LUND UNIVERSITY

AI@CS

Department of Computing Science

Learning for Agent-Based Systems

Sławomir Nowaczyk

Computer Science Lab
Department of Automatics
AGH University of Science and Technology
Kraków, Poland

April 27, 2009

LUND UNIVERSITY

AI@CS

Department of Computing Science

Agent-Based Systems

- Agent: *autonomous*
- Environment:
  - *fully, partially, not observable*
  - *deterministic, stochastic, strategic actions*
  - *static, dynamic, stationary, non-stationary*
  - *episodic, sequential, discrete, continuous*
- An autonomous agent is a system
  - situated within and a part of an environment
  - that senses that environment and acts on it,
  - over time,
  - in pursuit of its own agenda
  - and so as to affect what it senses in the future.

LUND UNIVERSITY

AI@CS

Department of Computing Science

Why Agents?

- Information integration & knowledge sharing
- Coordination & cooperative problem-solving
- Autonomous mobile robots
- Believable agents & artificial life
- Reactive
  - systems that respond in a timely fashion to various changes in the environment
- Goal-oriented, pro-active & purposeful
- Socially communicative
  - able to communicate with other agents
  - including people

LUND UNIVERSITY

AI@CS

Department of Computing Science

Types of Agents

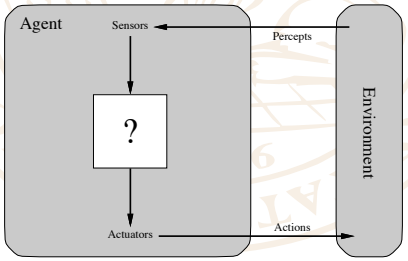
- For each possible percept sequence, an
  - ideal rational agent should choose the action
  - that is expected to
  - maximise its performance measure,
  - on the basis of the evidence provided by the percept sequence and
  - whatever built-in knowledge the agent has.
- Agent needs a *performance measure*
  - domain- and task-specific
  - often non-trivial to design and/or evaluate
- Omniscient vs rational agents
- Limits on available perceptual history

Agent Implementation

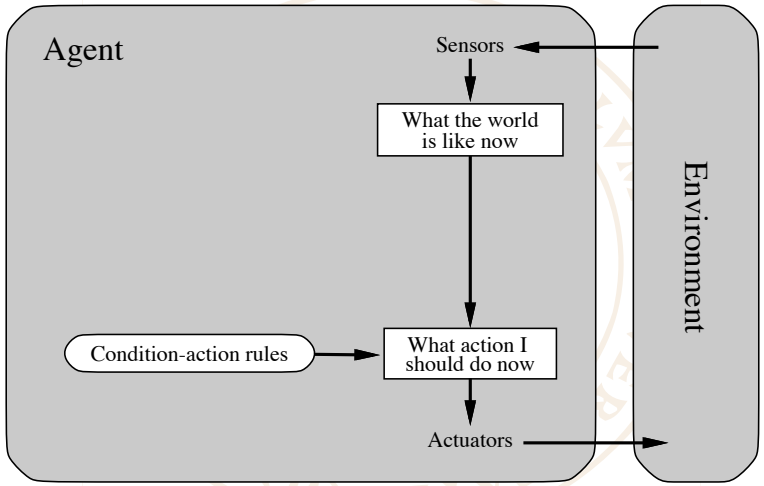
- Architecture
  - computational structures for encoding, representing and manipulating knowledge
  - and producing actions in pursuit of goals
  - like a specialised programming language
  - often specific theory of intelligent behaviour
- Agent program
  - content that is being processed by architectural computational structures
  - corresponding to particular problem domains
  - reflecting particular selection of algorithms
- Agent input data

Behaviour of an Agent

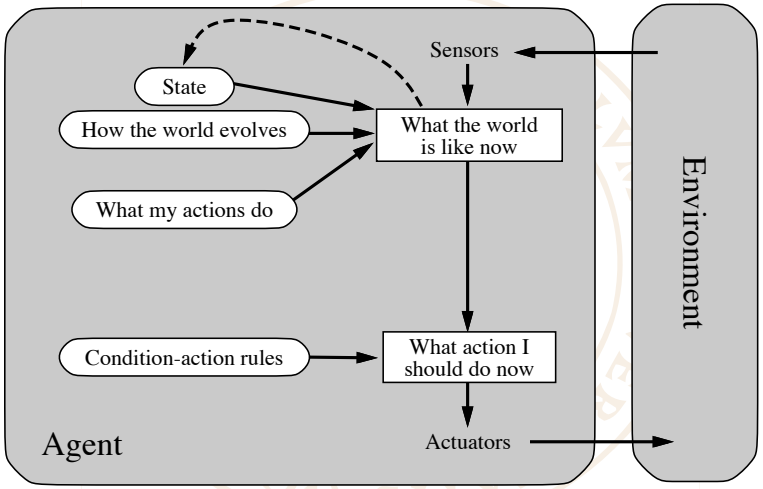
- `while True:`
  - `Observe_Environment()`
  - `Update_Memory()`
  - `Choose_Best_Action()`
  - `Update_Memory()`
  - `Execute_Action()`



