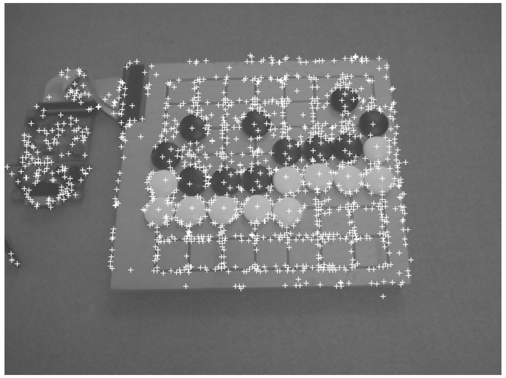
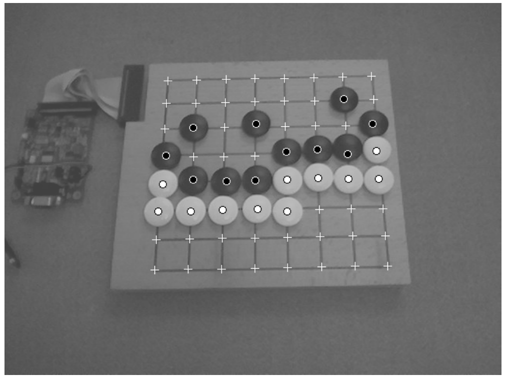
Artificial Intelligence
Basics





Univ.-Lektor Dr.techn. Alexander K. Seewald

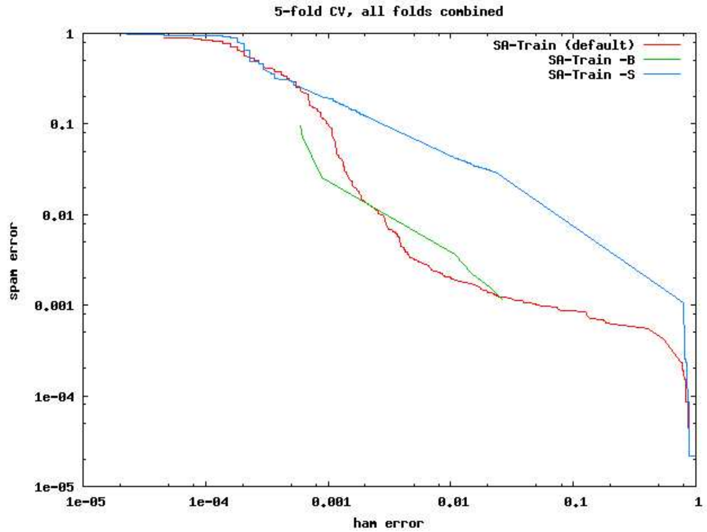
Spam Filtering

Problem

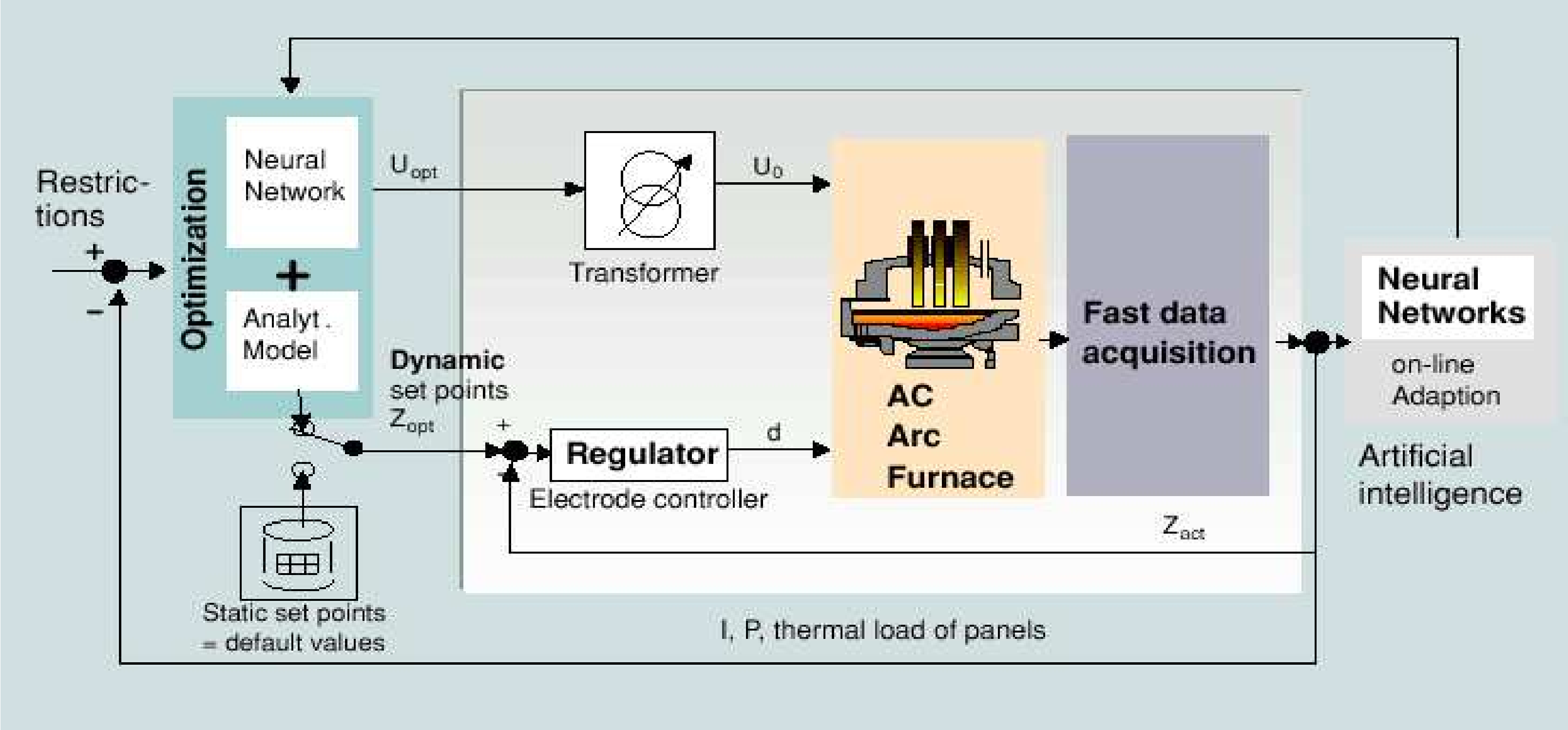
- Spam : Nonspam = 17 : 1; 200 spams/day

Solution: State-of-the-Art System

- Bayesian filter - provably better than human
- Deletes 99.8% of spam
- Few nonspam mails deleted (<0.1%)
- Low maintenance, working towards zero maintenance



Stahlwerk Bous & Siemens



- Optimization of melting process with NN and analytical model: Steel production +6,0%; Energy consumption -3,1%

RoboSail Systems

- Autopilot for one-person sailing
- Race-proven with various state-of-the-art AI and ML components.
- Human jargon like *gust*, *close-hauled*, *luff* as background knowledge!

What is Artificial Intelligence?

Systems that think like humans	**Systems that think rationally**
"The exciting new effort to make computers think... machines with minds, in the full and literal sense" (Haugeland, 1985) "[The automation of] activities that we associate with human thinking, activities such as decision-making, problem solving, learning..." (Bellman, 1978)	"The study of mental faculties through the use of computational models" (Charniak and McDermott, 1985) "The study of the computations that make it possible to perceive, reason and act" (Winston, 1992)
Systems that act like humans	**Systems that act rationally**
"The art of creating machines that perform functions that require intelligence when performed by people" (Kurzweil, 1990) "The study of how to make computers do thinks at which, at the moment, people are better" (Rich and Knight, 1991)	"A field of study that seeks to explain and emulate intelligent behavior in terms of computational processes" (Schalkoff, 1990) "The branch of computer science that is concerned with the automation of intelligent behavior" (Luger and Stubblefield, 1993)

Systems that act like humans

The Turing Test

Computing machinery and intelligence [Turing, 1950]

- "Can machines think?" ⇒ "Can machines behave intelligently?"
- Operational test for intelligent behavior = Imitation Game

- Pred. 30% chance for machine to fool lay person for 5mins
- Anticipated all major arguments against AI(!)
- Suggested major components of AI: knowledge, reasoning, language understanding, learning

Problem: Turing test is not reproducible and not constructive.

Systems that think like humans

Cognitive Science

1960s Cognitive Revolution: information processing psychology replaced prevailing orthodoxy of behaviourism

Requires scientific theories of brain's internal activities

- Abstraction - level of Knowledge, Assemblies, Neurons...
- Validation - requires predicting and testing behavior of human subjects (top-down = Cognitive Science); and direct identification from neurological data (bottom-up = Cognitive Neuroscience)

Both approaches are distinct from AI; but still share direction.

Systems that think rationally

Laws of Thought

- Normative (or prescriptive) rather than descriptive.
- Aristotle: what are correct arguments / thought processes?
- Several Greek schools developed various forms of logic = notation and rules of derivation for thoughts; may or may not have proceeded to the idea of mechanization.

- Direct line via mathematics and philosophy to modern AI

Problems

- Not all intelligent behavior is related to logical deliberation
- The purpose of thinking = What thoughts should I have?

Alexander K. Seewald
alex@seewald.at / alex.seewald.at

Systems that act rationally

Doing the right thing

- Rational behaviour: doing the right thing
- The right thing: which is expected to maximize goal achievement given the available information
- Doesn't necessarily involve thinking, but thinking should be in the service of rational action.

Aristotle (Nicomachean Ethics):

Every art and every inquiry, and similarly every action and pursuit, is thought to aim at some good.

