3 assignments, 15% each (since the assignments are mostly programming assignments, the students will demo them on the due date on lab machines in the TA lab or their own laptops. You may write the assignments in any language you like.)

2 in-class quizzes, 15% each (the quizzes will be based on the material covered in class and possibly also on material related to the assignments.)

1 final project, 25% (after a few weeks, students are expected to propose or ask for a suitable project. The project will be due during the second half of the exam period. At a date to be announced, all students will hand in a 1 or 2 page description of their chosen project. We'll talk about it in class...)

There is no final exam!

outline

MAJOR TOPICS:

1. history of AI; its role in CogSci
2. introduction to machine learning and data mining
3. decision tree induction
4. data mining; concept learning
5. rule covering; unsupervised acquisition of association rules
6. nearest neighbor methods, conceptual clustering
7. knowledge based systems, a basic inference engine with backward chaining, unification
8. first order logic, natural deduction; role of logic in AI, logic as a knowledge representation scheme, nonmonotonic reasoning
9. heuristic search as used in games
10. different approaches to uncertainty in knowledge based systems: fuzzy logic and fuzzy control systems
11. Bayesian inference and Bayesian networks in AI
12. introduction to evolutionary computation; genetic algorithms

PRESCRIBED TEXT(S): None. RECOMMENDED TEXT:
S. Russell, P. Norvig, 'Artificial Intelligence - A Modern Approach', Prentice Hall.

Course notes (will be posted after each lecture)

history of AI

- Leibniz, Babbage, Boole, Frege, Russell, Tarski …
- Turing (1930s)
  - **Turing machine**
  - **Turing test** - 'operationalizing' intelligence
  - Turing-Church thesis: if a problem is not solvable by a TM, then it is not solvable by people either ;-)
- 1940s: McCulloch-Pitts, N. Wiener, Ross Ashby, Grey Walter, J. v. Neumann
  - neuron models; cybernetics - feedback and teleological behavior; homeostat; machina speculatrix; self-reproducing automata
- 1950s: Simon, Newell, McCarthy, Minsky: "AI" (56)
  - Perceptron
- 1960s: Lisp, Adaline, fuzzy sets (Zadeh 65). GPS, Logic Theorist
- 1970s: backprop, fuzzy controllers, knowledge engineering, GA

history of AI, cont.

- 1970s: production systems, expert systems, nlp, SHRDLU, theorem proving, planning
- 1980s: NN / connectionist boom, Boltzmann machine, KR, more semantics in nlp (CD), symbolic machine learning
- 1990s to present: more NN, subsumption architecture, reinforcement learning, bayesian belief nets, data mining boom, more GA, GP, alife, "bottom-up or behavior-based AI" vs "top-down AI", "emergent computing", swarm intelligence, self-organization…

- Intelligence is
  intellectual (?) behavior that we admire but don't understand…
  or: intelligence is manifested in **behavior** and closely related to surviving in a complex world
  or: ….

"2" kinds of AI (or 3 or 4)
gofai vs nouvelle ai

- engineering vs "cog sci"
  - making usefully smart machines, somehow:
    - expert systems; Deep Blue; data mining
  - understanding how minds work
    - AI to express and test psych, linguistic etc theories
- classical / top-down / symbolic vs behavior-based / bottom-up / subsymbolic …. MIND vs BRAIN
  - "physical symbol system hypothesis"
    - hi-level approach is **brittle**
  - bottom-up approach often unimpressive..
- weak AI vs strong AI
  - Chinese room …
- scruffies vs neats

AI: all about tradeoffs

- theoretical insights in AI concern tradeoffs
  - tradeoffs between efficiency and generality
  - tradeoffs between robustness and power
  - tradeoffs between complexity of design and ability to degrade gracefully
  - tradeoffs between memory and inference
  - tradeoffs between memory and time
  - tradeoffs between prior cooking and achievement

AI must be scruffy...

- neatness is impossible in complex domains
- complex domains have a structure that requires solutions to be found by exploring branching paths in a search space.
  - # of branches is exponential function of path depth.
- any intelligent agent needs to find tricks, shortcuts, even in formally specified domains (like chess!)
  - Unless: infinitely large and fast computers.
- Good shortcuts cannot be worked out in advance, and they are not perfect (even in math).
- shortcuts, laziness: key to intelligence

The real world is even harder

- lack of complete initial info.
- range of things to do is large (branching factor!); search spaces are huge.
- things happen fast, and there are deadlines.
- Thus, **rapidly accessible and executable heuristics must be learned by trial and error.**
- Such heuristic rules are bound to be **fallible**
  - overgeneralization
  - poor observations, weak sensors
  - errors in measurement
  - inadequate concepts
  - noise, environmental variance

  etc...

problems with heuristics

- Rules and facts should be **consistent**
  - consistency is **undecidable**
  - (approximate) consistency checking is explosive
  - so is maintaining consistency.
- To **revise a belief**, you need fallible heuristics for finding related beliefs to be revised
  - identifying and retracting underlying assumptions etc.
- a huge reason maintenance system won't do.

