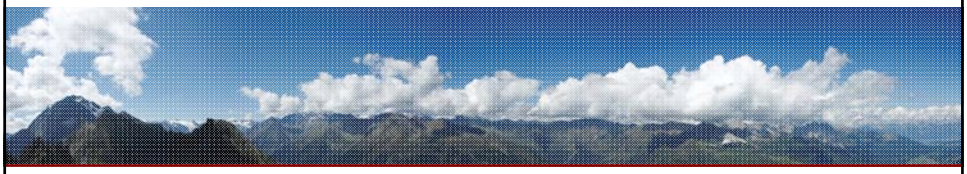
# Intelligent Systems

# Software Agents

Dr Anna Fensel





## Where are we?

STI · INNSBRUCK

| # | Title |
|---|---|
| 1 | Introduction |
| 2 | Propositional Logic |
| 3 | Predicate Logic |
| 4 | Reasoning |
| 5 | Search Methods |
| 6 | CommonKADS |
| 7 | Problem-Solving Methods |
| 8 | Planning |
| **9** | **Software Agents** |
| 10 | Rule Learning |
| 11 | Inductive Logic Programming |
| 12 | Formal Concept Analysis |
| 13 | Neural Networks |
| 14 | Semantic Web and Services |

## Agenda

STI · INNSBRUCK

1. Motivation
2. Technical solution and illustrations
   - Definitions
   - Properties
   - Environments
   - Agents as intentional systems
   - Abstract architecture
   - Multi-agent systems
3. Large example: Robocup
4. Extensions
5. Summary
6. References

3

STI · INNSBRUCK

# MOTIVATION

4

## Motivation

STI · INNSBRUCK

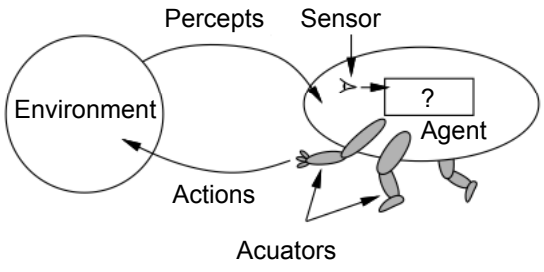
- Do tasks that normally human user needs to do automatically.
- Agents are supposed to carry out (probably complex) tasks autonomously.
- Idea: “intelligent agents” react to changes and act in order to reach specific goals.

5

## Motivation

STI · INNSBRUCK

- The agent perceives the environment via sensors and influences it through actuators.
- Agents can carry out tasks autonomously and react to changes in its environment.




6

## Motivating example

STI · INNSBRUCK

Assuming the tasks a taxi driver has to complete:

- Perception: Camera, speedometer, GPS
- Actions: steer, change gear, break, talk to guest
- Goals: Safe, fast, legal, comfortable ride, maximize profit
- Environment: streets, other actors, guests


7

STI · INNSBRUCK

# TECHNICAL SOLUTION AND ILLUSTRATIONS

8

STI · INNSBRUCK

# Definition



## Definition

STI · INNSBRUCK

- There are many different definitions from various areas, such as software engineering, classical logics, logic programming, robotic:
  - Genesereth/Ketchpel: A program is a software agent if it communicates correctly in an agent language, such as ACL (Agent Communication Language) or KQML (Knowledge Query and Manipulation Language).
  - BDI (Belief, Desire and Intentions) agents are described by their believes, desires, and intentions. This complies with three modalities of a complex modal logic, which can be found in the data structures of the system.

## Definition

STI · INNSBRUCK

- Kowalski follows a traditional approach of logic based agents and uses logic programming for the implementation of agents.
- Shoham's definition is more focused: A hard- or software is an agent if one analyses it with the help of mental terms.
- Wooldridge/Jennings consider hard- or software an agent if it is:
  - autonomous (independently follows its goals)
  - social (is cooperating with a human or other agents)
  - pro-active (takes initiative) und
  - reactive (perceives its environment and reacts to changes).
- An agent is a computer system capable of autonomous action in some environment in order to meet its design objectives.

11

## Definition

STI · INNSBRUCK

- Autonomous entities that perceive their environment and act upon it.
- Autonomy is the ability to control their own behavior and act without human intervention.
- Agents pursue goals in a such a way as to optimize some given performance measure
- They operate flexibly and rationally in a variety of circumstances
- Does NOT include omniscience, omnipotence, or perfection

12

STI · INNSBRUCK

# Properties



## Properties

STI · INNSBRUCK

1. Interaction
2. Reactivity
3. Proactiveness
4. Balancing Reactive and Goal-Oriented Behavior
5. Social Ability
6. Mobility
7. Veracity
8. Benevolence
9. Rationality
10. Learning/adaption

## Interaction

STI · INNSBRUCK

- Agents may be affected by other agents (including humans) in pursuing their goals
- May take place directly via a *communication language*
- May take place indirectly via the environment
  - Agents sense the actions of other agents and react accordingly

