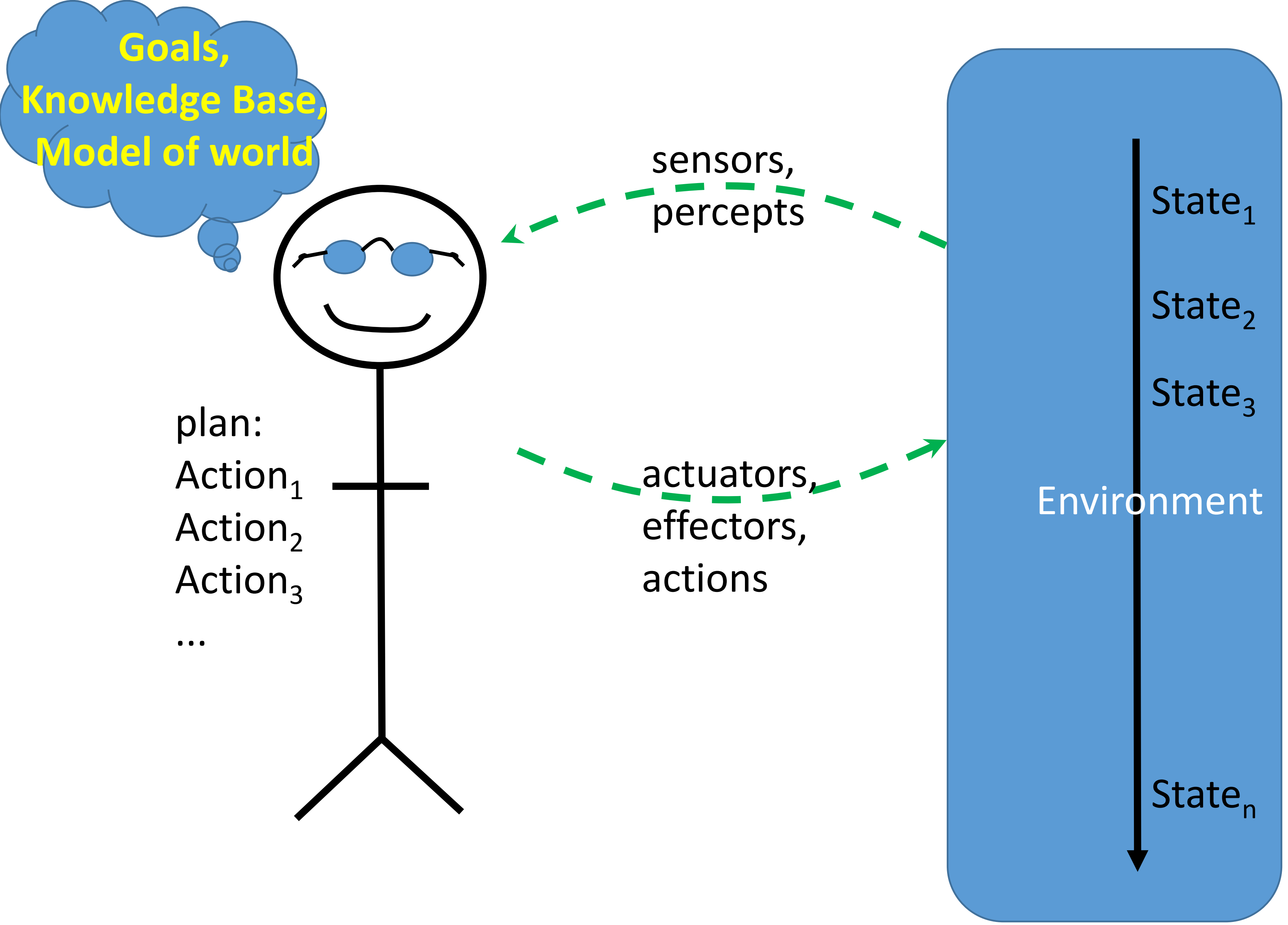
Intelligent Agents (Ch. 2)

- examples of agents
  - webbots, ticket purchasing, electronic assistant, *Siri*, news filtering, *autonomous vehicles*, printer/copier monitor, Robocup soccer, NPCs in Quake, Halo, Call of Duty...
- agents are a unifying theme for AI
  - use search and knowledge, planning, learning...
  - focus on <u>decision-making</u>
  - must deal with uncertainty, other actors in environment