Semantics is scruffy too

- our conceptual schemes are open-ended, unlike formal languages.
- There is no formal, recursive semantics for natural language:
  - we don't know the extension-assigning functions!
- concepts may turn out to be indeterminate, vague, or ambiguous, and prompt conceptual innovations.
  - empirical concepts do not have crisp necessary and sufficient conditions
  - many concepts are **theoretical**

scruffiness is inevitable

- Scruffiness is inevitable for any resource-limited being!
- No practical strategy to reduce scruffiness will work in all situations.

- **AI must be scruffy, for neat reasons**.

- Thus: study what **evolution** has come up with.
  - Of course, theories about such inevitably scruffy systems should be as neat as possible (to make them maximally falsifiable etc)

by the way...

- nearly anything you want to compute you can't compute!
  - because there are countably many Turing machines but uncountably many functions
- the interesting things you can compute are too expensive to compute - so you can't compute them!
  - exponential worst case run-time functions
    - $T(n) = kC^n$ e.g. 1 input item takes 10^{-7} sec, n=50, complexity is 2^n : $20 * 10^{13}$ years
- biological systems **must** use **approximate solutions**
  - learning: on-line regularity detection for prediction
  - experimentation and mental simulation
- "To be adaptable, an organism must be suboptimal". (S.J. Gould)

AI is **highly** interdisciplinary

many fields have contributed to AI in the form of ideas,viewpoints and techniques.

Philosophy: Logic, reasoning, mind as a physical system, etc.
Mathematics: Formal representation and proof algorithms, computation, (un)decidability, (in)tractability, probability.
Psychology: learning, phenomena of perception and motor control.
Economics: formal theory of rational decisions, game theory.
Linguistics: knowledge representation, grammar.
Neuroscience: physical substrate for mental activities.
Biology: adaptation, evolution of complex systems.
Control theory: homeostatic systems, stability, optimal agent design.
Complex Systems Theory etc etc etc....

AI systems

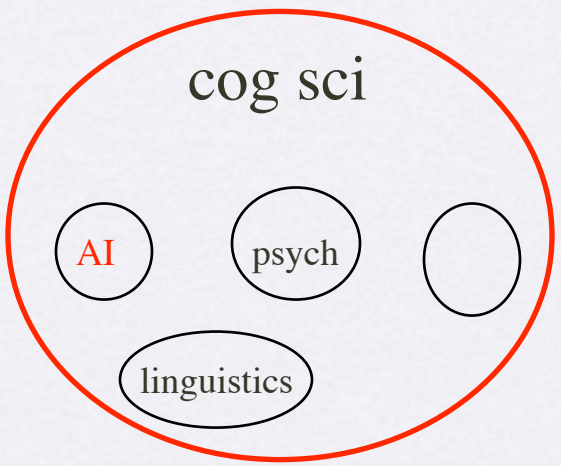
- think like humans
  - cognitive modelling (AI + psych) but ....
- act like humans
  - Turing test approach: needs nlp, kr, ml, …
- think rationally
  - FOL-based problem solving and planning, closely related to automated theorem proving
- act rationally
  - a rational agent acts so as to achieve its goals, given its beliefs - **limited rationality**

- autonomous agents, adaptive animals, evolutionary computation

14

some subareas of AI...

- heuristic search
  - problem solving, planning, game playing
- theorem proving
- knowledge-based systems
  - knowledge engineering; knowledge representation; expert systems
- nlp
  - story understanding; speech recognition; question answering
- perception, esp. vision
- robotics
- machine learning



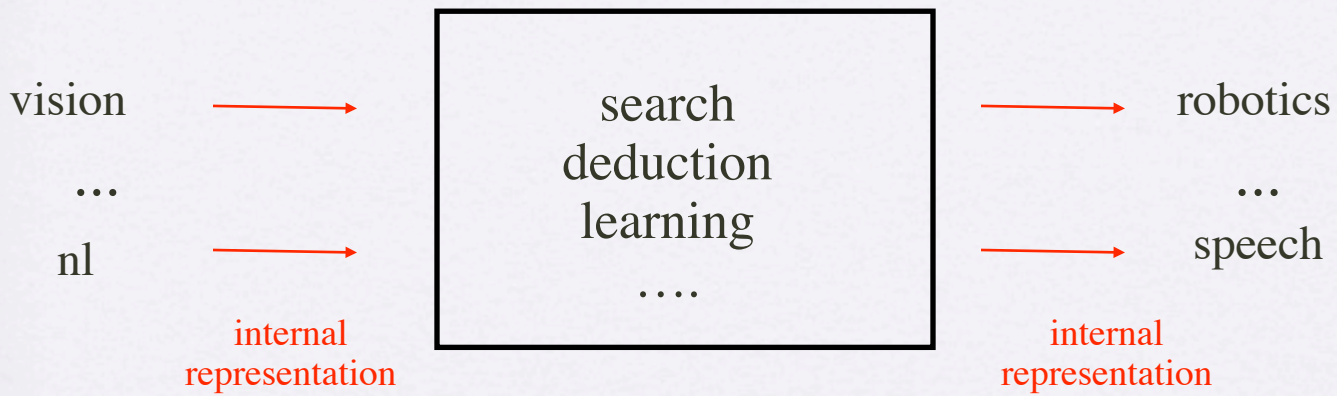
intelligence is (?) reasoning + knowledge