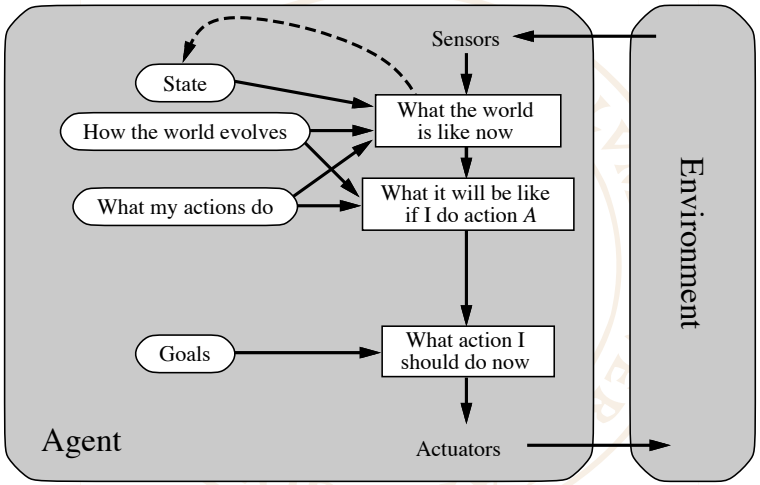
Reflex Agent



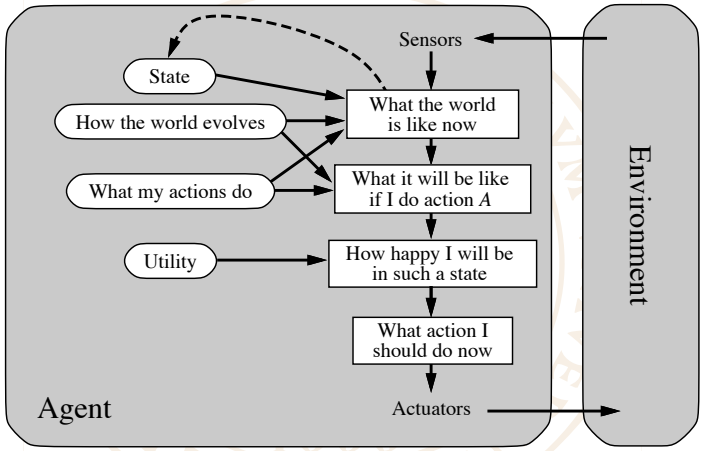
Stateful Agent



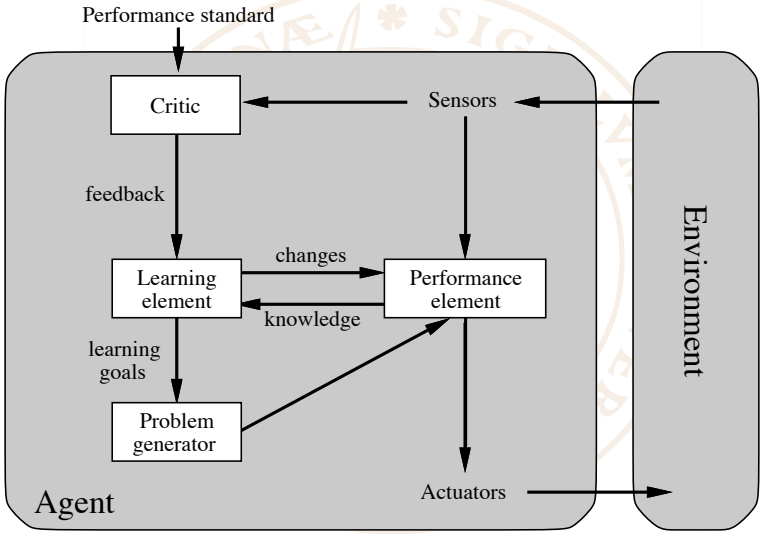
Goal Based Agent



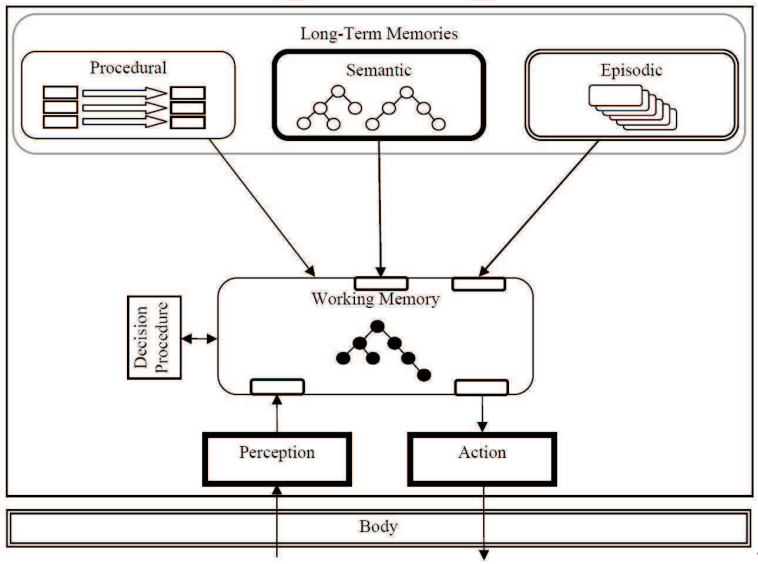
Utility Based Agent



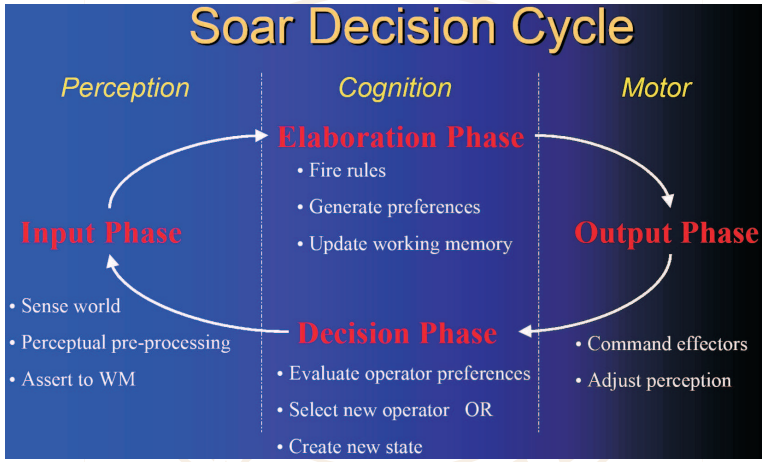
Learning Agent



SOAR Architecture



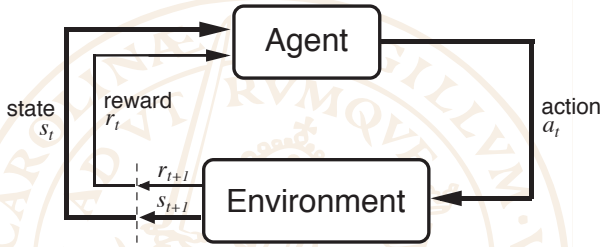
SOAR Architecture



Reinforcement Learning

- Learning from interactions with environment
  - no teacher to know the "right answers"
- Trial-and-error search
  - perform an action
  - evaluate response of the environment
- Delayed rewards
  - some actions yield immediate rewards
  - others simply lead to "better" states
- Some similarities to *baby playing*
  - cause-effect relationship
- Markov property

Reinforcement Learning



Reinforcement Learning

- Learning a mapping from situations to actions