15

## Reactivity

STI · INNSBRUCK

- If a program's environment is guaranteed to be fixed, the program need never worry about its own success or failure – program just executes blindly
  - Example of fixed environment: compiler
- The real world is not like that: things change, information is incomplete. Many (most?) interesting environments are *dynamic*
- Software is hard to build for dynamic domains: program must take into account possibility of failure – ask itself whether it is worth executing!
- A *reactive* system is one that maintains an ongoing interaction with its environment, and responds to changes that occur in it (in time for the response to be useful)

16

## Proactiveness

STI · INNSBRUCK

- Reacting to an environment is easy (e.g., stimulus → response rules)
- But we generally want agents to *do things for us*
- Hence *goal directed behavior*
- Pro-activeness = generating and attempting to achieve goals; not driven solely by events; taking the initiative
- Recognizing opportunities

17

## Balancing Reactive and Goal-Oriented Behavior

STI · INNSBRUCK

- We want our agents to be reactive, responding to changing conditions in an appropriate (timely) fashion
- We want our agents to systematically work towards long-term goals
- These two considerations can be at odds with one another
- Designing an agent that can balance the two remains an open research problem

18

## Social Ability

STI · INNSBRUCK

- The real world is a *multi*-agent environment: we cannot go around attempting to achieve goals without taking others into account
- Some goals can only be achieved with the cooperation of others
- Similarly for many computer environments: witness the Internet
- *Social ability* in agents is the ability to interact with other agents (and possibly humans) via some kind of *agent-communication language*, and perhaps cooperate with others

19

## Other Properties

STI · INNSBRUCK

Other properties, sometimes discussed in the context of agents:

- *mobility*: the ability of an agent to move around an electronic network
- *veracity*: an agent will not knowingly communicate false information
- *benevolence*: agents do not have conflicting goals, and that every agent will therefore always try to do what is asked of it
- *rationality*: agent will act in order to achieve its goals, and will not act in such a way as to prevent its goals being achieved — at least insofar as its beliefs permit
- *learning/adaption*: agents improve performance over time

20

STI · INNSBRUCK

# Environments



## Environments – Accessible vs. inaccessible

STI · INNSBRUCK

- An accessible environment is one in which the agent can obtain complete, accurate, up-to-date information about the environment's state
- Most moderately complex environments (including, for example, the everyday physical world and the Internet) are inaccessible
- The more accessible an environment is, the simpler it is to build agents to operate in it

## Environments – Deterministic vs. non-deterministic

STI · INNSBRUCK

- A deterministic environment is one in which any action has a single guaranteed effect — there is no uncertainty about the state that will result from performing an action
- The physical world can to all intents and purposes be regarded as non-deterministic
- Non-deterministic environments present greater problems for the agent designer

23

## Environments - Episodic vs. non-episodic

STI · INNSBRUCK

- In an episodic environment, the performance of an agent is dependent on a number of discrete episodes, with no link between the performance of an agent in different scenarios
- Episodic environments are simpler from the agent developer's perspective because the agent can decide what action to perform based only on the current episode — it need not reason about the interactions between this and future episodes

24

## Environments - Static vs. dynamic

STI · INNSBRUCK

- A static environment is one that can be assumed to remain unchanged except by the performance of actions by the agent
- A dynamic environment is one that has other processes operating on it, and which hence changes in ways beyond the agent's control
- Other processes can interfere with the agent's actions (as in concurrent systems theory)
- The physical world is a highly dynamic environment

25

## Environments – Discrete vs. continuous

STI · INNSBRUCK

- An environment is discrete if there are a fixed, finite number of actions and percepts in it
- Russell and Norvig give a chess game as an example of a discrete environment, and taxi driving as an example of a continuous one
- Continuous environments have a certain level of mismatch with computer systems
- Discrete environments could *in principle* be handled by a kind of "lookup table"

26

STI · INNSBRUCK

# Agents as intentional systems



## Agents as Intentional Systems

STI · INNSBRUCK

- When explaining human activity, it is often useful to make statements such as the following:
  - Janine took her umbrella because she *believed* it was going to rain.
  - Michael worked hard because he *wanted* to possess a PhD.
- These statements make use of a *folk psychology*, by which human behavior is predicted and explained through the attribution of *attitudes*, such as believing and wanting (as in the above examples), hoping, fearing, and so on.
- The attitudes employed in such folk psychological descriptions are called the *intentional* notions.