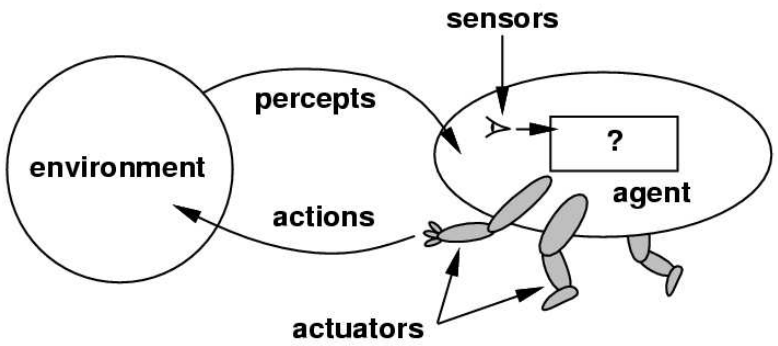
AI prehistory

Philosophy	logic, methods of reasoning mind as physical system foundations of learning, language, rationality
Mathematics	formal representation and proof algorithms, computation, (un)decidability, (in)tractability, probability
Psychology	adaptation, phenomena of perception and motor control, experimental techniques
Economics	formal theory of rational decisions
Linguistics	knowledge representation, grammar
Neuroscience	plastic physical substract for mental activity
Control theory	homeostatic systems, stability simple optimal agent designs

History of AI

1943	McCulloch & Pitts: Boolean circuit model of brain
1950	Turing's *Computing Machinery and Intelligence*
1952-69	Look, Ma, no hands! - Phase
1950s	Early AI programs: Samuel's checkers, Newell & Simon's Logic Theorist; Winograd's Blocks World
1956	Dartmouth meeting: Artificial Intelligence adopted
1965	Robinsons complete logical reasoning algorithm
1966-74	AI discovers computational complexity
1969-79	Early development of knowledge-based systems
1980-88	Expert systems industry booms
1988-93	Expert systems industry busts: "AI Winter"
1988-	Resurgence of probability; increase in technical depth
	"Nouvelle AI": ALife, GAs, soft computing
1995-	Agents metaphor

Agents and environments



An agent is everything that perceives and acts.

The whole field of AI can be viewed as being concerned with design of intelligent agents.

Agents include humans, robots, softbots, vacuums cleaners...

The agent function maps from percept histories to actions:

$$f: P^* \rightarrow A$$

For any given class of environments and tasks, we seek the agent with the best performance. Computational limitations make perfect rationality unachievable.

Alexander K. Seewald
alex@seewald.at / alex.seewald.at

Types of agents

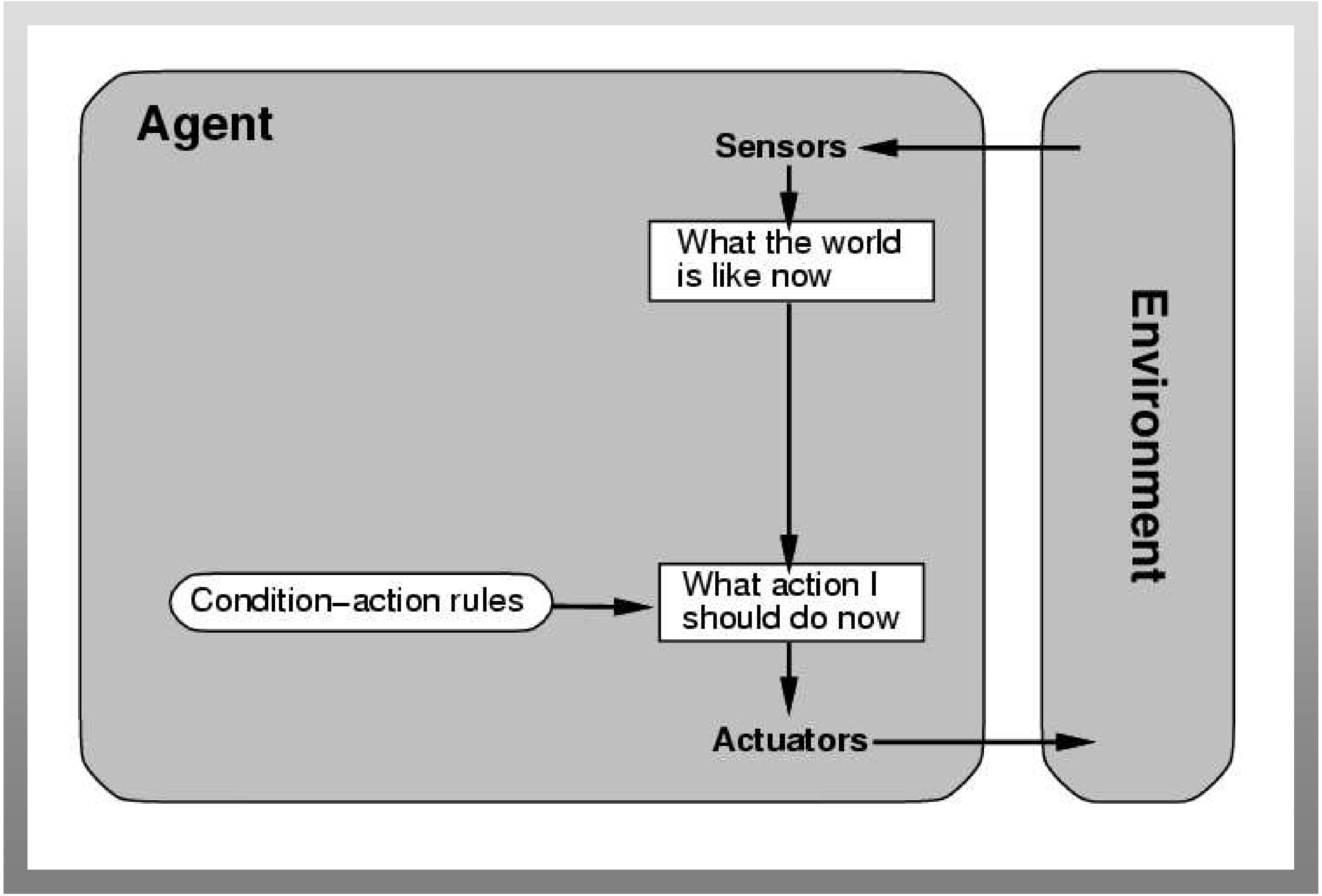
Four basic agent types in order of increasing generality:

- Simple reflex agent
- Reflex agent with state
- Goal-based agent
- Utility-based agent

All these can be turned into learning agents, where some aspects of the agent can be changed by experience.

Learning is the central issue for intelligent agents. The research fields of Machine Learning and Data Mining have investigated simpler learning model for decades. While a general learning agent is still decades away, ML & DM are well on the way towards a mature field.