Characteristics of Agents

- essential characteristics
  1. agents are situated:
     - can sense and manipulate an environment that changes over time
  2. agents are goal-oriented
  3. agents are autonomous
- other common (but not universal) aspects of agents:
  - adaptive (learns from experience)
  - optimizing (rational)
  - social (i.e. cooperative, teamwork, coordination)
  - life-like (e.g. in games, interactions with humans



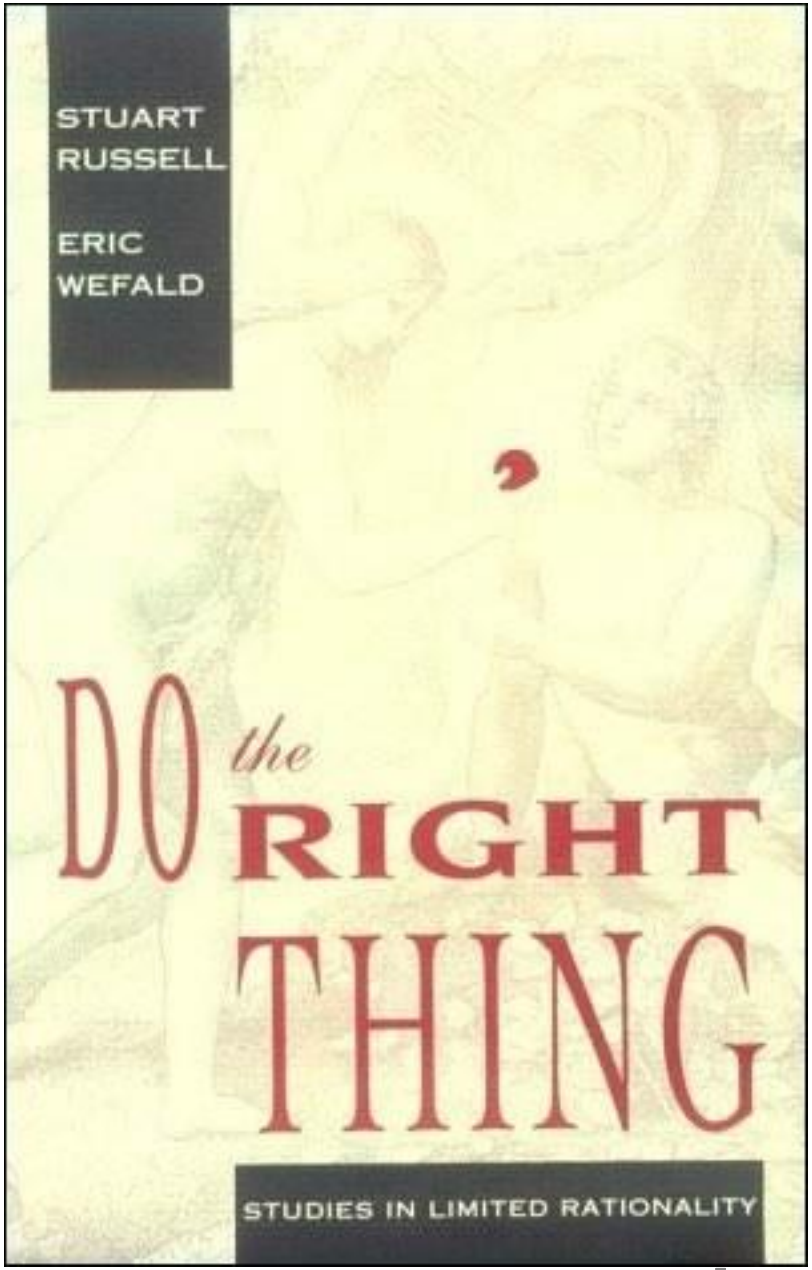
Goals,
Knowledge Base,
Model of world
sensors,
percepts
plan:
$Action_1$
$Action_2$
$Action_3$
...
actuators,
effectors,
actions
$State_1$
$State_2$
$State_3$
Environment
$State_n$

- policy - mapping of states (or histories) to actions
  - $\pi(s)=a$
  - $\pi(s_1,\ldots s_t)=a_t$

- Performance measures:
  - utility function, rewards, costs, goals
  - mapping of *states* (or *states*$\times$*actions*) into R,
    S |-> $\mathfrak{R}$ or S,A |-> $\mathfrak{R}$

Rational behavior (rationality)

- rationality: "for each possible percept sequence, a rational agent should *select an action that is expected to maximize its performance measure*, given the evidence provided by the percept sequence and whatever built-in knowledge the agent has"
- colloquially, being rational means "doing the right thing"