## Agents as Intentional Systems

STI · INNSBRUCK

- The philosopher Daniel Dennett coined the term *intentional system* to describe entities 'whose behavior can be predicted by the method of attributing belief, desires and rational acumen'
- Dennett identifies different 'grades' of intentional system: 'A *first-order* intentional system has beliefs and desires (etc.) but no beliefs and desires *about* beliefs and desires. …A *second-order* intentional system is more sophisticated; it has beliefs and desires (and no doubt other intentional states) about beliefs and desires (and other intentional states) — both those of others and its own'

29

## Agents as Intentional Systems

STI · INNSBRUCK

- The intentional notions are thus *abstraction tools*, which provide us with a convenient and familiar way of describing, explaining, and predicting the behavior of complex systems
- Remember: most important developments in computing are based on new *abstractions*:
  - procedural abstraction
  - abstract data types
  - objects

  Agents, and agents as intentional systems, represent a further, and increasingly powerful abstraction
- So agent theorists start from the (strong) view of agents as intentional systems: one whose simplest consistent description requires the intentional stance

30

## Agents as Intentional Systems

STI · INNSBRUCK

- This *intentional stance* is an *abstraction tool* — a convenient way of talking about complex systems, which allows us to predict and explain their behavior without having to understand how the mechanism actually works
- Now, much of computer science is concerned with looking for abstraction mechanisms (witness procedural abstraction, ADTs, objects,…)
  - *So why not use the intentional stance as an abstraction tool in computing — to explain, understand, and, crucially,* program *computer systems?*
- This is an important argument in favor of agents

31

STI · INNSBRUCK

# Agent architecture

32

## Abstract Architecture for Agents

STI · INNSBRUCK

- Assume the environment may be in any of a finite set $E$ of discrete, instantaneous states:

$$E = \{e, e', \ldots\}.$$

- Agents are assumed to have a repertoire of possible actions available to them, which transform the state of the environment:

$$Ac = \{\alpha, \alpha', \ldots\}$$

- A *run*, $r$, of an agent in an environment is a sequence of interleaved environment states and actions:

$$r : e_0 \xrightarrow{\alpha_0} e_1 \xrightarrow{\alpha_1} e_2 \xrightarrow{\alpha_2} e_3 \xrightarrow{\alpha_3} \cdots \xrightarrow{\alpha_{u-1}} e_u$$

33

## Abstract Architecture for Agents

STI · INNSBRUCK

- Let:
  - R be the set of all such possible finite sequences (over $E$ and $Ac$)
  - $R^{Ac}$ be the subset of these that end with an action
  - $R^{E}$ be the subset of these that end with an environment state

34

## State Transformer Functions

STI · INNSBRUCK

- A *state transformer* function represents behavior of the environment: $\tau : \mathcal{R}^{Ac} \rightarrow \wp(E)$
- Note that environments are…
  - *history dependent*
  - *non-deterministic*
- If $\tau(r)=\varnothing$, then there are no possible successor states to $r$. In this case, we say that the system has *ended* its run
- Formally, we say an environment $Env$ is a triple $Env = \langle E, e_0, \tau \rangle$ where: $E$ is a set of environment states, $e_0 \in E$ is the initial state, and $\tau$ is a state transformer function

35

## Agents

STI · INNSBRUCK

- Agent is a function which maps runs to actions:

$$Ag : \mathcal{R}^{E} \rightarrow Ac$$

An agent makes a decision about what action to perform based on the history of the system that it has witnessed to date. Let AG be the set of all agents

36

## Systems

STI · INNSBRUCK

- A *system* is a pair containing an agent and an environment
- Any system will have associated with it a set of possible runs; we denote the set of runs of agent $Ag$ in environment $Env$ by R$(Ag, Env)$
- (We assume R$(Ag, Env)$ contains only *terminated* runs)

37

## Systems

STI · INNSBRUCK

- Formally, a sequence

$$(e_0, \alpha_0, e_1, \alpha_1, e_2, \ldots)$$

represents a run of an agent $Ag$ in environment $Env = \langle E, e_0, \tau \rangle$ if:

1. $e_0$ is the initial state of $Env$
2. $\alpha_0 = Ag(e_0)$; and
3. For $u > 0$,

$$e_u \in \tau((e_0, \alpha_0, \ldots, \alpha_{u-1})) \quad \text{where}$$
$$\alpha_u = Ag((e_0, \alpha_0, \ldots, e_u))$$

38

## Purely Reactive Agents

STI · INNSBRUCK

- Some agents decide what to do without reference to their history — they base their decision making entirely on the present, with no reference at all to the past
- We call such agents *purely reactive*:

$$action : E \rightarrow Ac$$

- A thermostat is a purely reactive agent

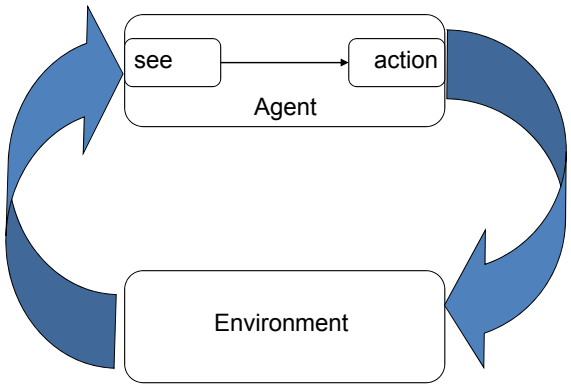
$$action(e) = \begin{cases} \text{off} & \text{if } e = \text{temperature OK} \\ \text{on} & \text{otherwise.} \end{cases}$$