  - in order to maximise scalar reward value
- Actions are selected based on past experiences
- Exploitation
  - try previously well-rewarded actions
  - expecting similar results
- Exploration
  - try new sequences of actions
  - they may turn out to be even better
- Proper balancing is difficult
  - especially in stochastic or non-stationary environments

LUND UNIVERSITY

AI@CS

Department of Computing Science

Policy

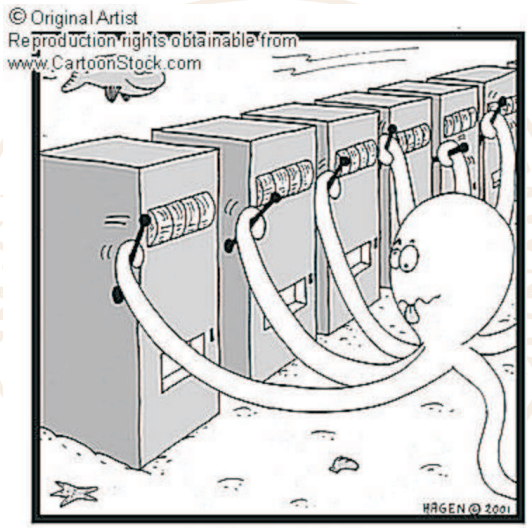
- In situation s_t agent chooses action a
  - world changes to s_{t+1}
  - agent perceives s_{t+1} and receives r_{t+1}
- Policy $\pi(s, a) = Pr\{a_t = a | s_t = s\}$
  - probability that agent will choose a
  - given that current state is s
- *n-armed bandit* problem
  - n actions to choose from
  - each one yields stochastic reward
  - exact distribution is unknown
  - maximise long-term profit