- reasoning
  - universal inference methods
  - "weak" methods, e.g. hill climbing
  - domain-independent search through symbolic state spaces
  - problem-solving and planning via theorem proving from first principles
- knowledge
  - universal methods --> combinatorial explosion
  - "strong" methods:
    - heuristics
    - domain-dependent knowledge
    - shallow deductions
  - ..... **expert systems**

16

goal of AI

build a person / animal



internal representation:

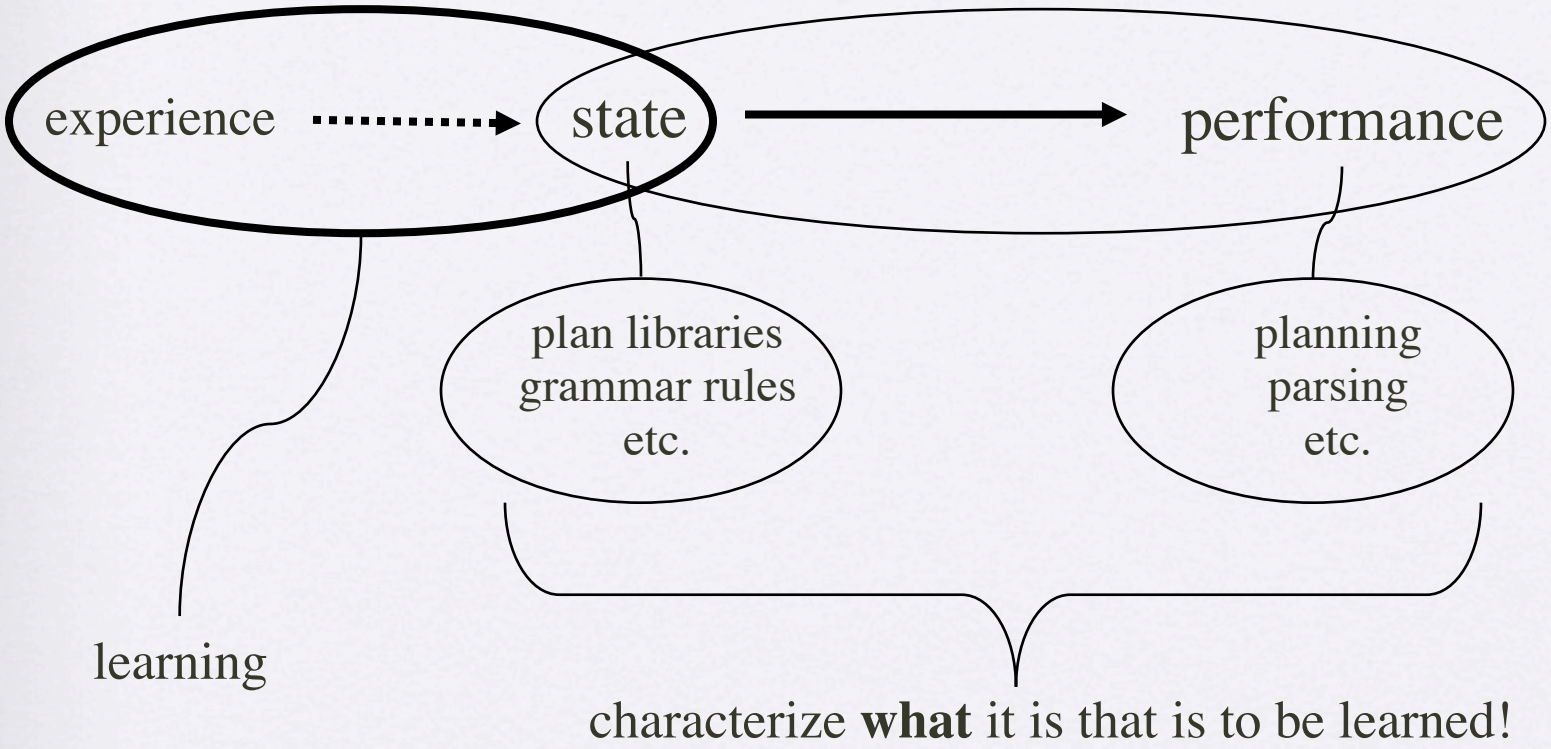
all representations inter-translatable;

not NL:

unambiguous, explicit referents, only gist remembered;

support inferences; ....