39

## Perception

STI · INNSBRUCK

- Now introduce *perception* system:




40

## Perception

STI · INNSBRUCK

- The *see* function is the agent's ability to observe its environment, whereas the *action* function represents the agent's decision making process
- *Output* of the *see* function is a *percept*:

$$see : E \rightarrow Per$$

which maps environment states to percepts, and *action* is now a function

$$action : Per^* \rightarrow A$$

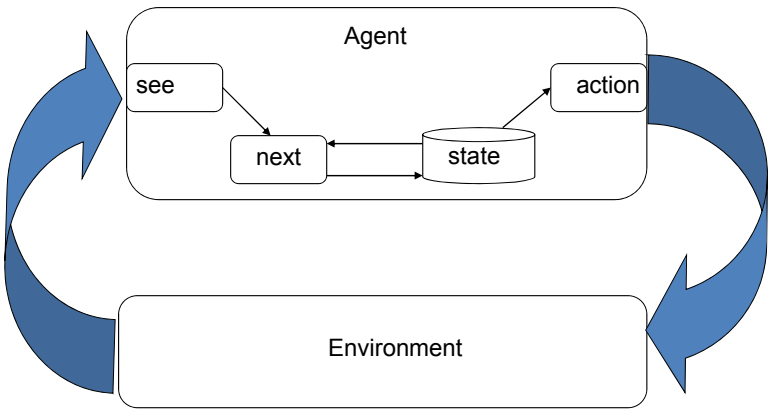
which maps sequences of percepts to actions

41

## Agents with State

STI · INNSBRUCK

- We now consider agents that *maintain state*:




42

## Agents with State

STI · INNSBRUCK

- These agents have some internal data structure, which is typically used to record information about the environment state and history.
  Let $I$ be the set of all internal states of the agent.
- The perception function *see* for a state-based agent is unchanged:

$$see : E \rightarrow Per$$

The action-selection function *action* is now defined as a mapping

$$action : I \rightarrow Ac$$

from internal states to actions. An additional function *next* is introduced, which maps an internal state and percept to an internal state:

$$next : I \times Per \rightarrow I$$

43

## Agent Control Loop

STI · INNSBRUCK

1. Agent starts in some initial internal state $i_0$
2. Observes its environment state $e$, and generates a percept $see(e)$
3. Internal state of the agent is then updated via *next* function, becoming $next(i_0, see(e))$
4. The action selected by the agent is $action(next(i_0, see(e)))$
5. Goto 2

44

## Tasks for Agents

STI · INNSBRUCK

- We build agents in order to carry out *tasks* for us
- The task must be *specified* by us…
- But we want to tell agents what to do *without* telling them how to do it



## Utility Functions over States

STI · INNSBRUCK

- One possibility: associate *utilities* with individual states — the task of the agent is then to bring about states that maximize utility
- A task specification is a function

$$u : E \rightarrow \square$$

which associates a real number with every environment state

## Utility Functions over States

STI · INNSBRUCK

- But what is the value of a *run…*
  - minimum utility of state on run?
  - maximum utility of state on run?
  - sum of utilities of states on run?
  - average?
- Disadvantage: difficult to specify a *long term* view when assigning utilities to individual states
  (One possibility: a *discount* for states later on.)

47

## Utilities over Runs

STI · INNSBRUCK

- Another possibility: assigns a utility not to individual states, but to runs themselves:

$$u : \mathrm{R} \rightarrow \square$$

- Such an approach takes an inherently *long term* view
- Other variations: incorporate probabilities of different states emerging
- Difficulties with utility-based approaches:
  - where do the numbers come from?
  - we don't think in terms of utilities!
  - hard to formulate tasks in these terms

48

## Expected Utility & Optimal Agents

STI · INNSBRUCK

- Write $P(r \mid Ag, Env)$ to denote probability that run $r$ occurs when agent $Ag$ is placed in environment $Env$
  Note: $$\sum_{r \in \mathcal{R}(Ag, Env)} P(r \mid Ag, Env) = 1.$$
- Then optimal agent $Ag_{opt}$ in an environment $Env$ is the one that *maximizes expected utility*:

$$Ag_{opt} = \arg \max_{Ag \in \mathcal{AG}} \sum_{r \in \mathcal{R}(Ag, Env)} u(r) P(r \mid Ag, Env). \quad (1)$$

49

## Bounded Optimal Agents

STI · INNSBRUCK

- Some agents cannot be implemented on some computers
  (A function $Ag : R^E \rightarrow Ac$ may need more than available memory to implement)
- Write $AG_m$ to denote the agents that can be implemented on machine (computer) $m$:

$$\mathcal{AG}_m = \{Ag \mid Ag \in \mathcal{AG} \text{ and } Ag \text{ can be implemented on } m\}.$$