Rationality

- <u>*select an action that is expected to maximize its performance measure*</u>
- consider a set of possible outcomes, $\{o_i\}$
- select the action *i* that leads to the outcome with the highest payoff/reward, $\boxed{argmax_i\ payoff(o_i)}$
- in uncertain (stochastic) environments, if an action could lead to several outcome, <u>take the average outcome, weighted by probability</u>

remember Expectiminimax?

$$Expectiminimax(s) = \begin{cases} u_1(s) \text{ if is a terminal node} \\ \max\{Expectiminimax(s') \mid s' \in succ(s)\} \text{ if max node} \\ \min\{Expectiminimax(s') \mid s' \in succ(s)\} \text{ if min node} \\ \Sigma_{s' \in succ(s)} P(s') \cdot Expectiminimax(s') \text{ if } \textit{chance node} \end{cases}$$

take action that leads to highest average mm score over childern

Task Environments

- The architecture or design of an agent is strongly influenced by characteristics of the environment

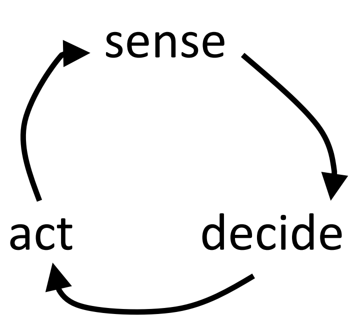
Discrete	Continuous
Static	Dynamic
Deterministic	Stochastic
Episodic	Sequential
Fully Observable	Partially Observable
Single-Agent	Multi-Agent

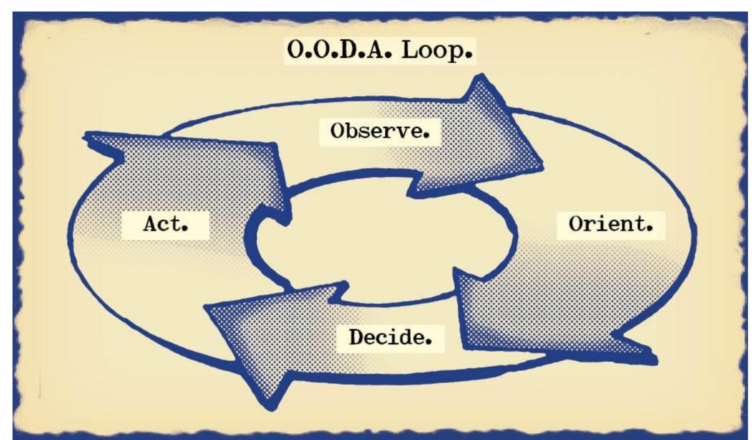
(read the definitions and examples in the textbook)

Agent Architectures

- Reactive/Reflex Agents
  - stimulus-response
  - condition-action lookup table
  - efficient
  - goals are implicit
  - sense-decide-act loop
  - OODA loop (observe-orient-decide-act)

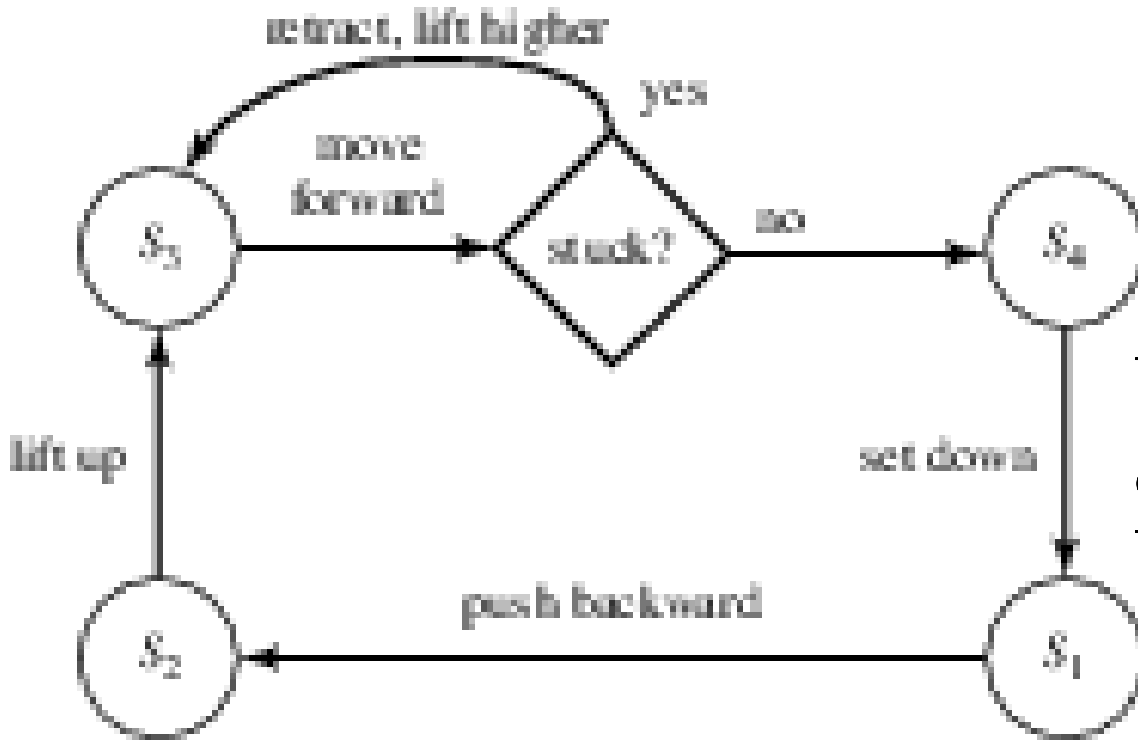
State	Action
S1	A1
S2	A2
S3	A3
...	...