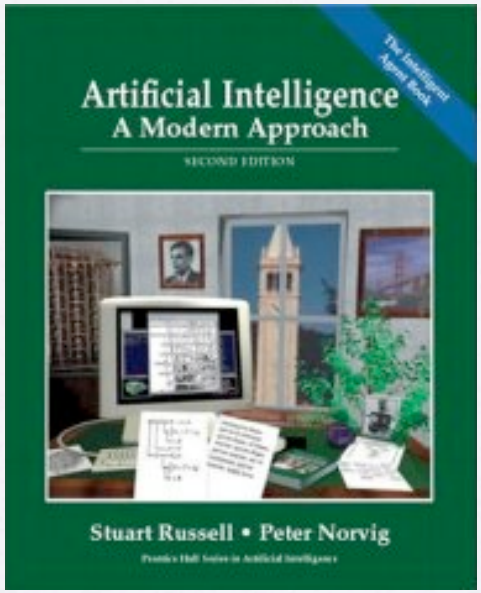
why is AI not just "learning"? why not just build a child?

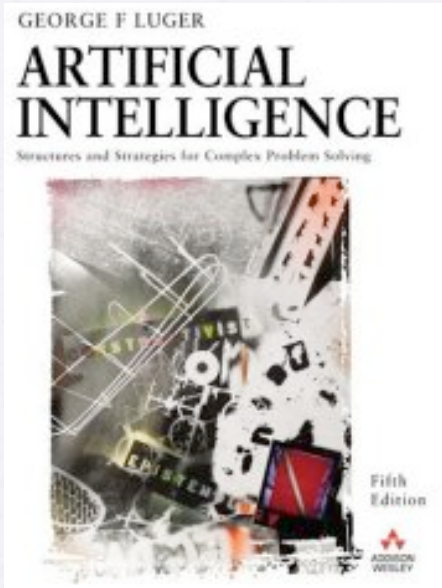


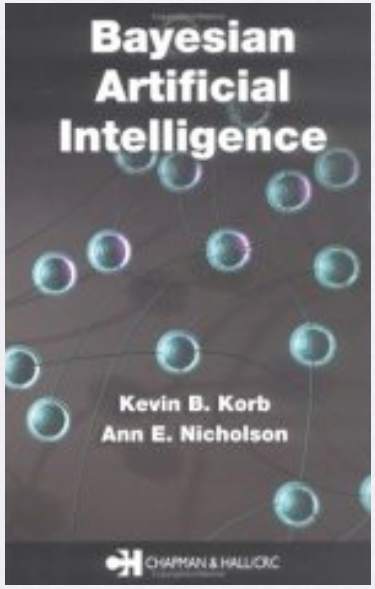
- to learn anything you should already 'know' a lot…
- without strong clues as to what is to be learned, nothing gets learned..
- there are many kinds of learning ...

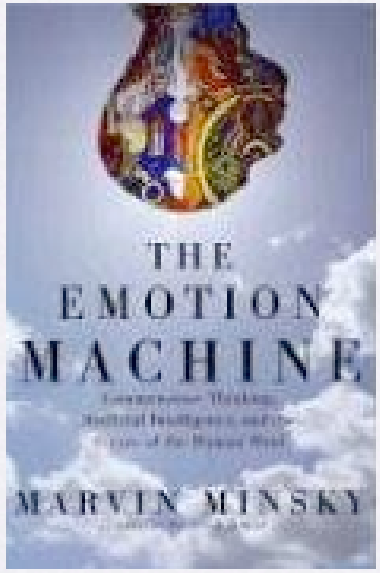
supp: NN starts talking… no better than yet another ill-understood person!!