- We can replace equation (1) with the following, which defines the *bounded optimal* agent $Ag_{opt}$:

$$Ag_{opt} = \arg \max_{Ag \in \mathcal{AG}_m} \sum_{r \in \mathcal{R}(Ag, Env)} u(r) P(r \mid Ag, Env). \quad (2)$$

50

## Predicate Task Specifications

STI · INNSBRUCK

- A special case of assigning utilities to histories is to assign 0 (false) or 1 (true) to a run
- If a run is assigned 1, then the agent succeeds on that run, otherwise it fails
- Call these *predicate task specifications*
- Denote predicate task specification by $\Psi$. Thus $\Psi : R \rightarrow \{0, 1\}$.

51

## Task Environments

STI · INNSBRUCK

- A *task environment* is a pair $\langle Env, \Psi \rangle$ where $Env$ is an environment,

$$\Psi : R \rightarrow \{0, 1\}$$

  is a predicate over runs.
  Let TE be the set of all task environments.
- A task environment specifies:
  - the properties of the system the agent will inhabit
  - the criteria by which an agent will be judged to have either failed or succeeded

52

## Task Environments

STI · INNSBRUCK

- Write $R_\Psi(Ag, Env)$ to denote set of all runs of the agent $Ag$ in environment $Env$ that satisfy $\Psi$:

$$\mathcal{R}_\Psi(Ag, Env) = \{r \mid r \in \mathcal{R}(Ag, Env) \text{ and } \Psi(r) = 1\}.$$

- We then say that an agent $Ag$ succeeds in task environment $\langle Env, \Psi \rangle$ if

$$\mathcal{R}_\Psi(Ag, Env) = \mathcal{R}(Ag, Env)$$

53

## Achievement & Maintenance Tasks

STI · INNSBRUCK

- Two most common types of tasks are achievement tasks and maintenance tasks:
  1. Achievement tasks are those of the form "achieve state of affairs $\phi$"
  2. Maintenance tasks are those of the form "maintain state of affairs $\psi$"

54

## Achievement & Maintenance Tasks

STI · INNSBRUCK

- An achievement task is specified by a set $G$ of "good" or "goal" states: $G \subseteq E$
  The agent succeeds if it is guaranteed to bring about at least one of these states (we do not care which one — they are all considered equally good).
- A maintenance goal is specified by a set *B* of "bad" states: $B \subseteq E$
  The agent succeeds in a particular environment if it manages to *avoid* all states in $B$ — if it never performs actions which result in any state in $B$ occurring

55

## Agent Synthesis

STI · INNSBRUCK

- *Agent synthesis* is automatic programming: goal is to have a program that will take a task environment, and from this task environment automatically generate an agent that succeeds in this environment:
  $$syn : \mathcal{TE} \rightarrow (\mathcal{AG} \cup \{\bot\}).$$
  (Think of $\bot$ as being like `null` in Java.)
- Synthesis algorithm is:
  - *sound* if, whenever it returns an agent, then this agent succeeds in the task environment that is passed as input
  - *complete* if it is guaranteed to return an agent whenever there exists an agent that will succeed in the task environment given as input

56

## Agent Synthesis

STI · INNSBRUCK

- Synthesis algorithm $syn$ is sound if it satisfies the following condition:

$$syn(\langle Env, \Psi \rangle) = Ag \text{ implies } \mathcal{R}(Ag, Env) = \mathcal{R}_{\Psi}(Ag, Env).$$

and complete if:

$$\exists Ag \in \mathcal{AG} \text{ s.t. } \mathcal{R}(Ag, Env) = \mathcal{R}_{\Psi}(Ag, Env) \text{ implies } syn(\langle Env, \Psi \rangle) \neq \bot.$$

57

STI · INNSBRUCK

# Multi-agent systems

58

## Multi Agent Systems

STI · INNSBRUCK

- Traditional Multiagent Systems
  - Several agents coordinate their knowledge and activities by reasoning about the problem solving process
- Distributed Problem Solving
  - A particular problem is solved by dividing tasks among a number of generally equivalent nodes who divide and share knowledge about the problem
- Modern *multiagent systems* actually cover both

59

## Characteristics of Multiagent Systems

STI · INNSBRUCK

- Each agent has incomplete information
- Control is decentralized
- Data is decentralized
- Computation is asynchronous

60

## Diversity of Multiagent Systems

STI · INNSBRUCK

| | |
|---|---|
| **Agents** | Number<br>Uniformity<br>Goals<br>Architecture<br>Abilities (sensor & effectors) |
| **Interaction** | Frequency<br>Persistence<br>Level<br>Pattern (flow of control)<br>Variability<br>Purpose |
| **Environment** | Predictability<br>Accessibility<br>Dynamics<br>Diversity<br>Availability of resources |