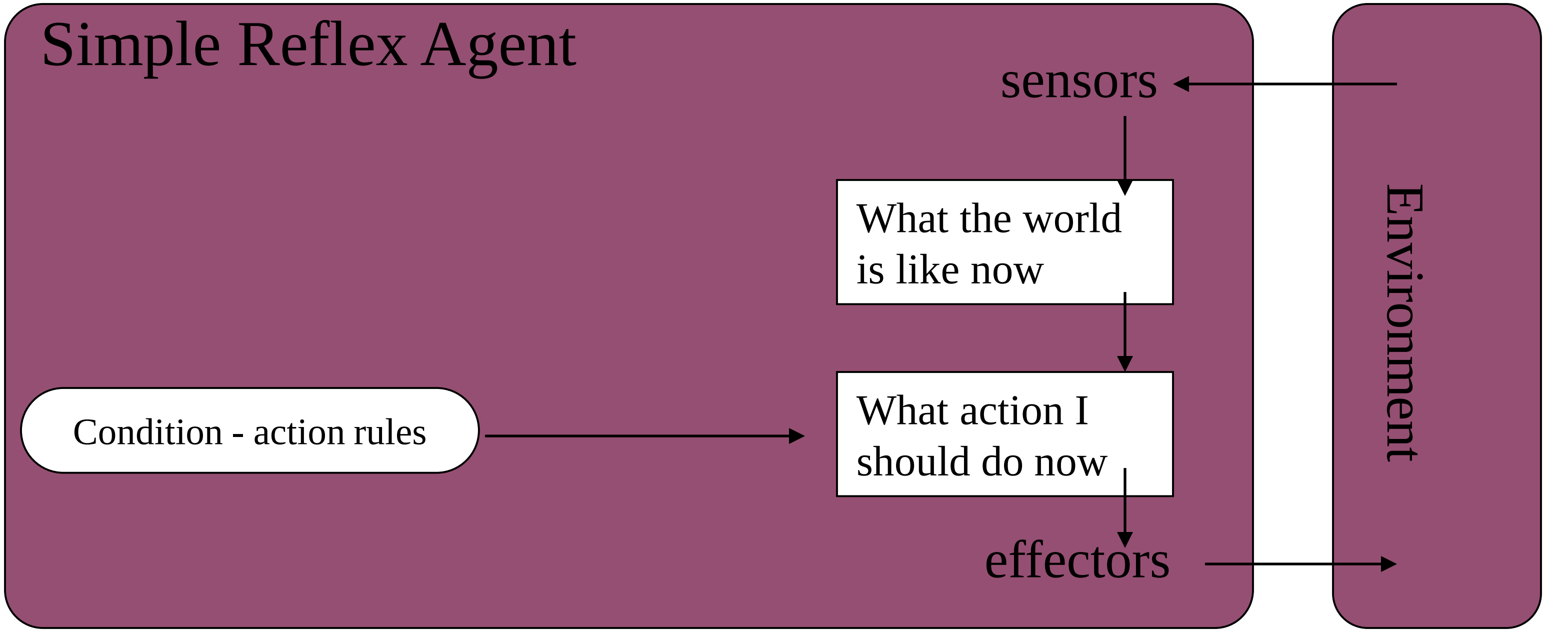


Ghengis
(Rodney Brooks, MIT)

simple
reactive
controllers



finite-state
machines are a
common alternative
to lookup tables



function SIMPLE-REFLEX-AGENT(*percept*) **returns** action
static: *rules*, a set of condition-action rules

state ← INTERPRET-INPUT (*percept*)
rule ← RULE-MATCH (*state*,*rules*)
action ← RULE-ACTION [*rule*]
return *action*

Stop after First match.
Rules should be prioritized.