some references

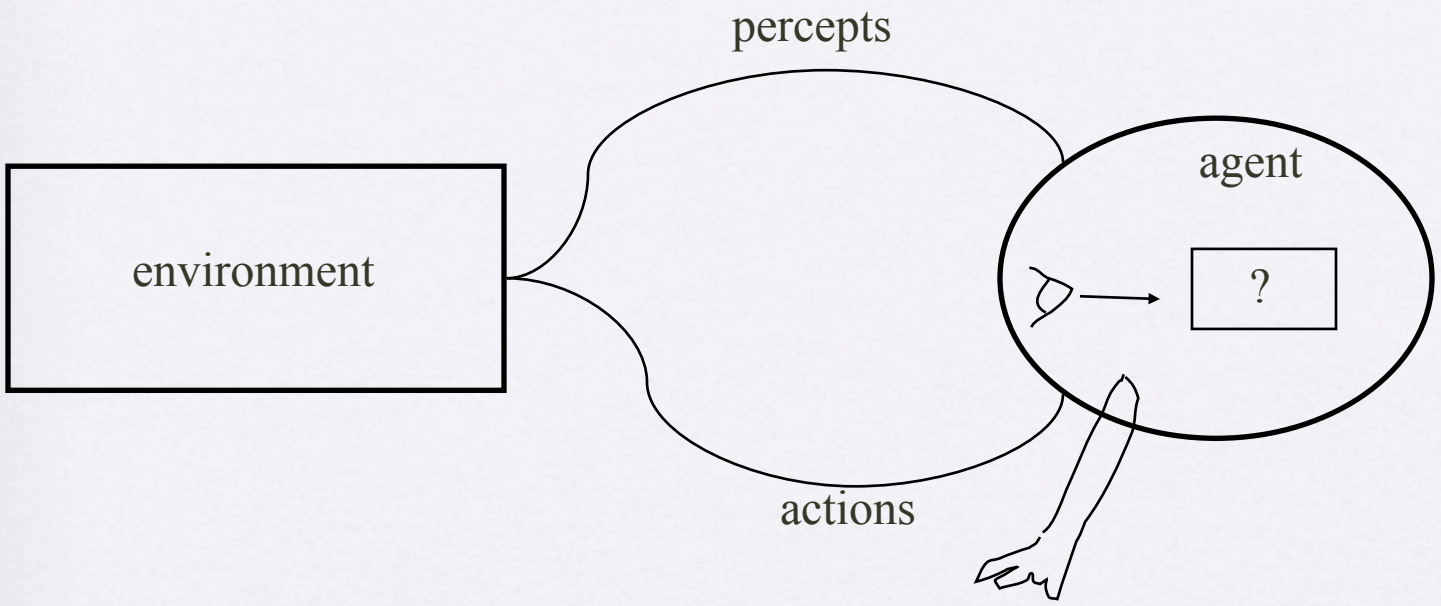








intelligent, autonomous agents?



agent: mapping: percept sequences ⇒ actions

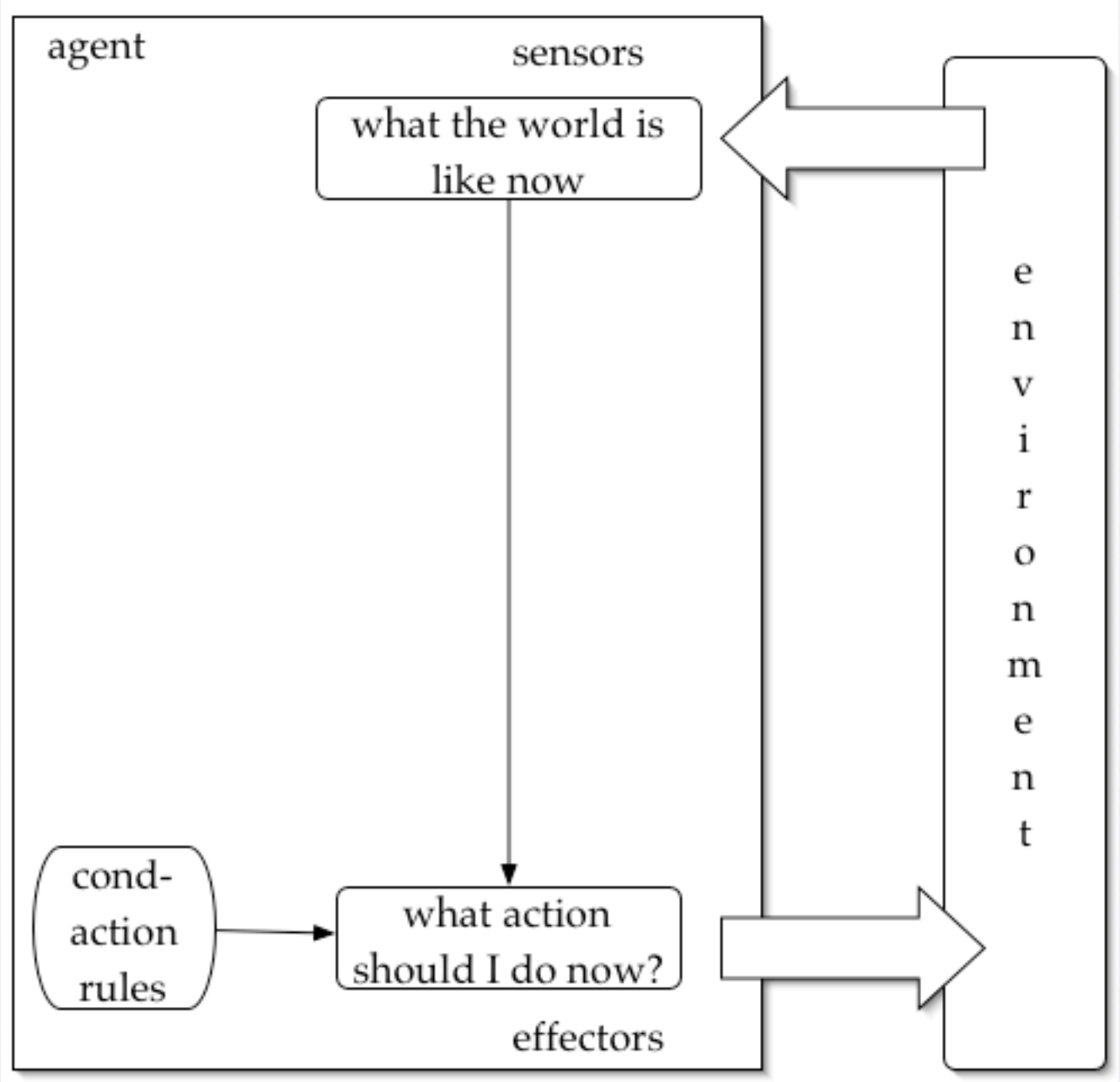
- performance measure
- percept sequence
- agent's knowledge about the environment
- agent's action repertoire