LUND UNIVERSITY

AI@CS

Department of Computing Science

n-armed Bandit



Compulsive gambling

LUND UNIVERSITY

AI@CS

Department of Computing Science

ϵ-greedy Policy

- Agent can estimate payoff of each arm
  - based on past action executions
- Such estimate is called Q *value*
- Obvious solution: greedy policy
  - always choose action with highest Q value
- But this completely ignores exploration
- ϵ-greedy policy
  - choose random action every now and then
- $\pi(s, a^* | a^* = \arg\max Q(a)) = 1 - \epsilon + \frac{\epsilon}{|A|}$
- $\pi(s, a | a \neq \arg\max Q(a)) = \frac{\epsilon}{|A|}$

LUND UNIVERSITY

AI@CS

Department of Computing Science

Value Function

- *n-armed bandit* environment is episodic
  - actions of the agent do not change world state
  - we only care about immediate reward
- In most interesting environments, however, some *states* are better than others
  - agent should think in a longer perspective
- Reward function
  - immediate payoff for executing action a
- Value function
  - expected *future* reward from a given state
  - long-term perspective

General Reinforcement Learning Algorithm

- Initialise agent's internal state
  - Q values, V values, policy π, etc.
- `while not Good_Enough():`
  - choose action a using policy π
  - execute action a
  - observe immediate reward r
  - observe new world state s'
  - update internal state based on s, a, r, s'
- Output resulting policy π

Problem Specification

- Decision on what constitutes an internal state
  - representation of agent's knowledge
- Decision on what constitutes a world state
  - as complete as possible
- Means of sensing a world state
- Action-choice mechanism
  - policy
- an evaluation function
  - of current world and internal state
- A means of executing the action
- A way of updating the internal state

The Environment

- Definition of the environment must consist of
  - *state transition function*
  - probability that executing action a in state s will transform world into state s'
  - *reward function*
  - how much reward agent gets for carrying out particular actions or ending in particular states
- This is often called *model* of the environment
- If acting in real world, transition function is given
  - in simulator, it must be programmed
- Reward function is always specified explicitly
  - always make sure you measure the right thing

Tic-Tac-Toe

- Play against *imperfect* opponent
- Reward is 1 for win, -1 for loss or draw
  - 0 for every other move
- $V(s)$ is estimate probability of winning
  - from state s
- $V(XXX) = 1$
- $V(OOO) = 0$
- $V(*) = 0.5$

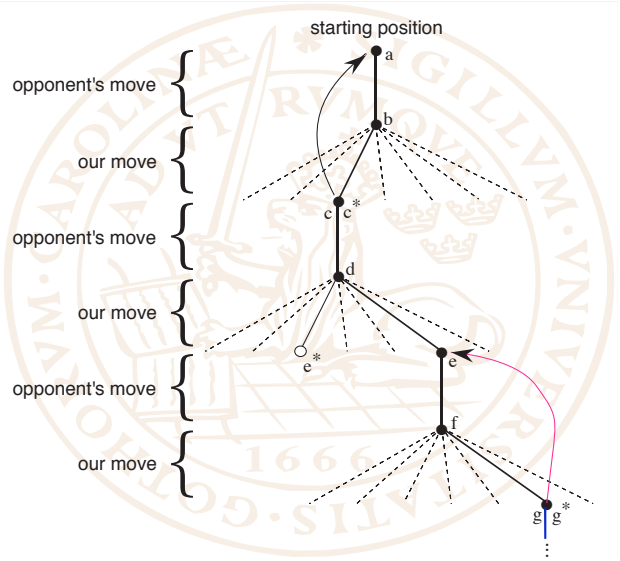
X	O	O
O	X	X
		X

LUND UNIVERSITY

AI@CS

Department of Computing Science

Tic-Tac-Toe



LUND UNIVERSITY

AI@CS

Department of Computing Science

Value adjustment

- Play many games
  - choose move leading to the highest $V(s)$
  - sometimes explore other possibilities
- Adjust estimates of V
  - make them more accurate estimates of winning probability
  - after each *non-exploratory* move, "back-up" the value of new state to the old one
  - $V(s_k) = V(s_k) + \alpha[V(s_{k+1}) - V(s_k)]$
- Under reasonable assumptions, V converges to real probabilities of winning
  - optimal policy