Simple reflex agent



Example: Vacuum cleaner agent

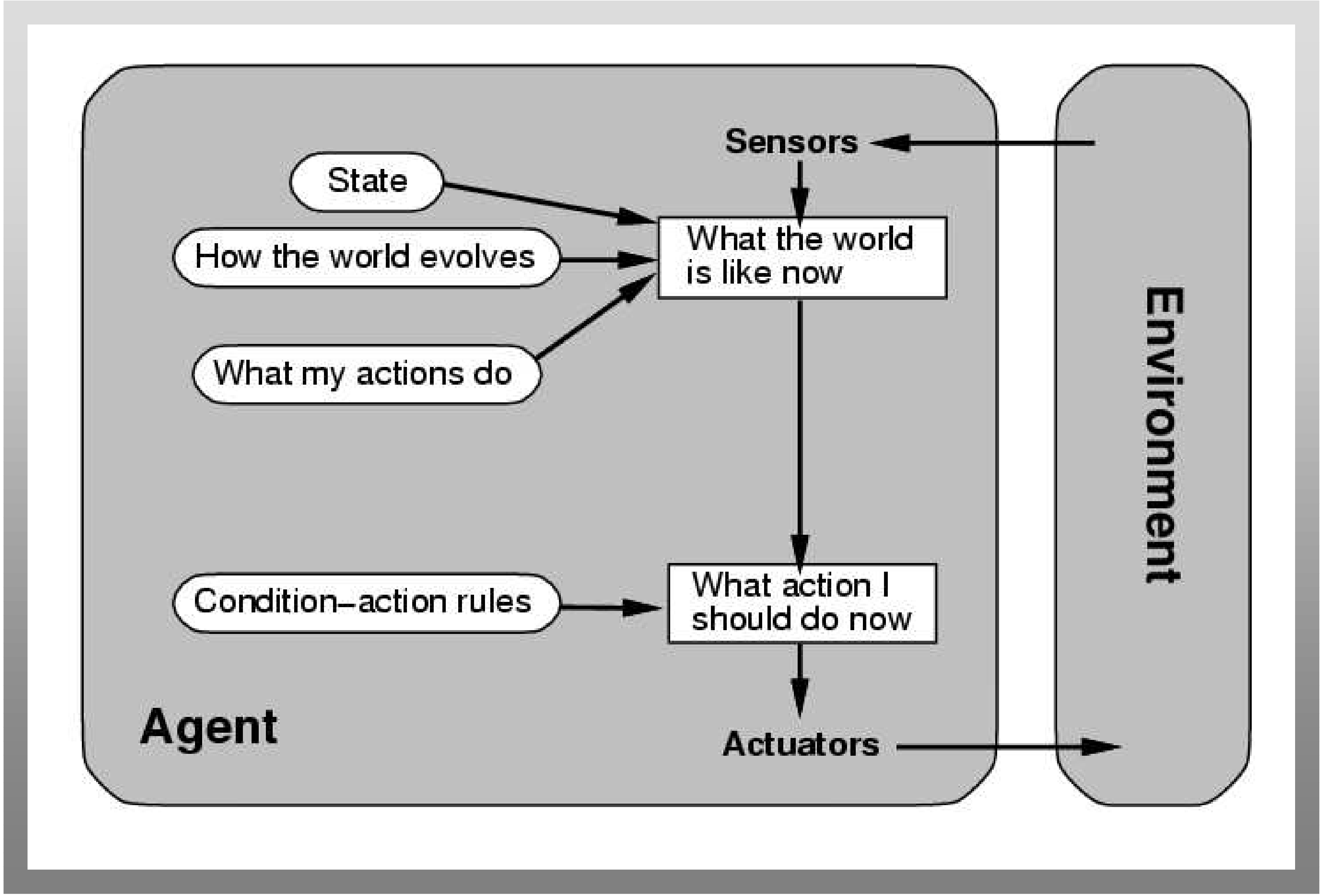
Percepts: clean/dirty, wall, stairs

Actions: move, rotate, clean

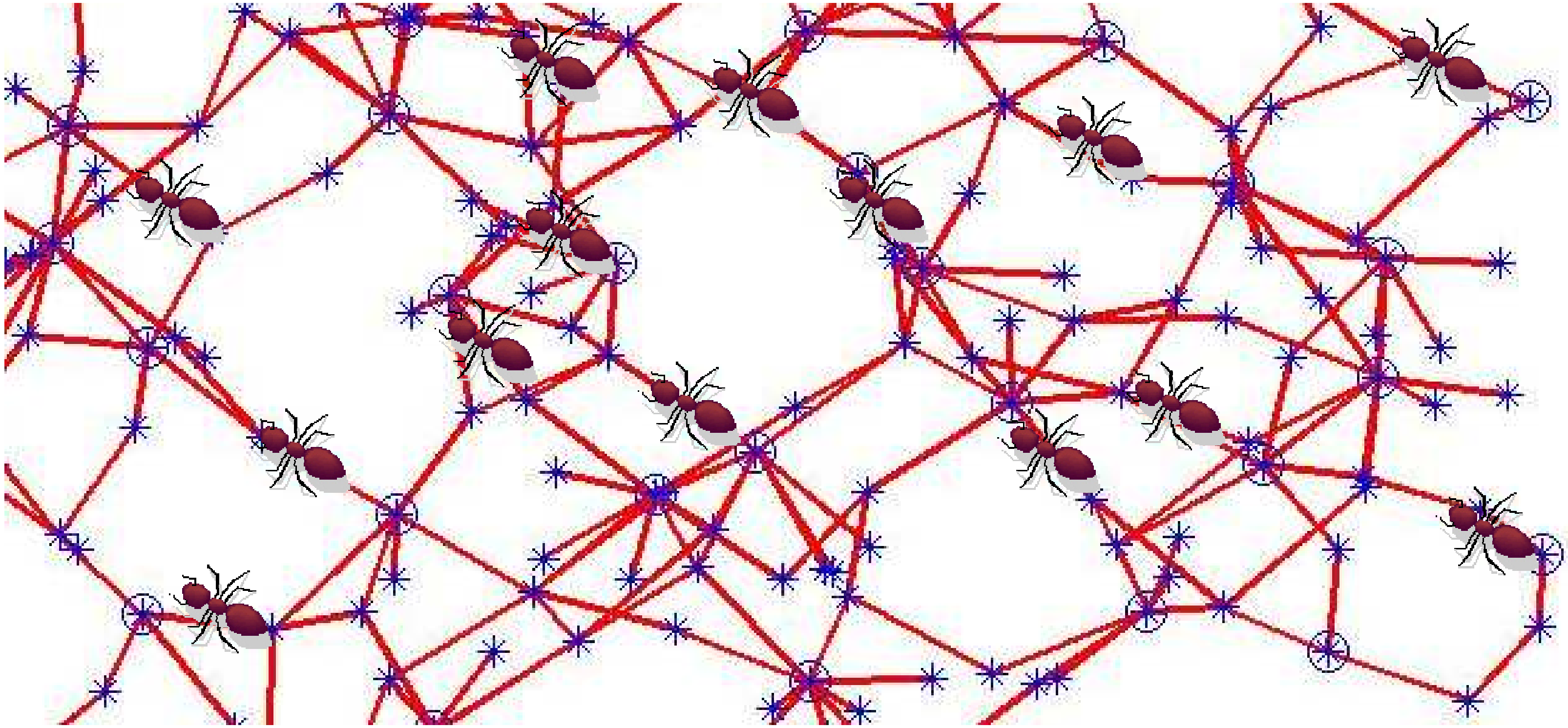
Goals: maximize amount of dirt collected / cleanliness

Environment: single-level household

Reflex agent with state

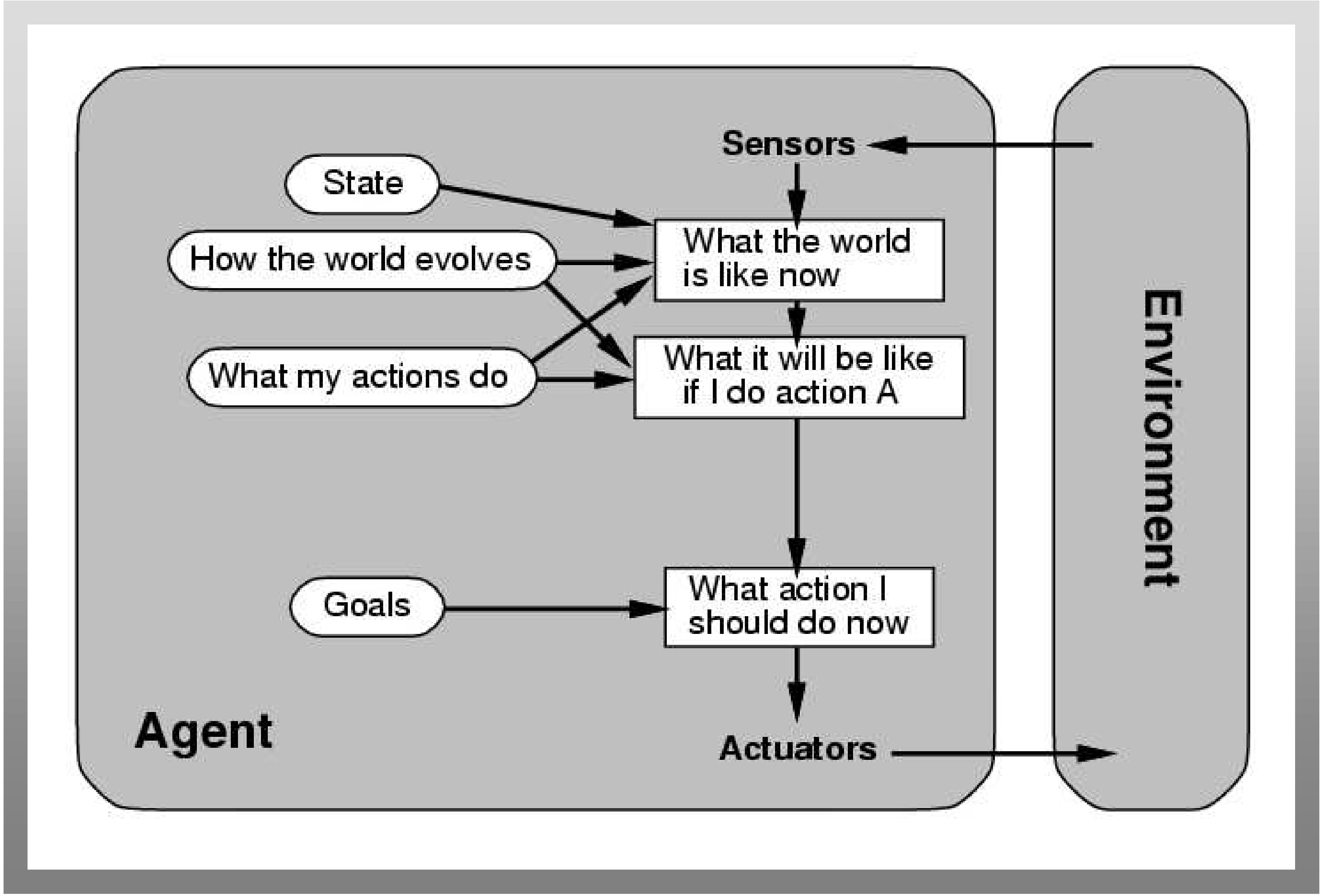


Example: Ant-based routing

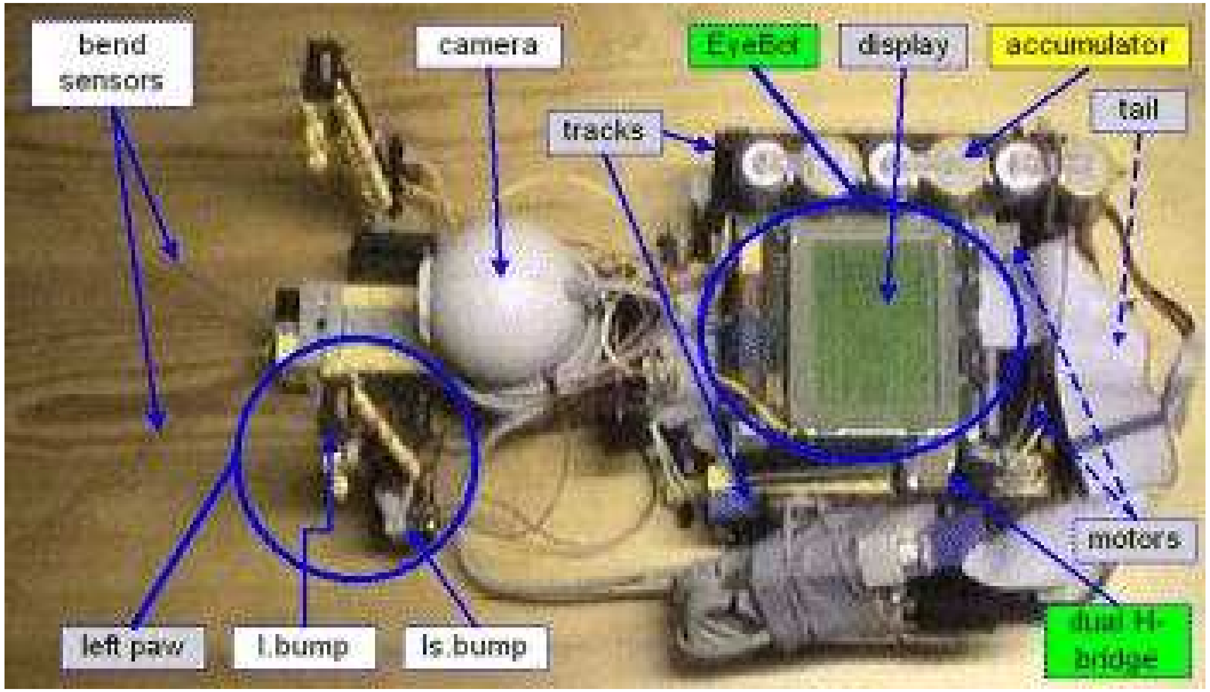


[Di Caro & Dorigo, 1998] have shown that ant-based routing outperforms other common routing methods. State is the history of visited nodes; similar to pheromone tracks in real ants.