61

## Common Application Characteristics

STI · INNSBRUCK

- Inherent Distribution
  - Geographically
  - Temporally
  - Semantics – requires different ontologies and languages
  - Functional – requires different cognitive capabilities
- Inherent Complexity
  - Too large to be solved by single, centralized system

62

## Properties of Multiagent Systems

STI · INNSBRUCK

- Speed & Efficiency
- Robustness & Reliability
- Scalability & Flexibility
- Cost
- Distributed Development
- Reusability

63

## Challenging Issues in Multi-Agent Systems

STI · INNSBRUCK

- When and how should agents interact – cooperate and compete – to successfully meet their design objectives?
- Two approaches
  - Bottom up – search for specific agent-level capabilities that result in sufficient group capabilities
  - Top down – search for group-level conventions that appropriately constrain interaction at the agent level
- Leads to several interesting issues …

64

## Challenging Issues in Multi-Agent Systems

STI · INNSBRUCK

1. How to enable agents to decompose their tasks and goals (and allocate sub-goals and sub-tasks to other agents) and synthesize partial results
2. How to enable agents to communicate, what languages and protocols to use
3. How to enable agents to represent and reason about the actions, plans, and knowledge of other agents in order to interact with them

65

## Challenging Issues in Multi-Agent Systems

STI · INNSBRUCK

4. How to enable agents to represent and reason about the state of their interactions
5. How to enable agents to recognize and handle conflicts between agents
6. How to engineer practical multiagent systems

66

## Challenging Issues in Multi-Agent Systems

STI · INNSBRUCK

7. How to effectively balance local computational versus communication
8. How to avoid or mitigate harmful (chaotic or oscillatory) system wide behavior
9. How to enable agents to negotiate and contract with each other
10. How to form and dissolve organizational structures to meet specific goals and objectives

67

## Challenging Issues in Multi-Agent Systems

STI · INNSBRUCK

11. How to formally describe multiagent systems and the interaction between agents and how to ensure multiagent systems are correctly specified
12. How to realize *intelligent processes* such as problem solving, planning, decision making, and learning in a multiagent systems context

68

## Applications of Multiagent Systems

STI · INNSBRUCK

- Electronic commerce
- Real-time monitoring and control of networks
- Modeling and control of transportation systems
- Information handling
- Automatic meeting scheduling

69

## Applications of Multiagent Systems (cont.)

STI · INNSBRUCK

- Industrial manufacturing and production
- Electronic entertainment
- Re-engineering of information flow in large organizations
- Investigation of complex social phenomena such as evolution of roles, norms, and organizational structures

70

STI · INNSBRUCK

# LARGE EXAMPLE



## Overview of RoboCup

STI · INNSBRUCK

- The Robot World Cup Initiative
- "By the year 2050, develop a team of fully autonomous humanoid robots that can win against the human world soccer champion team." www.robocup.org
- A standard problem for AI research
- Started in 1992 as the Robot J-League (Japan)
- First games and conferences in 1997
- Workshops, conferences and yearly competition
- Check out this YouTube video: http://www.youtube.com/watch?v=VaXtnqjk4Bc (Graz 2009)

Slides adapted from Kevin Lam

## RoboCup Leagues

STI · INNSBRUCK

| | |
|---|---|
|  | humanoid |
|  | small size |
|  | middle size |
|  | simulation |

73

## Robocup Leagues: humanoid

STI · INNSBRUCK

- In the humanoid league, autonomous robots with a human-like body plan and human-like senses play soccer against each other.
- In addition to soccer competitions technical challenges take place.
- Research issues addressed in the humanoid league include:
  - Walking
  - Running
  - Kicking the ball while maintaining balance
  - Visual perception of the ball, other players, and the field
  - Self-localization,
  - Team play
- Several of the best autonomous humanoid robots in the world compete in the RoboCup Humanoid League.
- http://www.tzi.de/humanoid/bin/view/Website/WebHome

74

## Robocup Leagues: small size

STI · INNSBRUCK

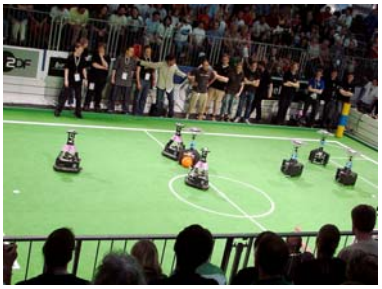
- A Small Size robot soccer game takes place between two teams of five robots each.
- Each robot must conform to the dimensions as specified in the F180 rules:
  - The robot must fit within an 180mm diameter circle.
  - It must be no higher than 15cm unless they use on-board vision.
- Robots come in two flavours, those with local on-board vision sensors and those with global vision.
  - Global vision robots use an overhead camera and off-field PC to identify and track the robots as they move around the field.
  - Local vision robots have their sensing on the robot itself.
- The vision information is either processed on-board the robot or is transmitted back to the off-field PC for processing.
- Communications is wireless and typically uses dedicated commercial FM transmitter/receiver units.