LUND UNIVERSITY

AI@CS

Department of Computing Science

Reward

- Maximise total reward received
  - given immediate rewards r_t
  - $R_t = r_{t+1} + r_{t+2} + r_{t+3} + \cdots + r_T$
- $R_t = \infty$ whenever $T = \infty$
  - discounted reward

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots$$

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

  - $\gamma \simeq 0$ for myopic agents
  - $\gamma \simeq 1$ for far-sighted agents
  - $0 \leq \gamma < 1$

LUND UNIVERSITY

AI@CS

Department of Computing Science

Stochastic Environment

- State transition probability

$$P^a_{ss'} = Pr\{s_{t+1} = s' | s_t = a, a_t = a\}$$

- Expected reward

$$R^a_{ss'} = E\{r_{t+1} | s_t = a, a_t = a, s_{t+1} = s'\}$$

- P and R form a complete environment model
- Expected action reward

$$\rho(s, a) = \sum_{s'} P^a_{ss'} R^a_{ss'}$$

Action Selection

- Policy π maps situations to actions
- Value of state s under policy π is

$$V^\pi(s) = E_\pi\{R_t|s_t = s\} = E_\pi\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}|s_t = s\}$$

- Value of taking action a in s under policy π is

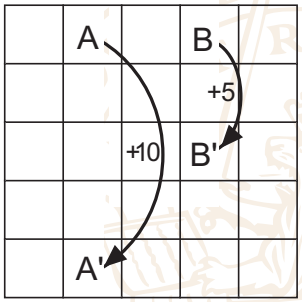
$$Q^\pi(s,a) = E_\pi\{R_t|s_t = s, a_t = a\} = E_\pi\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}|s_t = s, a_t = a\}$$

Bellman Equation

$$\begin{aligned}
V^\pi(s) &= E_\pi\{R_t|s_t = s\} = \\
&= E_\pi\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}|s_t = s\} = \\
&= E_\pi\{r_{t+1} + \gamma\sum_{k=0}^{\infty}\gamma^k r_{t+k+2}|s_t = s\} \\
&= \sum_a \pi(s,a)\sum_{s'} P^a_{ss'}[R^a_{ss'} + \gamma E_\pi\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+2}|s_{t+1} = s'\}] \\
&= \sum_a \pi(s,a)\sum_{s'} P^a_{ss'}[R^a_{ss'} + \gamma V^\pi(s')]
\end{aligned}$$

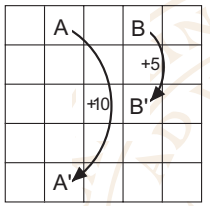
$$Q^\pi(s,a) = \sum_{s'} P^a_{ss'}[R^a_{ss'} + \gamma\sum_{a'}\pi(s',a')Q(s',a')]$$

Grid World Example

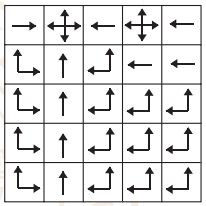


22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7

Grid World Example



22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7



LUND UNIVERSITY

AI@CS

Department of Computing Science

Multi-Agent Learning

- More than one agent in the environment
  - non-stationary
  - stochastic model is not enough
- Environment *adapts* to agent's behaviour
  - cooperation or competition
  - explicit or implicit communication
- Common goal is an emergent property
  - agents may not be aware of it
- Assumption: all agents act rationally
  - often violated
  - modelling other agents is difficult

LUND UNIVERSITY

AI@CS

Department of Computing Science

Game Theory

- Strategic game G consists of:
- A finite set N (players)
- For each player $i \in N$
  - a non-empty set A_i (actions)
  - $A = \prod_i A_i$ (actions)
  - a function $u_i : A \rightarrow \mathbb{R}$ (payoff)
- Playing the game
  - each player chooses an action a_i
  - action profile $a^* = (a_1, a_2, ..., a_n)$
  - all players make their moves simultaneously
  - each player gets the payoff $u_i(a^*)$

LUND UNIVERSITY

AI@CS

Department of Computing Science

Prisoner's Dilemma

- 2 players, 2 actions
  - both confess: 3 years in prison
  - neither confess: 1 year in prison
  - only you confess: 0 years in prison
  - only other confesses: 5 years in prison

u_A/u_B	B confesses	B does not
A confesses	2/2	5/0
A does not	0/5	4/4

LUND UNIVERSITY

AI@CS

Department of Computing Science

Dominant Strategy

- Strategy is *strictly dominant* iff it will result in a greater payoff than any other strategy
  - independent of actions by other players
- Prisoner's Dilemma — confess
  - if the other confesses, it is better to confess
  - if the other does not confess, it is *also* better to confess
- Rational agent should always confess
- *Iterated* prisoner's dilemma is a different issue
  - play prisoner's dilemma n times
  - play prisoner's dilemma ∞ times
  - play prisoner's dilemma random times