Goal-based agent

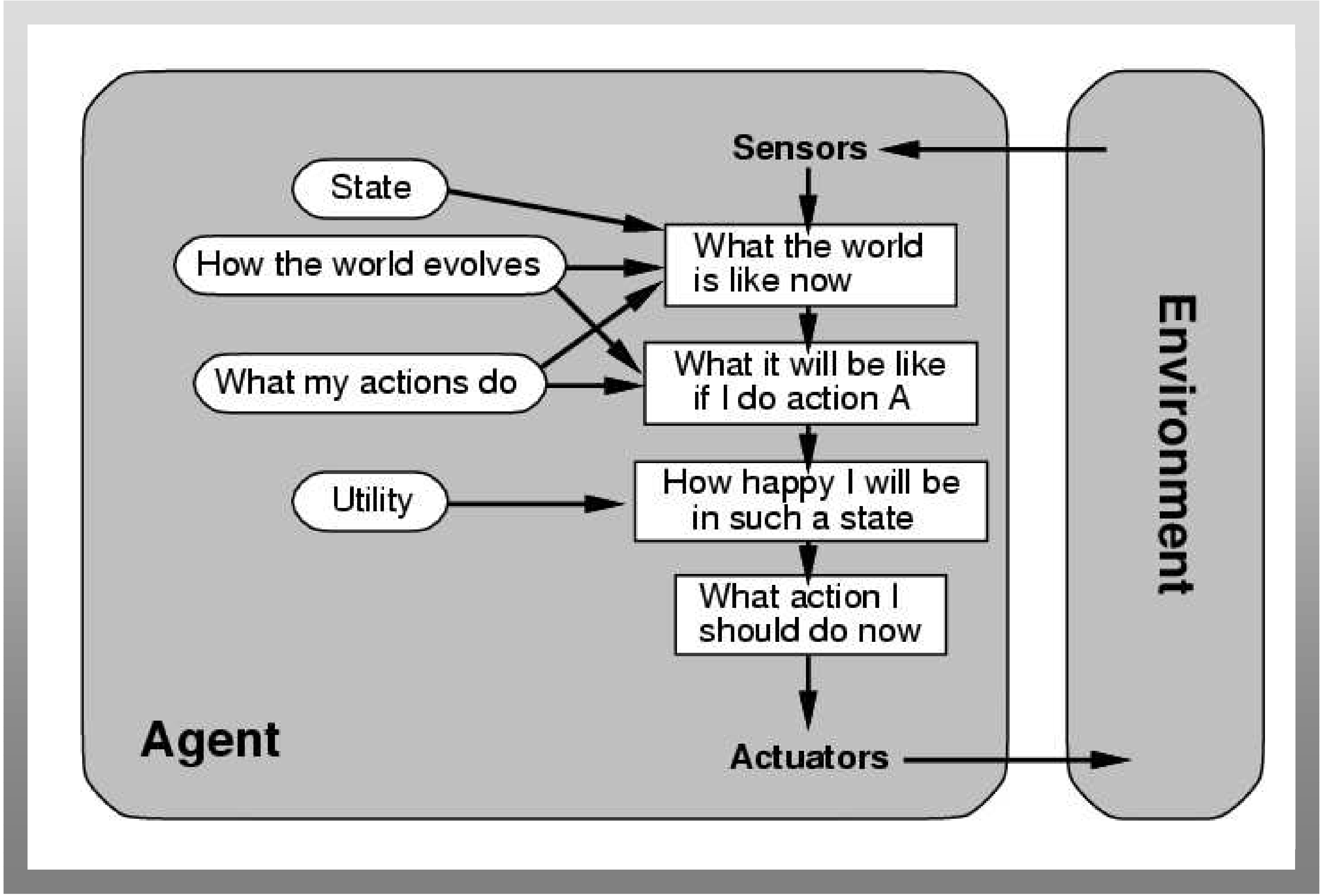


Example: RoboCat



RoboCat (Seewald, 1999; Diploma thesis) is an example for a goal-based robot. The goal in that case was to follow and hit blue objects - balls, mostly.

Utility-based agent

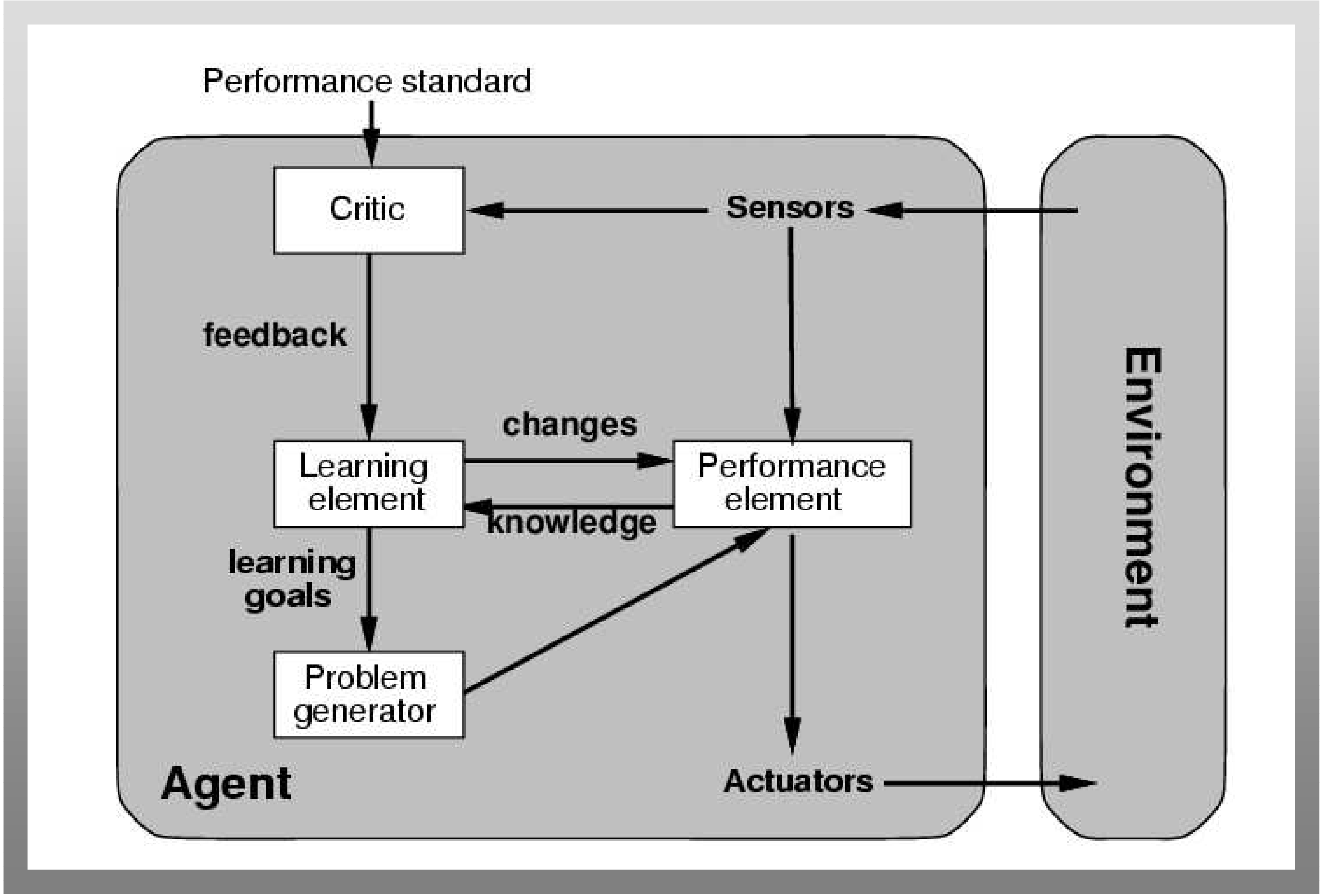


Example: Invisible Person

sources\IP_TTT.MPG

The Invisible Person project with the Technical Museum in Vienna was concerned with the creation of an engaging playful agent. The agent group at ÖFAI was responsible for modelling its behaviour.

Simple Learning Agent (reflex-based)



Example: Stanley

Autonomous robot vehicle which won the DARPA Challenge 2005. Built at Stanford University in about 15 months by a team of around 35 people. Uses Machine-Learned Laser Perception and Speed Strategy.

Presenting Stanley

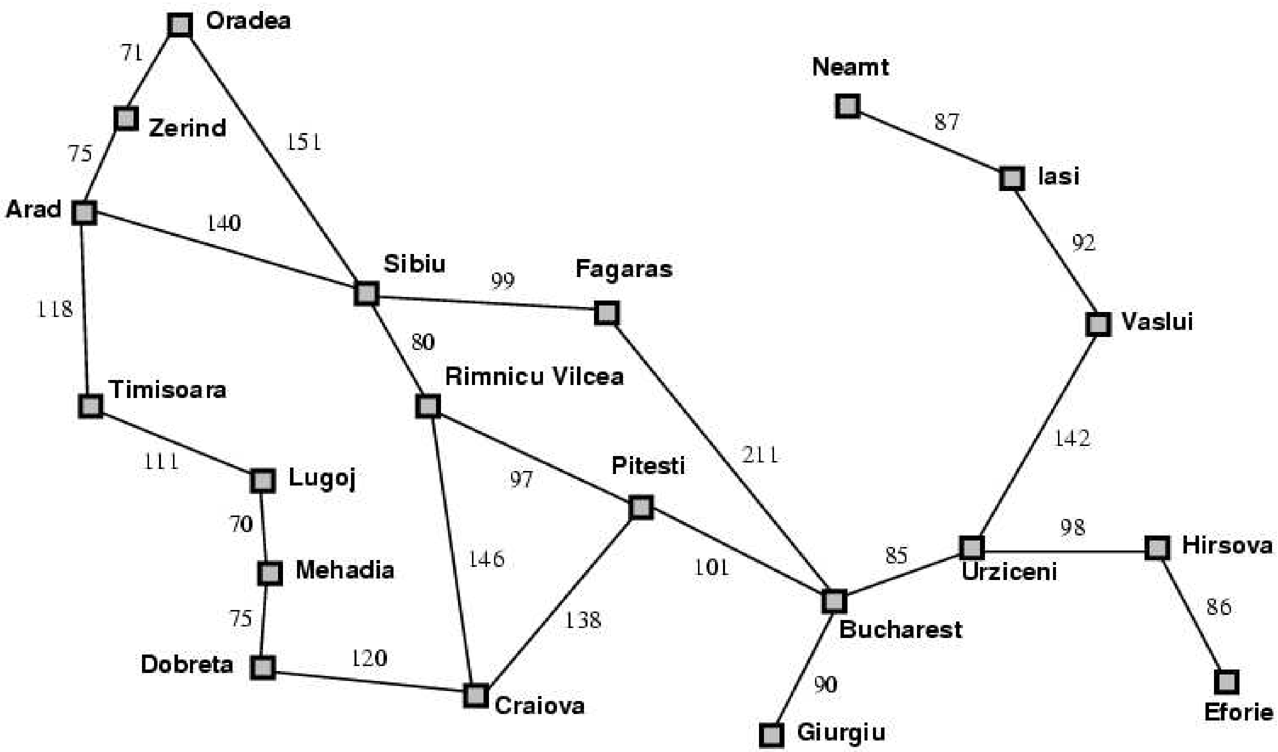
How can we build such agents?