75

## Robocup Leagues: middle size

STI · INNSBRUCK

- The middle size league involves robots that must fit inside a 50 cm x 50 cm x 80 cm box.
- The maximum weight of the robot is 40 kg.
- Two teams of 5 mid-sized robots with all sensors on-board play soccer on a field.
- Communications is wireless and typically uses dedicated commercial FM transmitter/receiver units.

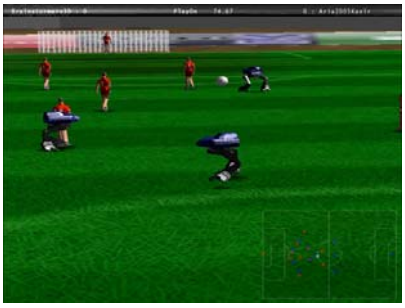


76

## Robocup Leagues: simulation

STI · INNSBRUCK

- The simulation league does not involve real robots but simulates their behavior on a screen.
- It consists of a number of competitions with computer simulated soccer matches as the main event.
- There are no actual robots in this league but spectators can watch the action on a large screen, which looks like a giant computer game.
- Each simulated robot player may have its own play strategy and characteristic and every simulated team actually consists of a collection of programs.
- Many computers are networked together in order for this competition to take place.
- The games last for about 10 minutes, with each half being 5 minutes duration.

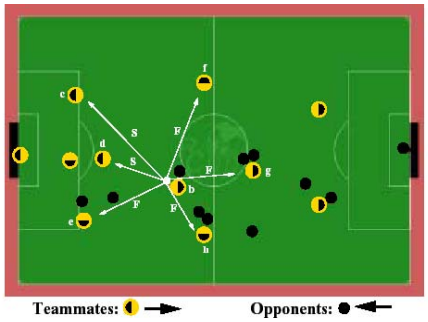


77

## Challenges involved in Robocup Simulation League

STI · INNSBRUCK

- Dynamic behavior of agents (players) must be enabled.
- There are limited means of communication because of limited shared bandwidth which might lead to delays.
- Players must be able to perform real-time actions.
- There is a tremendous state space.
- The multi-agent system must allow cooperation and competition.

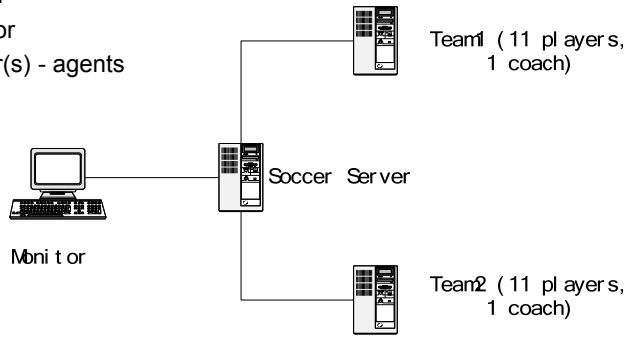



78

## Robocup System – Overview for simulation league

STI · INNSBRUCK

- Provides a platform to develop software techniques, without the necessity of creating physical robots.
- Consists of three main applications:
  - Soccer Server
  - Soccer Monitor
  - Soccer Player(s) - agents




79

## Robocup System – Soccer Server

STI · INNSBRUCK

- A system allowing several autonomous program agents to play a virtual soccer game.
- The games are carried out in a client/server style – where each client = one player.
- The communication can be done over a network connection, or even the Internet.

80

## Robocup System – Monitor

STI · INNSBRUCK

- Used to display the visual progress of the game.
- Several Monitors can be connected to the server.
- It can also be used to interrupt game play by doing simple tasks such as dropping a ball.

81

## Robocup System – Soccer Team - players

STI · INNSBRUCK

- The agents are the brains of the players.
- Players receive sensory information from the server, upon which decisions are made.
- Commands are formatted and sent to the server.

82

## Robocup System – Soccer Team - players

STI · INNSBRUCK

**Sensory Input**

time
play mode
my body
landmarks
other players
speech
coach
instructions

**decision-making strategy**

Player Agent

**Outputs**

dash
turn
turn head
kick
catch
speak

83

## Robocup System – Team – coaches

STI · INNSBRUCK

- Privileged clients used to provide assistance.
- Receives noise-free view of the whole field.
- Can only send occasional messages to players (info, advice, freeform, etc.).
- Used for
  - opponent modelling,
  - game analysis,
  - giving strategic tips to teammates.