rational agent: acts so as to maximize expected performance measure, given percept sequence and knowledge.

basic agent

```
function Agent (percept) returns action
        static: memory
        memory ← UpdateMemory(memory, percept)
        ; agent builds up sequences of percepts in memory; only one
        input percept per invocation
        action ← ChooseBestAction(memory)
        memory ← UpdateMemory(memory, action)
        ; a performance measure is applied externally
return action
```

- reflex agents
- keeping track of the world agents
- goal-based agents
- utility-based agents
-
- ….


agent
sensors
what the world is like now
environment
cond-action rules
what action should I do now?
effectors

reflex agent

works only if a correct decision can be made on basis of current percept.

à la **expert system**
à la **subsumption architecture**

```
function Agent (percept) returns action
        static: rules
        state ← InterpretInput(percept) ; abstract description of world state from percept
        rule ← RuleMatch(state, rules); returns 1. rule matching state description
        action ← RuleAction(rule)
return action
```

reflex agent with (internally maintained) world state

needed when world is only partially observable

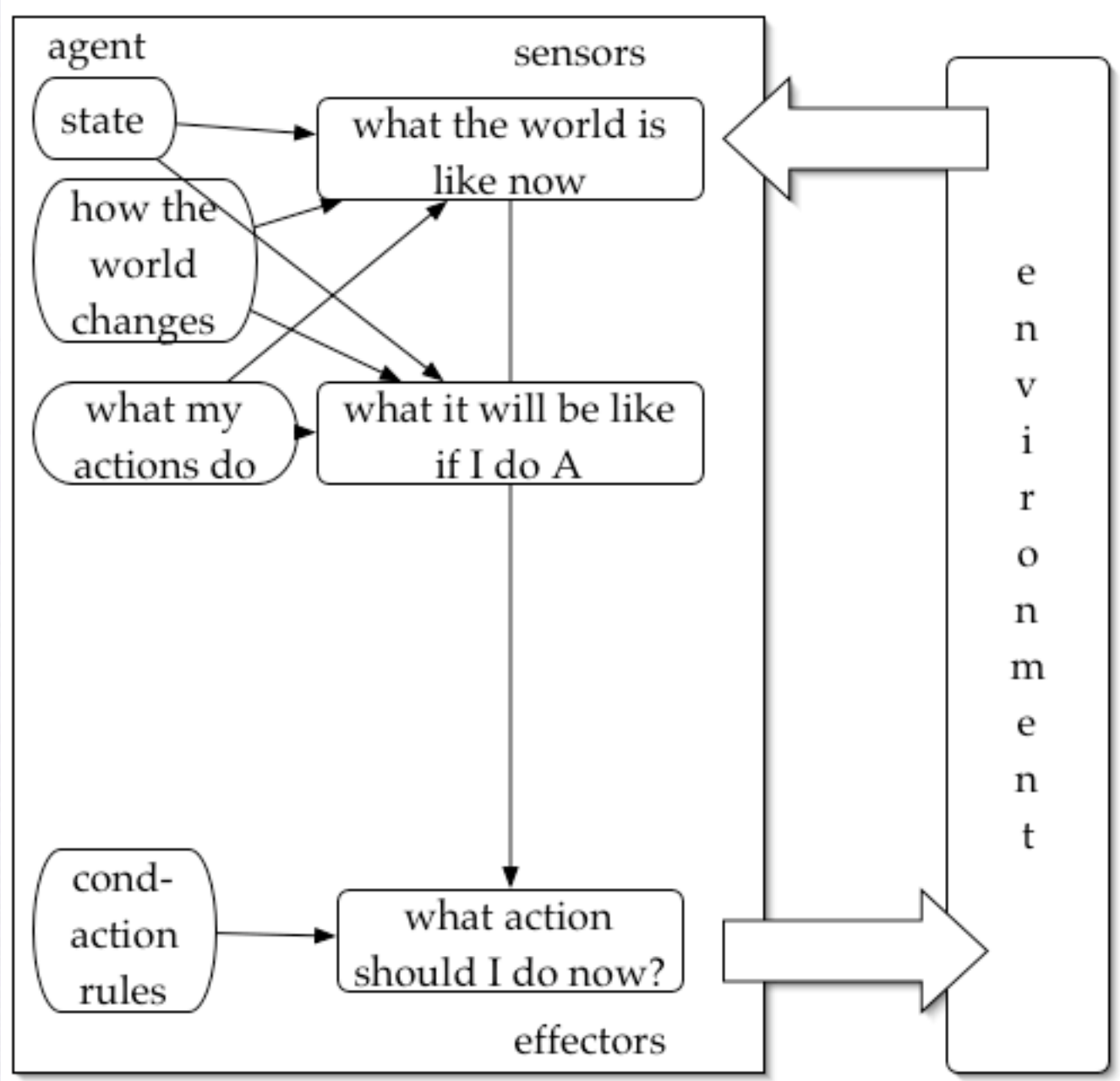
```
function Agent (percept) returns action
        static: rules
                state  ; world state
        state ← InterpretInput(percept) ; abstract
```
description of world state from percept

```
        state ← UpdateState(state, percept)
```
; **hard**! presupposes knowledge about 1) how world changes independently of agent and 2) how agent's actions affect the world!

```
        rule ← RuleMatch(state, rules); returns 1. rule
```
matching state description

```
        action ← RuleAction(rule)
        state ← UpdateState(state, action)
```
; **hard**! keep track of unsensed parts of the world and of effects of agent's actions!

```
return action
```