- Search / Problem Solving
- Knowledge and Reasoning; Planning
- Acting under Uncertainty
- Decision Theory
- Communication / NLP

- Learning

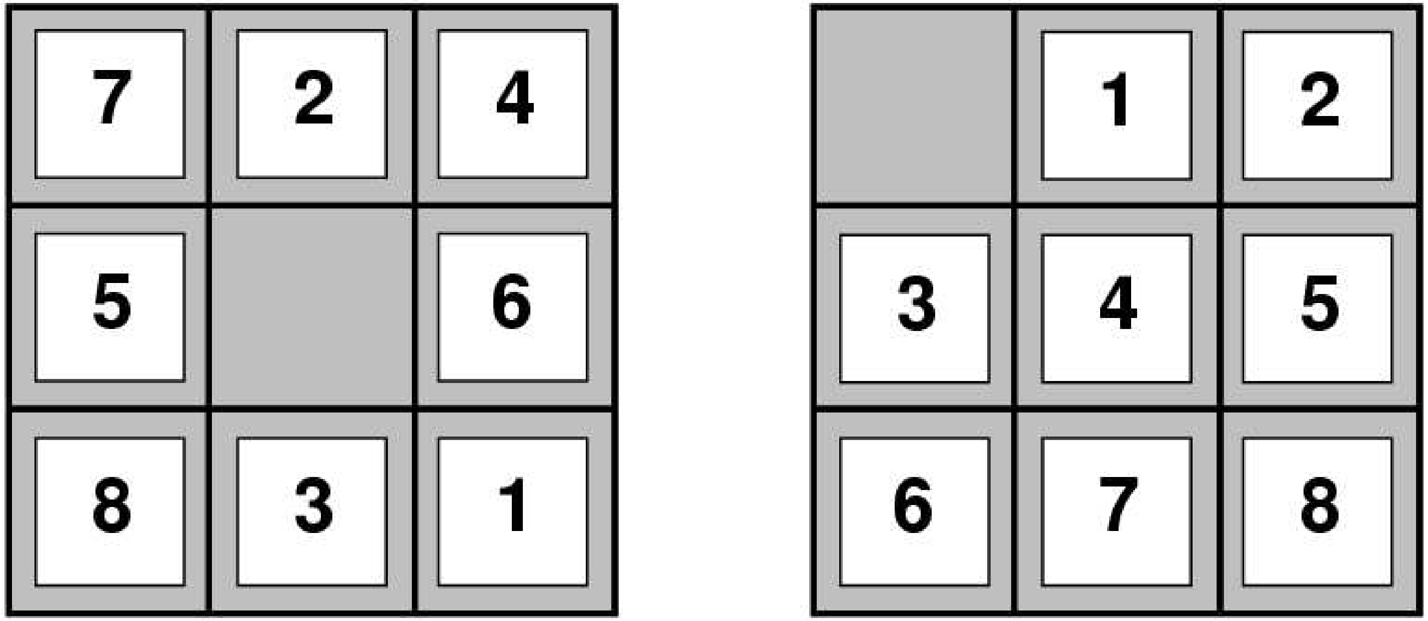
Search / Problem Solving

Search is a central theme in AI. The fastest path through a city; VLSI layout; the correct interpretation of a given sentence; and even general learning - all these can be formulated as search problems.



Search / Problem Solving

A problem consists of: the **initial state**, a set of **operators**, a **goal test** function, and a **path cost** function. The environment of the problem is represented by a **state space**.



Start State Goal State

Search / Problem Solving

A single **general search** algorithm can be used to solve any problem. Search algorithms are judged on **completeness**, **optimality**, **time complexity** and **space complexity**. Complexity depends on *b*, the branching factor; and *d*, the depth of the shallowest solution.

Breadth-first search expands the shallowest nodes in the search tree first. It is complete, optimal for unit-cost operators, and has time and space complexity of $O(b^d)$.

Uniform-cost search expands the least-cost leaf node first. It is complete, and optimal for any cost function. Its space and time complexity is the same as Breadth-first search.

Search / Problem Solving

Depth-first search expands the deepest node in the search tree first. It is neither complete nor optimal, and has time complexity of $O(b^m)$ and space complexity of $O(bm)$, where *m* is the maximum depth.

Depth-limited search places a limit on the depth of depth-first search. It is complete if the limit is greater than the depth of the shallowest solution.

Iterative deepening search calls depth-limited search with increasing limits until a goal is found. It is complete; optimal for unit-cost operators, and has time complexity of $O(b^d)$ and space complexity of $O(bd)$. *Preferred method in large search spaces when depth of solution is not known.*

Search / Problem Solving

Searching the full state-space is only feasible for very small problems. Informed search algorithms take advantages of **heuristics** to prune large portions of the search space to improve time complexity in the average case. Worst case time complexity is unchanged.

Best-first search expands the minimum cost node first. The following search strategies are variants of best-first search.

Greedy search minimizes the estimated cost to reach the goal. Search time is usually reduced, but optimality and completeness are lost.

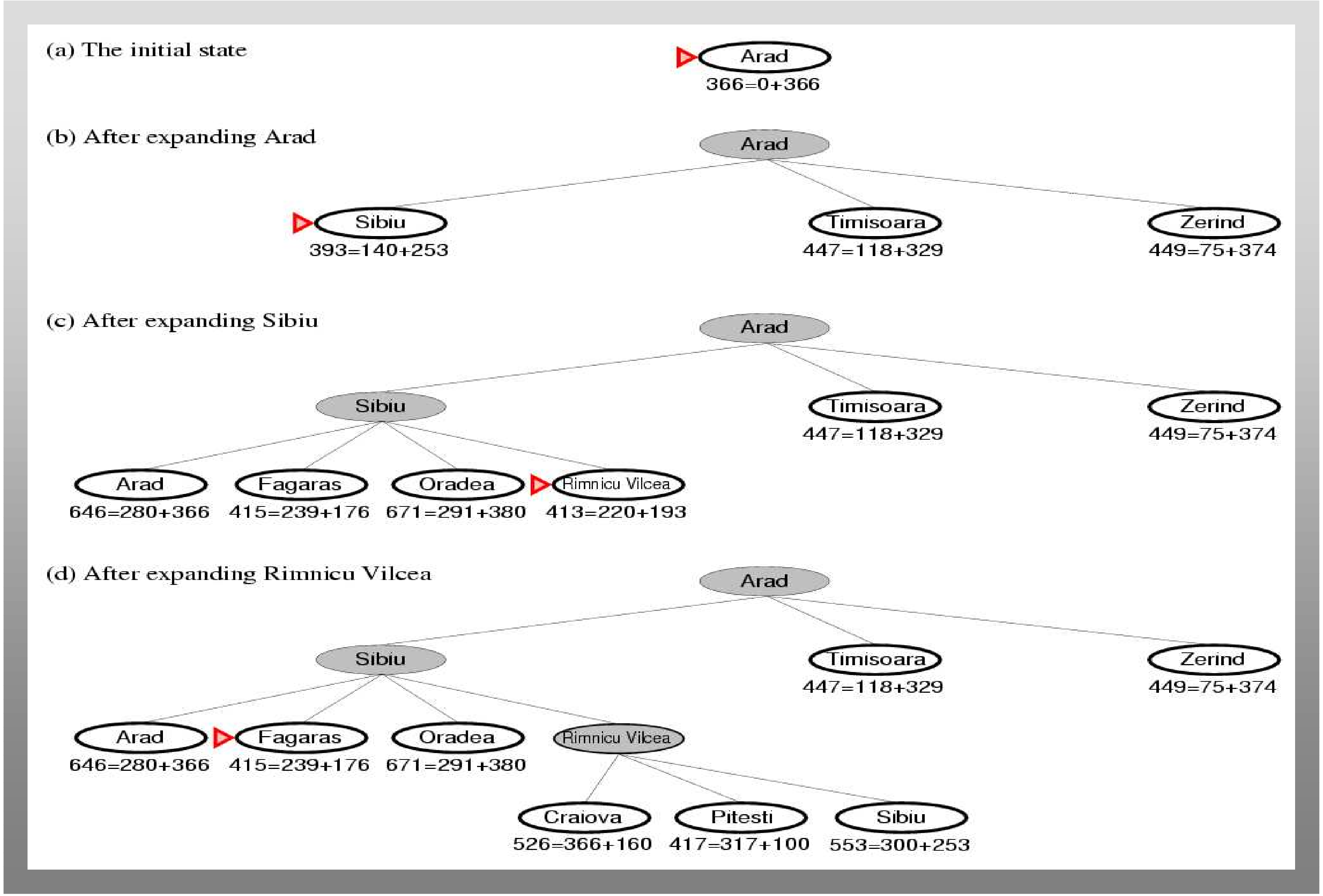
Search / Problem Solving

A* search minimizes the current cost plus the estimated cost to the goal. If the latter is never overestimated (**admissible heuristic**) and we handle repeated states, A* is complete, optimal, and **optimally efficient** among all optimal search algorithms for a given admissible heuristic. Its space complexity is still exponential in problem size.

Refinements such as **iterative deepening A*** and **simplified memory-bounded A*** address this problem.

Interestingly, some search problems are quite hard for humans, so even our refined in-built heuristics are not perfect.