LUND UNIVERSITY

AI@CS

Department of Computing Science

Nash Equilibrium

- Nash equilibrium is a set of strategies
  - one for each player
- Such that no single player can improve their payoff by deviating from assigned strategy
  - next best thing after dominant strategy
- *Battle of the Sexes*

u_A/u_B	Bob: theatre	Bob: football
Ann: theatre	2/1	0/0
Ann: football	0/0	1/2

LUND UNIVERSITY

AI@CS

Department of Computing Science

Mixed Strategy

- Probability distribution of player i over actions A_i
- Mixed Nash equilibrium
  - $Pr(T) = 0.5$ $Pr(B) = 0.5$
  - $Pr(L) = 0.5$ $Pr(R) = 0.5$
- Every finite strategic game has got at least one mixed Nash equilibrium (Nash, 1950)

	L	R
T	1/2	2/1
B	2/1	1/2

LUND UNIVERSITY

AI@CS

Department of Computing Science

Intelligent Situated Agent

- Acting within dynamic, potentially adverse, partially unknown, complex environment
  - continuously observing the world
  - actively pursuing its goals
  - aware of own limitations
- No "*stop the world, I need to think*" mentality
- Reactive and deliberative components
- Domain-independent, conceptually similar to General Game Playing Competition
  - accept the rules of a new, unknown game
  - play if effectively right away
  - improve with experience

LUND UNIVERSITY

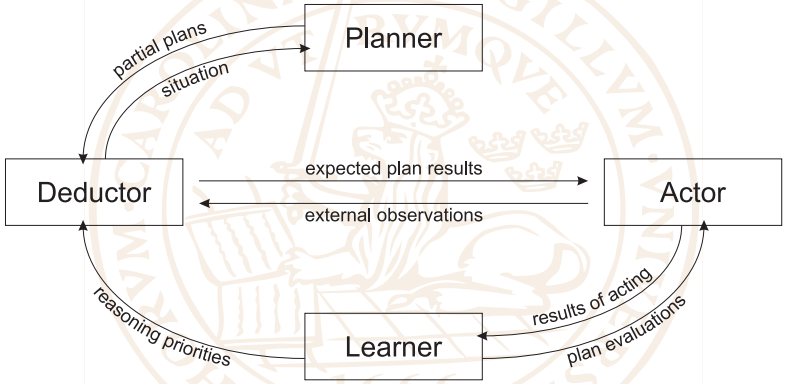
AI@CS

Department of Computing Science

Intelligent Resource-Bounded Agents

- Making rational decisions about solutions that may turn out to be *good enough*
  - a chance to generate a better plan
  - early commitment to some course of action
- Unfortunately, this is known to be *impossible*
  - in general, reasoning progress is unpredictable
- But an agent should be *intelligent*
  - so fallibility is perfectly fine
- As long as we are *improving with experience*
  - failing at a new task is acceptable
  - failing again, in a *novel* way, is equally fine

Agent Architecture



Conditional Partial Plans

- Knowledge representation for exchanging information among modules of the agent
  - integration of several subfields of AI
- Focus on agent's knowledge about the world
  - in any situation, either real or hypothetical
  - often incomplete, uncertain, or contradictory
- Staying responsive in a dynamic world
  - predicting new state of the world
  - often not determined uniquely
  - optimistic versus pessimistic attitude
- Deciding if the plan is worth executing
  - and whether execution proceeds as expected

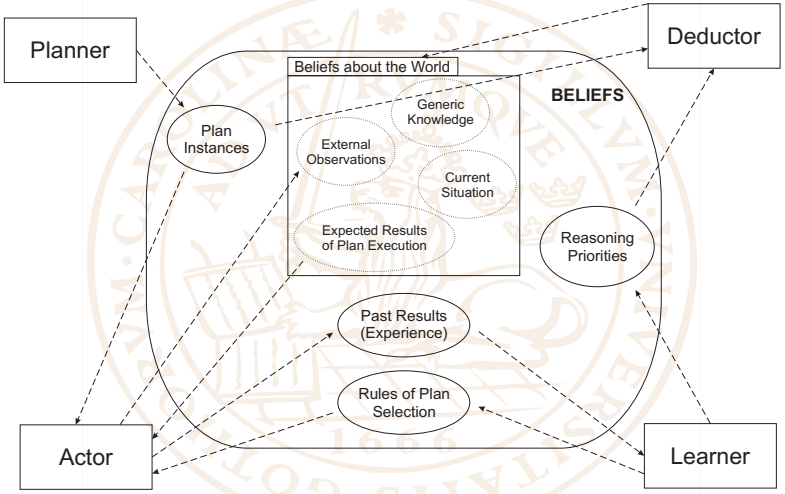
Conditional Partial Plans, cont.