agent
sensors
state
what the world is like now
how the world changes
what my actions do
what it will be like if I do A
cond-action rules
what action should I do now?
effectors
environment

goal and utility-based agents

actions depend on current state and goal..

• often, goal satisfaction requires sequences of actions:
what **will** happen if I do this?

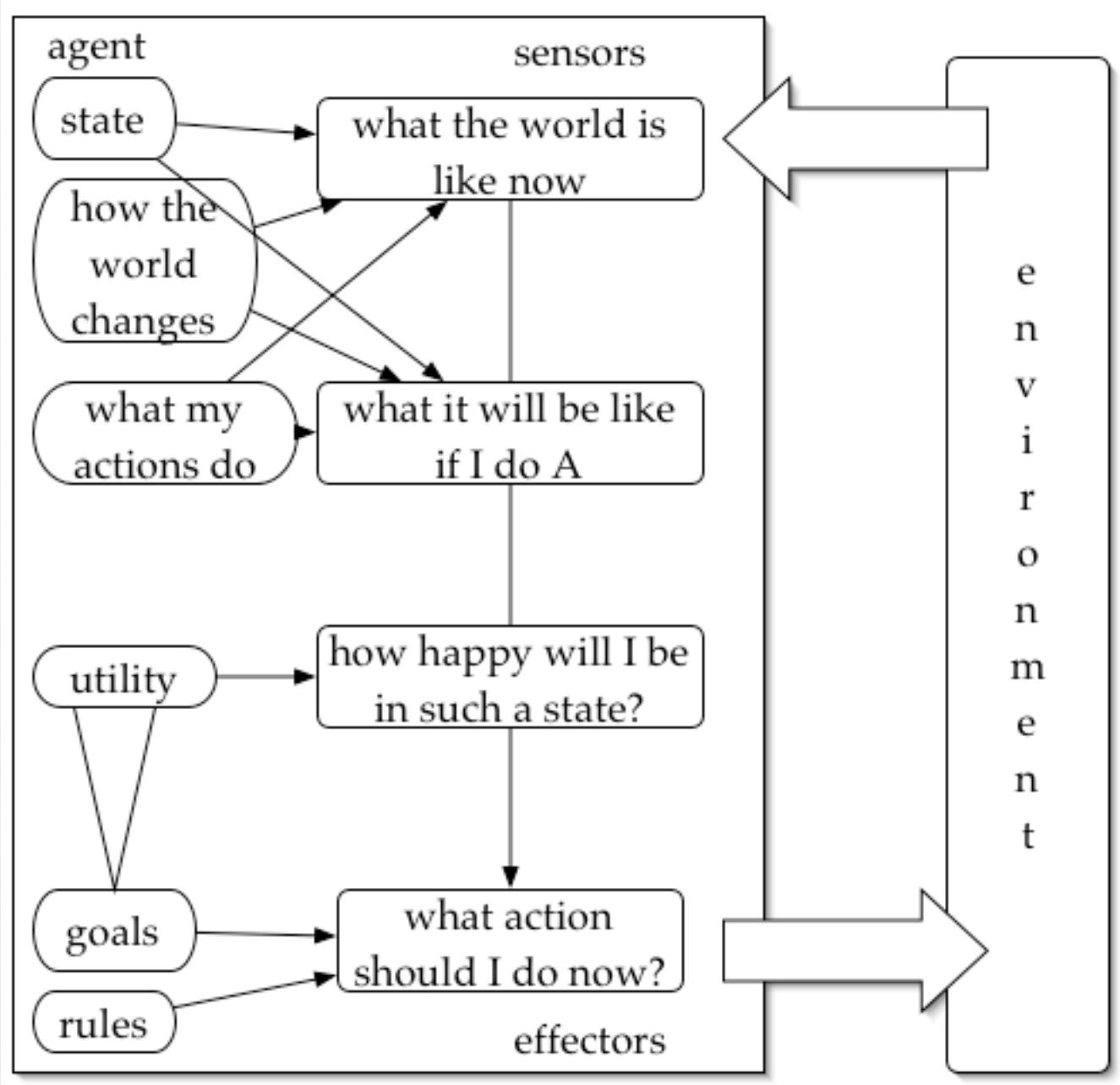
……. credit assignment

goals are not enough:

some goal-achieving sequences are cheaper, faster, etc. than others.

utility: states → reals, goal tradeoffs ….

26


agent
sensors
state
what the world is like now
how the world changes
what my actions do
what it will be like if I do A
how happy will I be in such a state?
utility
goals
what action should I do now?
rules
effectors
environment

evaluating agents....

```
function RunEvalEnvironment (state, UpdateFn, agents,
termination, PerformanceFn) returns scores
```

; state, UpdateFn simulate Environment; unseen by agents! The agent's states must be constructed from percepts alone. Agents have no access to PerformanceFn!

```
repeat
    for each agent in agents do
