs*) are then used to filter out the assignment information

Multi-robotic system: machine (cont.)

```
$reattach\_robots \;=\;$ ANY $rbs,\; bs1,\; bs2$
    WHERE $rbs \subseteq ROBOT$
          $bs1 \in active\_bs$
          $bs2 \in active\_bs$
          $rbs \subseteq attached^{\sim}[\{bs1\}]$
          $rbs \cap dom(assigned\_zone) = \varnothing$
    THEN
        $attached := attached \mathbin{\lhd\!\!\!-} (rbs \times \{bs2\})$
    END
...
```

Multi-robotic system: notes (cont.)

- The event describes a situation when a group of robots are transferred from one base station to another (because all the robots of some base station had failed or all sectors for which one base station is responsible are already cleaned)
- The transferred robots are a subset of all the attached robots: $rbs \subseteq attached^{\sim}[\{bs\}]$
- Only robots that are not involved in any current assignment can be transferred: $rbs \cap dom(assigned\_zone) = \varnothing$

Multi-robotic system: machine (cont.)

```
$update\_map\_from\_others \;=\;$ ANY $bs,\ map,\ zones$
    WHERE $bs \in active\_bs$
          $map \in \mathbb{P}(Zones \times Sectors)$
          $zones \subseteq Zones$
          $zones = responsible^{\sim}[\{bs\}]$
    THEN
      $local\_map(bs) := map \mathbin{⩤} (zones \lhd local\_map(bs))$
    END
...
```

Multi-robotic system: notes (cont.)

- The event describes updating the local map of a base station after receiving broadcasted update from some another base station
- Since the base station view on its own zones is more "fresh", only the information about sectors from other zones must be updated
- Here, the responsible zones are calculated as
  $zones = responsible^{\sim}[\{bs\}]$
  while the "fresher" part of the local map for the base station bs is
  $zones \lhd local\_map(bs)$
- Then
  $local\_map(bs) := map \mathbin{\lhd\!\!\!-} (zones \lhd local\_map(bs))$

  overwrites the received data with a possibly more up-to-date information