- Plans need to be *partial*
  - agent's computational resources are limited
  - the world around it is continuously changing
  - finding and verifying a complete plan is infeasible in the domains we find interesting
  - world model is not sufficiently accurate
- Plans need to be *conditional*
  - agent's environment is only partially known
  - at the very least it needs to verify that obtained observations match its expectation
  - simple sequences of actions are not flexible enough to achieve the intelligence we aim at

Plan Evaluation

- A complete plan can be evaluated easily
  - for partial ones, figuring out if it is a step in the right direction is very difficult
  - not all domains are *safely explorable*
- Solution: create multiple plans
  - deductively expand agent's knowledge about expected results of each of those plans
  - inductively analyse similarities to past scenarios and how successful were they
- Combine several levels of abstraction
- Take advantage of experience
  - the problem is unsolvable without it

Agent Architecture



Planner

- Based on agent's understanding of the current state of external world
  - as determined from past observations
  - potentially incomplete or incorrect
- Develop a number of plans
  - applicable in agent's current situation
- Very efficient reasoning mechanism
  - employs several simplifying assumptions
  - uses streamlined model of the world
- Impossible to determine with certainty whether a given plan leads in the right direction
  - done by other modules of the architecture

Actor

- Guided by both Deductor and Learner
- Oversee deliberation progress
  - decide when to start acting
  - prioritise reasoning
- Select which plan should be executed now
  - compare available alternatives
- Supervise interactions with the external world
  - convert sensor input into appropriate symbolic representations
  - react to events requiring immediate attention
    - maintaining good self-localisation
    - collision avoidance

Deductor

- Flexible and efficient reasoning framework
  - reasoning is an evolving process
  - ongoing deliberation in a changing world
- Non-omniscient agents
  - not all consequences are readily available
- Inherently epistemic: *agent's point of view*
  - focuses on knowledge of the agent
  - models reasoning process in a realistic way
- Interactions within the world are complex
  - non-monotonic and modal operators
  - reasoning is computationally expensive

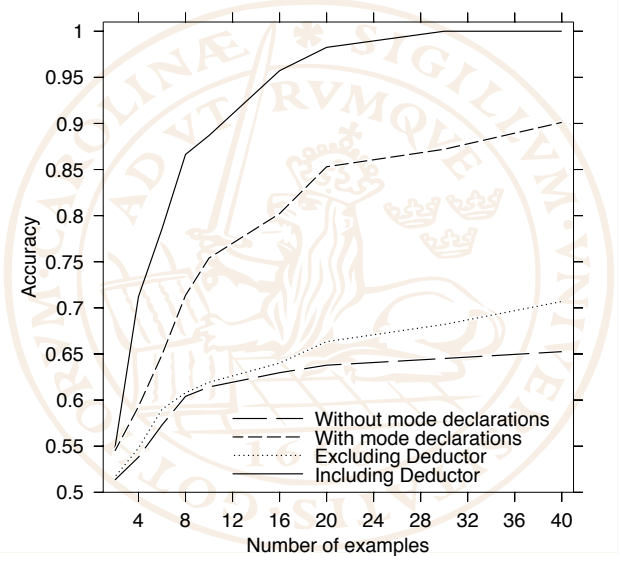
Learner

- Capitalise on previous experience
- Improve criteria for plan selection
  - induce rules for evaluating available plans
  - based results of deduction
  - find the best plan to execute
- Determine which plans
  - are the most promising ones
  - should be developed further
  - can be safely ignored immediately
- Discover interesting reflex reactions
  - reactions to be performed instinctively

Learner, cont.