84

## Some RoboCup Teams

STI · INNSBRUCK

- UvA Trilearn (Amsterdam) (2003 champion)
- CMUnited (Carnegie Mellon)
- Everest (China)
- FC Portugal 2003 (Portugal)
- HELIOS (Japan)
- Magma Furtwangen (Germany)

85

STI · INNSBRUCK

# EXTENSIONS

86

## Agents and Objects

STI · INNSBRUCK

- Are agents just objects by another name?
- Object:
  - encapsulates some state
  - communicates via message passing
  - has methods, corresponding to operations that may be performed on this state

87

## Agents and Objects

STI · INNSBRUCK

- Main differences:
  - *agents are autonomous:*
    agents embody stronger notion of autonomy than objects, and in particular, they decide for themselves whether or not to perform an action on request from another agent
  - *agents are smart:*
    capable of flexible (reactive, pro-active, social) behavior, and the standard object model has nothing to say about such types of behavior
  - *agents are active:*
    a multi-agent system is inherently multi-threaded, in that each agent is assumed to have at least one thread of active control

88

## Agents and Expert Systems

STI · INNSBRUCK

- Aren't agents just expert systems by another name?
- Expert systems typically disembodied 'expertise' about some (abstract) domain of discourse (e.g., blood diseases)
- Example: MYCIN knows about blood diseases in humans
  - It has a wealth of knowledge about blood diseases, in the form of rules
  - A doctor can obtain expert advice about blood diseases by giving MYCIN facts, answering questions, and posing queries

89

## Agents and Expert Systems

STI · INNSBRUCK

- Main differences:
  - agents *situated in an environment:*
    MYCIN is not aware of the world — only information obtained is by asking the user questions
  - agents *act:*
    MYCIN does not operate on patients
- Some *real-time* (typically process control) expert systems *are* agents

90

STI · INNSBRUCK

- When building an agent, we simply want a system that can choose the right action to perform, typically in a limited domain
- We *do not* have to solve *all* the problems of AI to build a useful agent:

  *a little intelligence goes a long way!*
- Oren Etzioni, speaking about the commercial experience of NETBOT, Inc:
  "We made our agents dumber and dumber and dumber…until finally they made money."



STI · INNSBRUCK

## SUMMARY

## Summary

STI · INNSBRUCK

- Agent preceives the environment and acts.
- Agent = architecture + program
- Ideal agent takes action that maximizes performance at a given perception.
- Actions of an autonomous agent depend on experience.
- Mapping of perceptiuon to actions.
- Reactive agents act on perceptions, goal-oriented agents act to reach a goal, utility-based agents maximize their profit.
- Representation of knowledge!
- Various environments. Most difficult: not accessible, episodic, dynamic, and continuous.

93

## Summary

STI · INNSBRUCK

- Multiagent systems are systems in which multiple interacting agents interact to solve problems
- Key concepts of multiagent systems are agents and agent coordination
- There are many important issues for multiagent systems that attempt to answer when and how to interact with whom

94

## Summary

STI · INNSBRUCK

- Common characteristics of multiagent systems are their inherent distribution and complexity
- Distributed and flexible nature of multiagent systems leads to increased speed, robustness, scalability and reusability

95

STI · INNSBRUCK

# REFERENCES

96

## References

STI · INNSBRUCK

- Mandatory Reading:
  - G. Görz et al., Handbuch der künstlichen Intelligenz, Oldenbourg, 2003.
    Kapitel 24: Software Agenten.
  - Bradshaw, J.; „An Introduction to Software Agents“; AAAI Press/The MIT Press
  - Finin, Tim and Fritzson, Richard and McKay, Don and McEntire, Robin: KQML as an agent communication language. CIKM '94: Proceedings of the third international conference on Information and knowledge management. ACM. 1994.

97

## References

STI · INNSBRUCK

- Further Reading:
  - Wooldridge, M. & Jennings, M; „Intelligent Agents: Theory and Practice“; The Knowledge Engineering Review 10
  - Nwana, H.; „Software Agents: An Overview“; The Knowledge Engineering Review, Vol. 11 No 3
  - Rao A. & Geogreff M.; „BDI Agents: From Theory to Practice“; Tech. Rep. 56, Australian Artificial Intelligence Institue, Melbourne, Australia, Apr 1995
  - http://www.robocup.org/

98

## References

STI · INNSBRUCK

- Wikipedia links:
  - http://en.wikipedia.org/wiki/Agent
  - http://en.wikipedia.org/wiki/Software_agent
  - http://en.wikipedia.org/wiki/Intelligent_agent
  - http://en.wikipedia.org/wiki/Mycin
  - http://de.wikipedia.org/wiki/Knowledge_Query_and_Manipulation_Language
  - http://en.wikipedia.org/wiki/RoboCup



## Next Lecture

STI · INNSBRUCK

| # | Title |
|---|---|
| 1 | Introduction |
| 2 | Propositional Logic |
| 3 | Predicate Logic |
| 4 | Reasoning |
| 5 | Search Methods |
| 6 | CommonKADS |
| 7 | Problem-Solving Methods |
| 8 | Planning |
| 9 | Software Agents |
| **10** | **Rule Learning** |
| 11 | Inductive Logic Programming |
| 12 | Formal Concept Analysis |
| 13 | Neural Networks |
| 14 | Semantic Web and Services |

## Questions?

STI · INNSBRUCK