Example: A* search

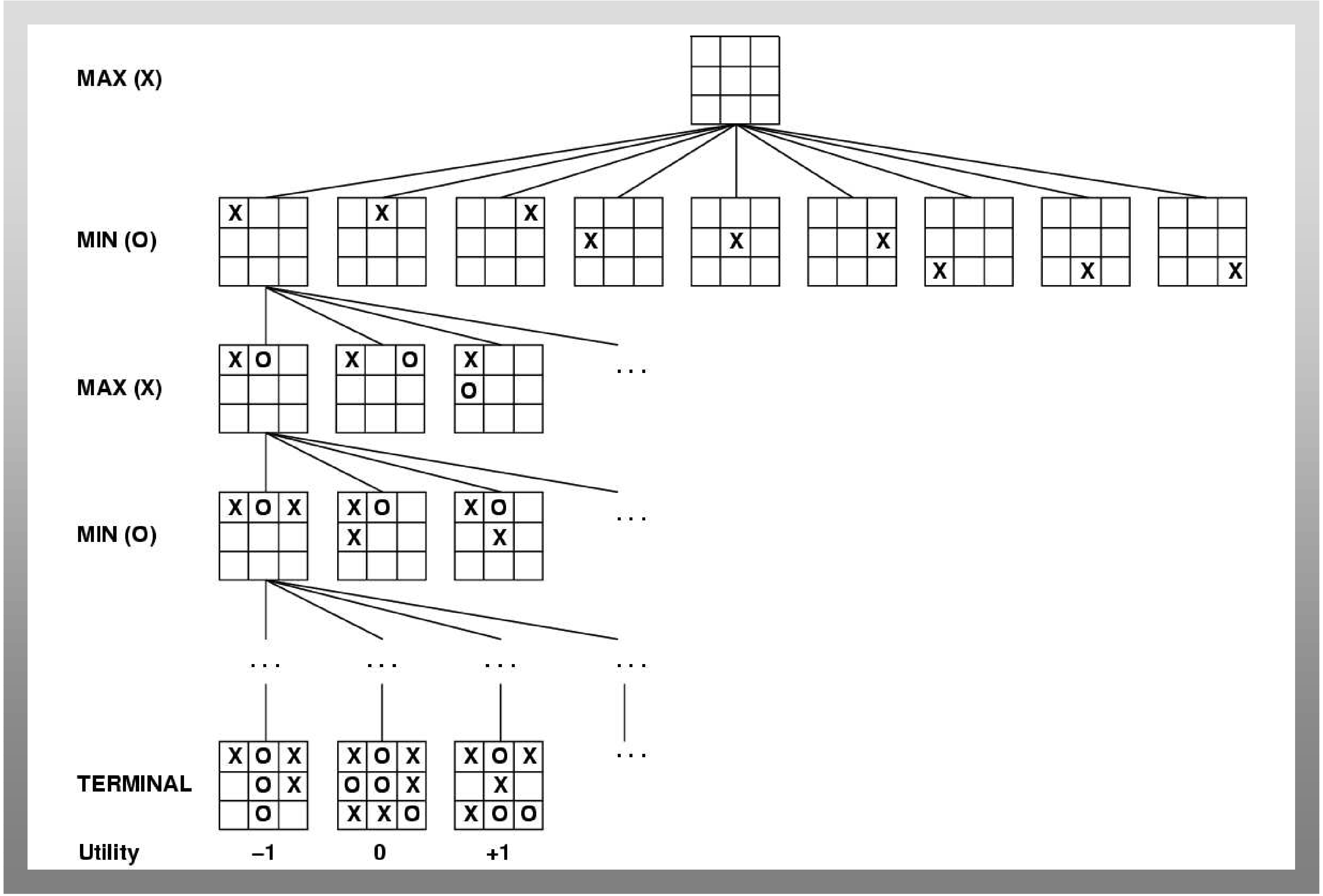


Search / Problem Solving

Iterative improvement keeps only a single state in memory, but can get stuck on local maxima. **Simulated annealing** provides a way to escape local maxima, and is complete and optimal given a long enough cooling schedule.

For **constraint satisfaction problems**, variable and value ordering heuristics provide solutions very quickly even for very large problems. Appropriate understanding and modeling of the problem domain is essential.

Example: Game as Search



Knowledge and Reasoning

Intelligent agents need **knowledge** about the world in order to reach good decisions. Humans use huge amounts of **common-sense knowledge** to solve even tiny tasks.

Knowledge is stored in the form of **sentences** in a **knowledge representation language** that are stored in a **knowledge base**.

A knowledge-based agent operaters by storing sentences about the world in its knowledge base; using an **inference mechanism** to infer new sentences, and using them to decide what action to take.

Knowledge and Reasoning

A representation language is defined by its **syntax** and **semantics**, which specify the structure of sentences and how they relate to facts in the world.

The **interpretation** of a sentence is the fact to which it refers. If it refers to a fact that is part of the world, then it is **true**.

Inference is the process of deriving new sentences from old ones. We try to design **sound** inference processes that derive true conclusions given true premises. An inference process is **complete** if it can derive *all* true conclusions from a set of premises.

Knowledge and Reasoning

A sentence that is true in all worlds under all interpretations is **valid**. If an implication sentence can be shown to be valid, then we can derive ist consequent if we know ist premise. The ability to show validity independent of meaning is essential.

Different **logics** make different commitments about what the world is made of and what kinds of belief we can have regarding facts. Logics are useful for commitments they *do not* make, because the lack of commitment gives the knowledge base writer more freedom.

Knowledge and Reasoning

Propositional logic commits only to the existence of facts that may or may not be the case in the world being represented. It has a simple syntax and semantics.

First-order logic commits to the existence of objects and relations in the world. It is useful for complex concepts.

Knowledge about actions and their effects can be represented via a **situation calculus**. This knowledge enables the agent to keep track of the world and to deduce the effects of plans of action.

Knowledge and Reasoning

Knowledge engineering is concerned with building a useful knowledge base. **Knowledge acquisition** is the process by which the knowledge engineer becomes educated about the domain and elicits the required knowledge.

The process of representing knowledge consists of deciding what kinds of **objects** and **relations** (= the ontology) need to be represented. Then a **vocabulary** is selected, and used to encode general knowledge of the domain.

After encoding specific problem instances, automated **reasoning** procedures can solve them - via a process strongly related to search with admissible heuristics.

Knowledge and Reasoning

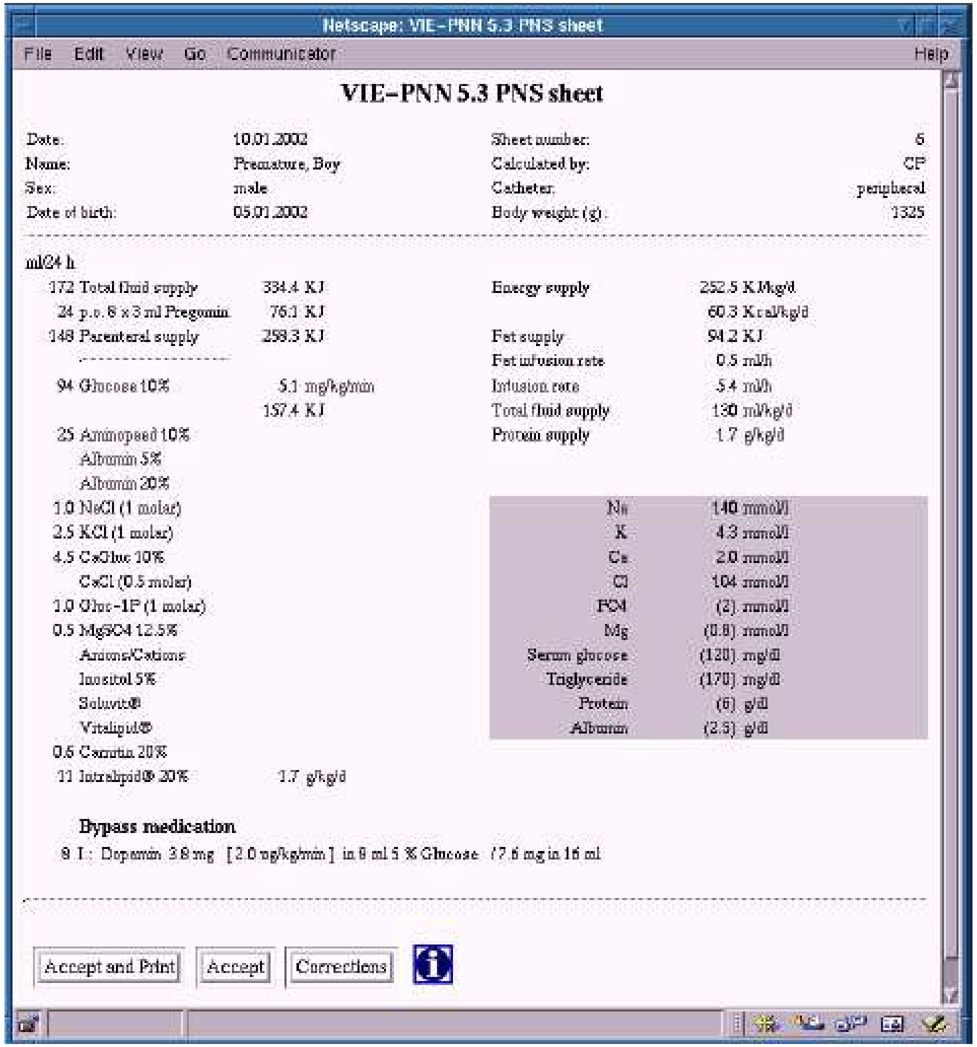
Good **representations** eliminate irrelevant detail, capture relevant distinctions, and express knowledge at the most general level possible, without being overly comprehensive

Constructing **knowledge-based systems** has advantages over programming, but is not feasible for all problems. Modeling relevant knowledge for a task may be infeasible.

State-of-the-Art are **embedded AI** systems, where AI is used complementary to other programming techniques.

Alexander K. Seewald
alex@seewald.at / alex.seewald.at

Example: VIE-PNN



- Knowledgebased system for neo-natal nutrition
- Rules derived from expert knowledge.
- HTML-based interface.
- In clinical use for >5 years at AKH Vienna

Planning

Planning agents *look ahead* to come up with actions that will contribute to goal achievement. They differ from problem-solving agents in their use of more flexible representations of state, actions, goals, and plans.

Planning systems can be seen as efficient special-purpose reasoning systems designed to reason about actions; or as efficient search algorithms for the space of possible plans.

The **STRIPS** language describes actions in terms of their preconditions and effects. It captures much of the expressive power of situation calculus. Not all domains and problems can be described in STRIPS.

Planning

STRIPS is too restricted for complex, realistic domains, but can be extended in several ways; extensions of STRIPs are still used in many realistic planning domains.

Hierarchical decomposition allows nonprimitive operators to be included in plans, with a known decomposition into move primitive steps. This is most effective when it serves to prune the search space.

Many actions consume **resources**. It makes sense to treat these as numeric measures in a pool. Time is one of the most important resources. With a few exceptions, time can be handled like any other resource.

Planning

It is not feasible to search through the **space of situations** in complex domains. Instead we search through the **space of plans**. For problems in which most subplans do not interfere with each other, this will usually be efficient; otherwise more complex domain-specific search strategies are needed.

The principle of **least commitment** states that a planner should avoid making decisions until they are needed. Partial ordering constraints and uninstantiated variables allows to follow a least commitment approach.