- We use Inductive Logic Programming
  - allows for rich knowledge representation
  - can use full knowledge of the agent as input
  - resulting hypothesis "makes sense"
  - takes advantage of domain knowledge
  - state of the art algorithms are efficient
- Classification versus evaluation paradigm
  - pairwise "betterThan" comparisons
- Generation of training examples
  - problems with noise in the data
- Inconsistencies in case of dynamic domains

Pure Learning Evaluation



Conservative Agent

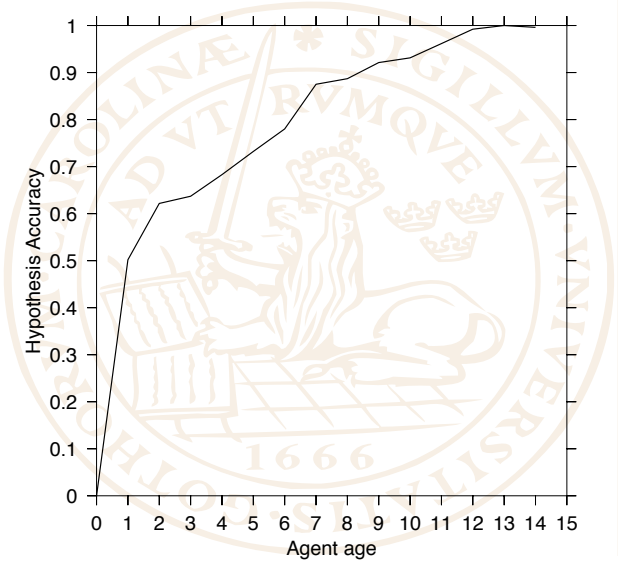
- Agent only executes plans proven to be safe
  - knows the domain good enough
  - still gains experience after each episode
- Plans to be executed are selected at random
  - from among the safe ones
- Learning can still be useful
  - not for learning any *new* information
  - but for saving computational effort
- Limited retention ability
  - only two training examples per episode
- Ignoring useless plans early allows the agent not to waste deduction effort on them

LUND UNIVERSITY

AI@CS

Department of Computing Science

Conservative Agent Evaluation

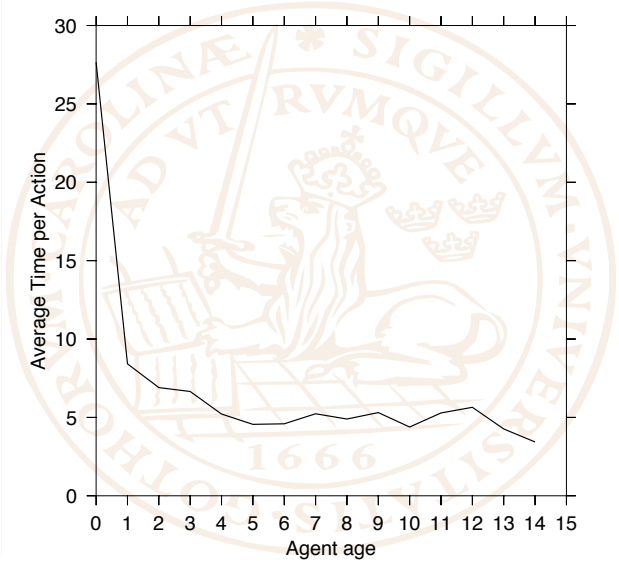


LUND UNIVERSITY

AI@CS

Department of Computing Science

Conservative Agent Evaluation



LUND UNIVERSITY

AI@CS

Department of Computing Science

Ranking Plans

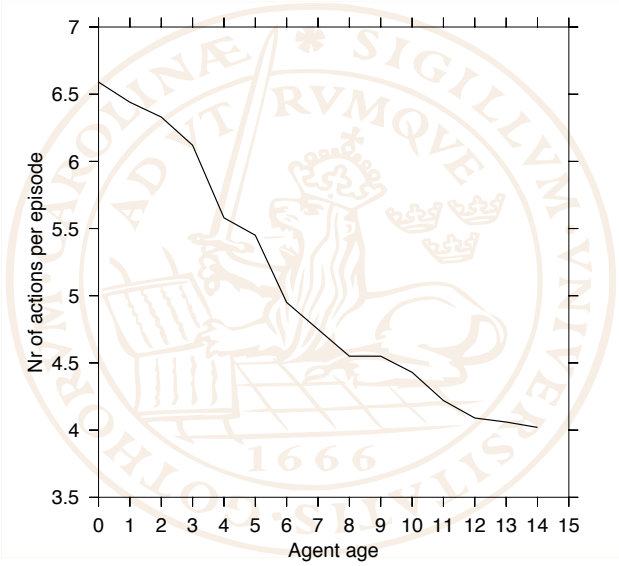
- Using classification learning algorithm
- Rank plans via set of pairwise comparisons
  - using simulator of an environment
  - Monte Carlo method
- Only sparse data set is reliable
  - not usable directly for action selection
- Knowledge representation is not rich enough
- Able to distinguish plans providing *some* new knowledge from plans which provide *nothing*
  - but not between two reasonable plans
- Still, much better than random walks

LUND UNIVERSITY

AI@CS

Department of Computing Science

Plan Selection Evaluation



LUND UNIVERSITY

AI@CS

Department of Computing Science

Questions?