       Percept[agent] ← GetPercept(agent, state)

    for each agent in agents do
       Action[agent] ← Program[agent](Percept[agent])

    state ← UpdateFn(actions, agents, state)

    scores ← PerformanceFn(scores, agents, state)

until termination
return scores
```

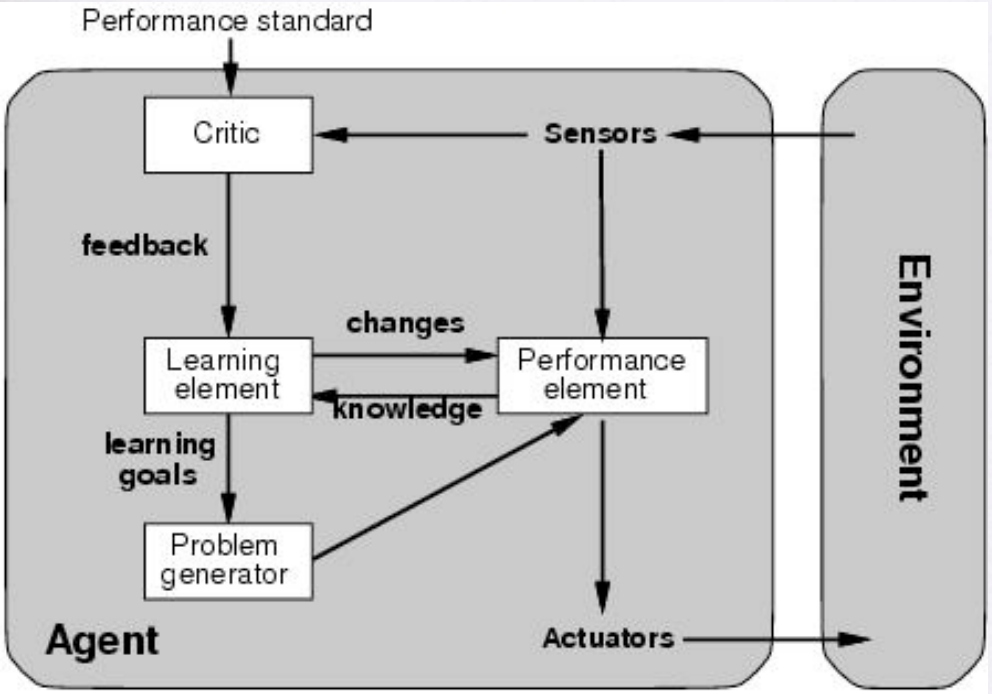
types of environments

- fully accessible to agent's senses (or not)
- deterministic (or not)
  - next state completely determined by current state and action
- episodic (or not)
  - quality of action depends only on current episodes, not on earlier actions
- static (or not)
  - environment does not change while the agent deliberates
- discrete (or not)
  - fixed number of well defined percepts and actions

---

e.g. chess: acc, det, ¬epis, static, discrete
medical diagnosis: ¬acc, ¬det, ¬epis, ¬static, ¬discrete

where do the actions selected by the agent programs come from?



Performance Element: agent program to select actions

Learning Element: improves PE and makes agent behavior robust in initially unknown environments

Problem Generator: suggests actions that may lead to new, informative experiences :

exploitation vs exploration