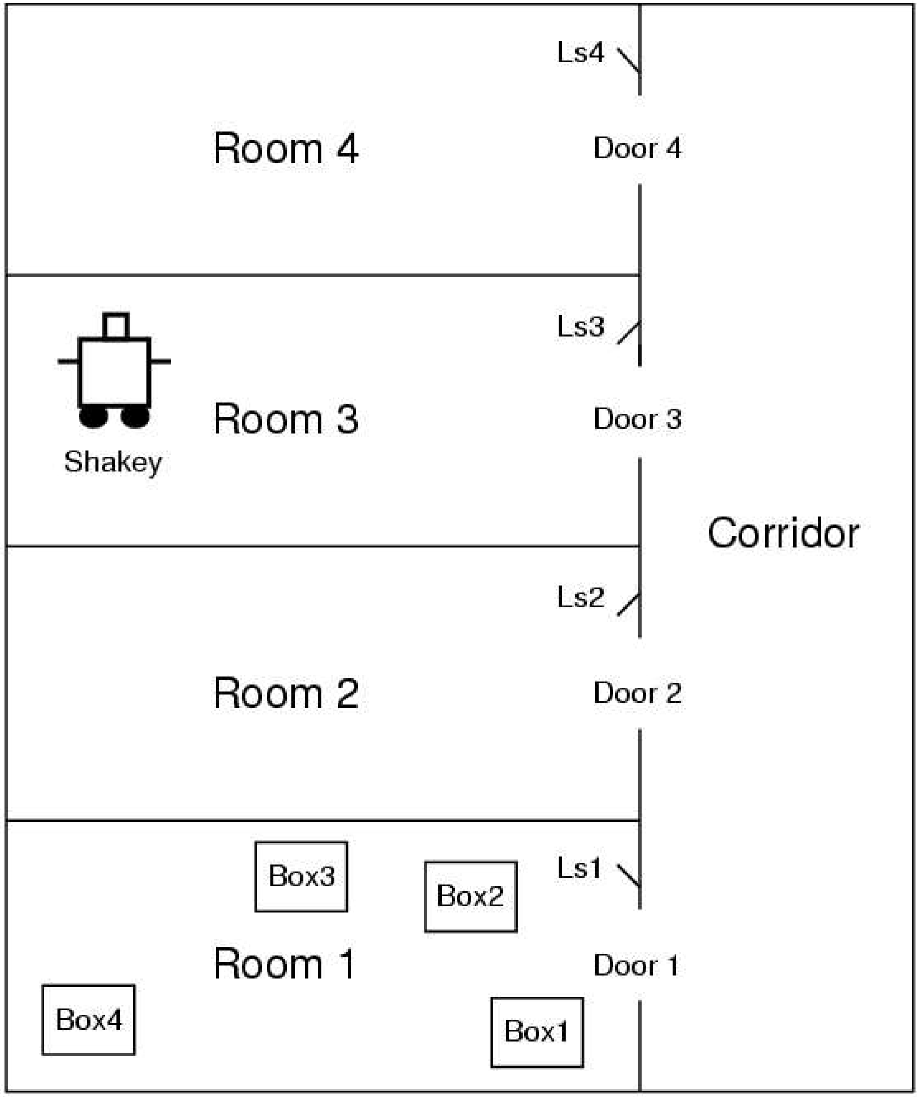
Planning

Execution monitoring is essential to ensure robustness. **Conditional planning** takes failures into account when planning; **Replanning** recomputes the whole plan on failure. These are two points on a continuous spectrum.

Scheduling takes a given plan and creates an appropriate schedule of execution. Scheduling can be formulated as constraint-satisfaction problem, with time being treated mostly like any other resource.

Automatic planners and schedulers have proven capable of handling complex domains such as spacecraft missions and manufacturing.

Example: Shakey



Acting under Uncertainty

Uncertainty is inescapable in complex, dynamic or inaccessible worlds; and means that many simplifications that are possible with deductive inference are no longer valid. **Probability theory** provides a way of summarizing the uncertainty that comes from laziness and ignorance.

Basic probability statements include **prior probabilities** and **conditional probabilities** over simple and complex propositions. The **joint probability distribution** specifies the probability for assigning values on all variables.

Bayes' Rule allows unknown probabilities to be computed from known, stable ones.

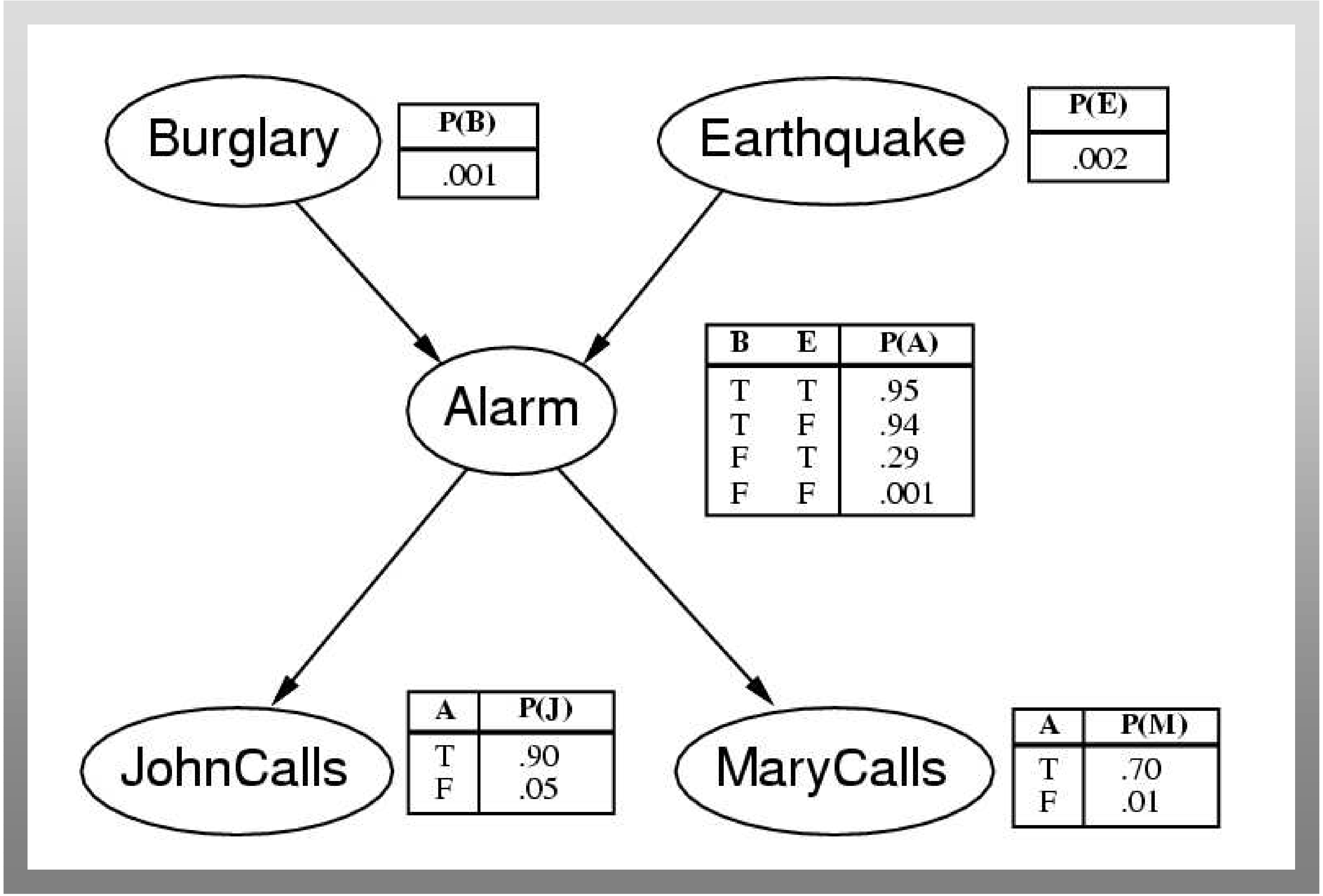
Acting under Uncertainty

Conditional independence information is a vital and robust way to structure information about uncertain domains.

Belief networks are a natural way to represent conditional independence information. The links between nodes represent the qualitative aspects of the domain, and the conditional probability tables represent the quantitative aspects.

The complexity of belief network inference depends on the network structure. Inference mechanisms are of exponential complexity in the worst case; in real domains, the local structure makes inference more feasible.

Example: Burglar alarm



Decision Theory

Simple decision problems can be solved by **decision theory**, which relates what an agent wants (**utility theory**) to what an agent should believe on the basis of evidence (**probability theory**) Utility theory associates a utility value to each state of the agent.

We can use decision theory to build a system that make decisions by considering all possible actions and choosing the one that leads to the best expected outcome. Such a system is known as a **rational agent**.

Decision theory is **normative** - it describes rational behaviour. It is probably not **descriptive** - people systematically violate the axioms of utility theory.

Decision Theory

More complex sequential decision problems in uncertain environments can be solved by calculating a **policy** that associates an optimal decision with every state that the agent might reach.

Methods to calculate optimal policies are closely related to the general computational technique of **dynamic programming**, which considers all possible paths in an efficient way.

Question to the audience

What would you prefer?

A) 80% chance of winning €4000

B) 100% chance of winning €3000

[Allais, 1953] found that people strongly prefer B)

C) 20% chance of winning €4000

D) 25% chance of winning €3000

[Allais, 1953] found that people strongly prefer C)

No consistent utility theory for humans is possible!

$0.8U(€4000) < U(€3000)$ and $0.25U(€3000) < 0.2U(€4000)$ cannot both be satisfied.

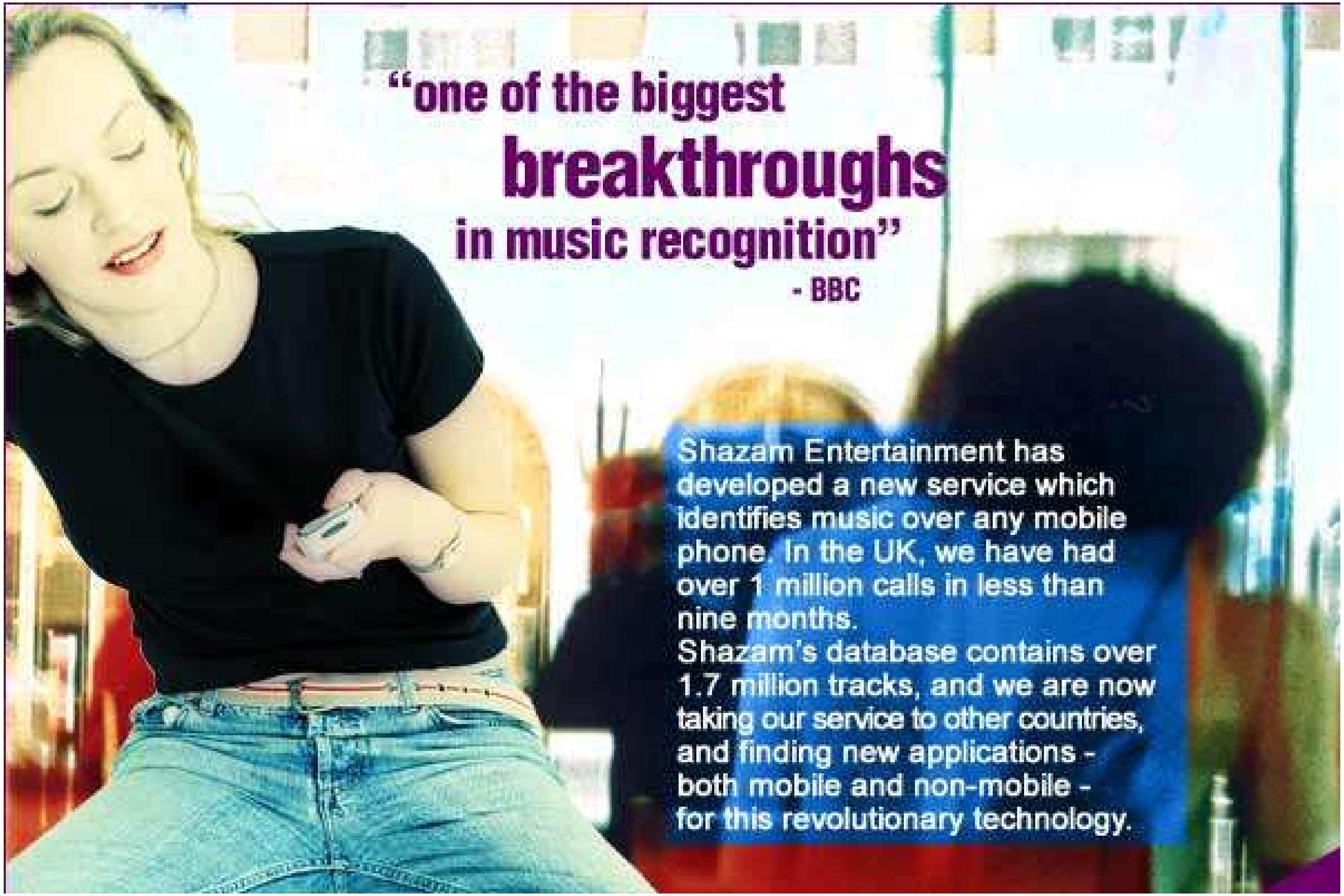
Communication

Agents need to communicate to each other and to the users. Communication between learning agents is an active research area which sheds light on the development of language in humans.

Natural language processing techniques make it practical to develop programs that make queries to a database, extract information from texts, translate languages, or recognize spoken words.

In all these areas, there exist programs that are useful, but there are no programs that do a throrough job in an open-ended domain.

Shazam Entertainment



Agents as programming metaphor

- Procedural (classic) programming
- Declarative programming
- Object-oriented programming
- Constraint logic programming
- Event-oriented programming
- Knowledge-based software engineering
- Agent-based software engineering

...

Each of these gives an unique viewpoint on programming; makes solving some problems easier and others harder. *But* you still need a programmer!

For learning systems, you don't need a programmer. Most of the work is done by learning systems.

Learning

Learning in intelligent agents is essential for dealing with unknown environments; and for building agents without prohibitive amount of work. All learning suffers from the **credit assignment** problem = which steps are responsible for a good or bad outcome?

Reinforcement learning is an active research topic, and computationally very expensive. Temporal difference learning and Q-Learning are common learning algorithms.

Genetic algorithms achieve reinforcement by increasing the proportion of successful functions. They achieve generalization by mutating and cross-breeding programs.

Learning

Learning a function from examples of its inputs and outputs is called **inductive learning**. Learning in the inductive setting is supervised and needs a set of training inputs and outputs.

Unsupervised learning uses the structure of training data to infer hidden relationships, which are harder to validate.

Inductive logic programming can learn relational knowledge, as used in knowledge-based systems. This kind of learning is generally very hard for larger problems.

Bias

"Bias refers to any criterion for choosing one generalization over another other than strict consistency with the observed training instances" (Mitchell, 1980)

Each learning algorithm is biased twofold:

- **language bias** = restricts possible concepts to be learned
- **search bias** = prefers certain models over others

Overfitting occurs when the structure of training data is learned too well; and the generalization performance on unseen data suffers.

Bias is essential to learning!

Learning

A large variety of learning algorithms is available, which can learn:

- A state evaluation function to play checkers
- A belief network to model sleep stages
- A function to predict steel quality in production
- A function to predict insurance risks
- Logic programs to determine cancerogenity
- Association rules in supermarket basket analysis
- Time-dependent models of speech
- Response models of mailable customers

...

But learning is still hard! Why?

Inductive learning is inherently risky

- There is no safe way to predict the future.
- Bias is essential, but may be wrongly chosen.

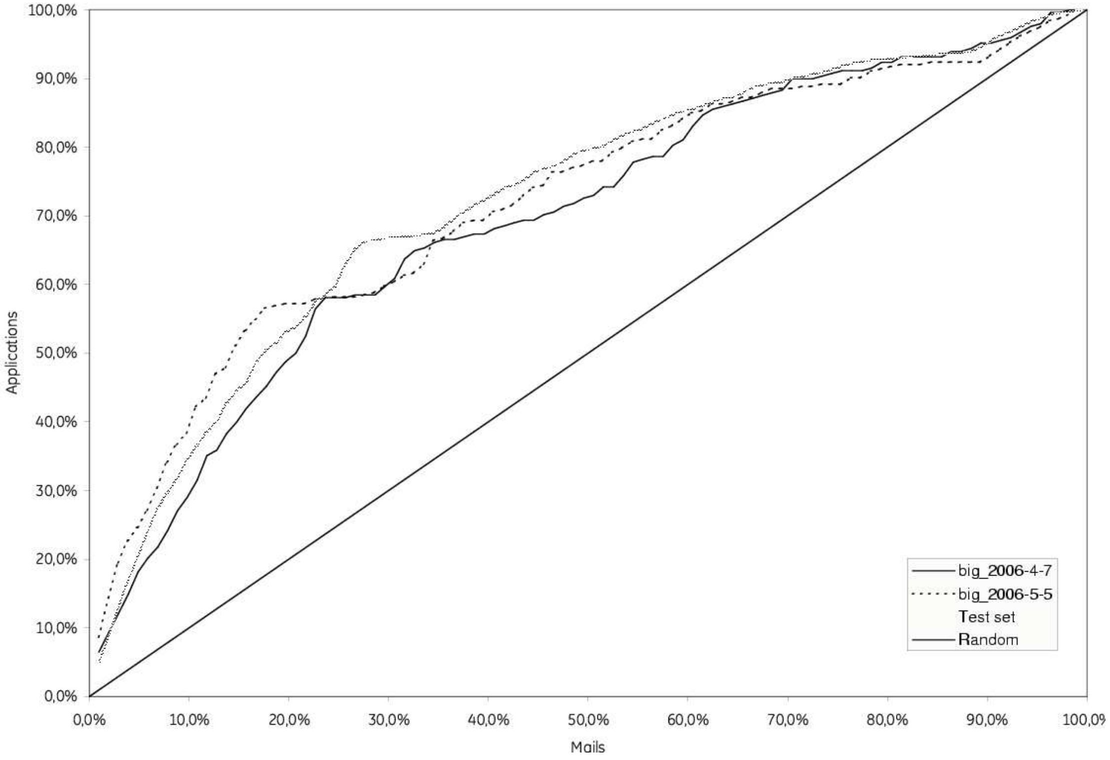
No Free Lunch!

- Theoretically, it is not possible to learn anything.
- Practically, the world shows an enormous variety of patterns. Life has adapted over billions of years to take advantage of these specific patterns.

Example problem: Response Model

Problem: Not enough capacity to mail all customers.

⇒ Improve effectiveness by learning a response model: 30% higher volume with same cost.



Example problem: Churn

Given: A set of customers with state, area code, telephone number, and time/cost information for calls in one month; plus churn = have they switched to another provider by the end of the month?

Create a useful model of customer churn, so it can be reduced significantly!

Rules for Churn (1)

(total_day_minutes >= 245) and (total_eve_minutes >= 225.2) and (voice_mail_plan = no) and (total_night_minutes >= 170.6) => churn=True. (64.0/0.0)

(total_day_minutes >= 236.9) and (total_night_minutes >= 230.6) and (voice_mail_plan = no) and (total_eve_minutes >= 197.7) => churn=True. (12.0/1.0)

(total_day_minutes >= 223.3) and (total_day_minutes >= 264.8) and (voice_mail_plan = no) and (total_eve_minutes >= 188) and (total_night_minutes >= 132.9) => churn=True. (52.0/1.0)

(total_day_minutes >= 222.3) and (total_day_minutes >= 286.2) and (voice_mail_plan = no) and (total_eve_minutes >= 150.8) => churn=True. (17.0/2.0)

(total_day_minutes >= 221.9) and (total_eve_minutes >= 261.6) and (voice_mail_plan = no) => churn=True. (41.0/7.0)

Rules for Churn (2)

(number_customer_service_calls >= 4) and (total_day_minutes <= 160) and (total_eve_minutes <= 233.2) and (total_night_minutes <= 254.9) => churn=True. (69.0/0.0)

(number_customer_service_calls >= 4) and (total_day_minutes <= 182.1) and (total_eve_minutes <= 190.7) and (total_night_minutes <= 285) => churn=True. (22.0/0.0)

(number_customer_service_calls >= 4) and (total_day_minutes <= 135.9) and (account_length >= 72) => churn=True. (14.0/0.0)

(number_customer_service_calls >= 4) and (total_eve_minutes <= 135) => churn=True. (12.0/4.0)

Rules for Churn (3)

(international_plan = yes) and (total_intl_minutes >= 13.2) => churn=True. (54.0/0.0)

(international_plan = yes) and (total_intl_calls <= 2) => churn=True. (50.0/0.0)

=> churn=False. (2926.0/91.0)

Demo

Demonstration of the WEKA Machine Learning Workbench

Open Source, available at
http://www.cs.waikato.ac.nz/~ml/weka

Integrated into Pentaho's Open Source Business Intelligence Suite
http://www.pentaho.com

Past Projects

2000-2005	Employed at OFAI as junior researcher
2001	EEG data analysis (contributed by Brain Research institute, Vienna)
2000-2002	*A New Modular Architecture for Data Mining* (FWF)
2002	*3DSearch* (multi-document summarization, EU & uma AG)
2002-2003	*Intelligent Go Board* (embedded device to capture moves of Japanese Go during play, presented at Innovation Workship in ' 05)
2003-2005	*BioMinT* (integrated system for biological text mining, EU FP5)
2004-2006	*SA Train* (Spam training methodology for SpamAssassin, Evaluation of commercial and open-source spam filter systems)
2005	*Digits* (handwritten digit recognition: open source corpus and preliminary experiments)
2006	Employed at GE Money Bank as CRM Analyst
2006-2007	*IGO-2* (image mining on images of Go final board states)
2007	*Websuit* (image mining on GFP/DIC images contributed by Univ. of Colorado at Boulder; related to my recent ERC Ideas proposal)
2007-	Employed at Ikarus in R&D for spam filtering and virus detection