A simple reflex agent works by finding a rule whose condition matches the current situation (as defined by the percept) and then doing the action associated with that rule.

(this is specifically a Rule-based Reflex Agent)

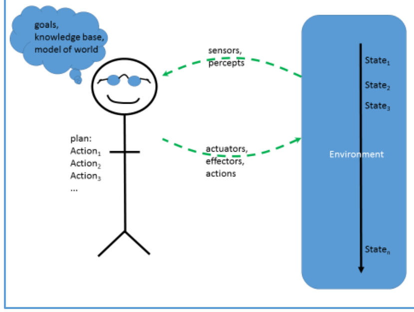
Agent Architectures

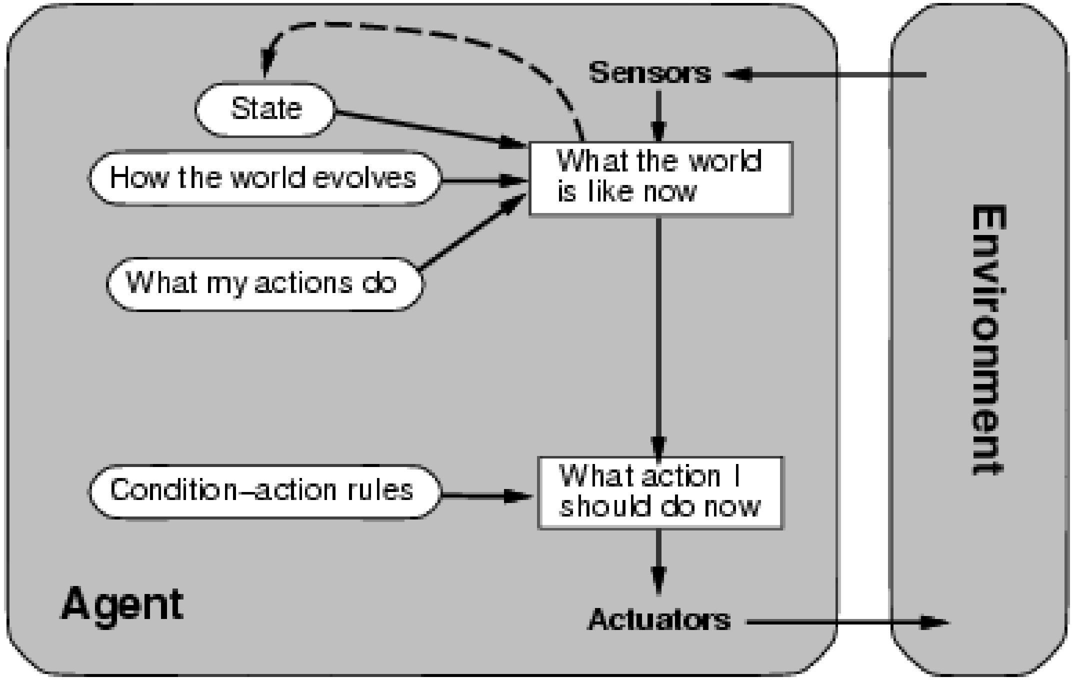
- Rule-based Reactive Agents
  - condition-action trigger rules
    - *if carInFrontIsBraking then InitiateBraking*
  - more compact than table
  - issue: how to choose which rule to fire?
    - must prioritize rules, if more than one rule can fire
- implementations
  - if-then-else cascades
  - CLIPS; JESS - Java Expert System
  - Subsumption Architecture (Rodney Brooks, MIT)
    - hierarchical - design <u>behaviors</u> in layers
    - e.g. obstacle avoidance overrides moving toward goal

Agent Architectures

- Model-based Agents
  - use local variables to represent and remember the state of the world and infer unobservable aspects

Model-based agent





function MODEL-BASED-REFLEX-AGENT (*percept*) **returns** action
static: *state*, a description of the current world state
rules, a set of condition-action rules

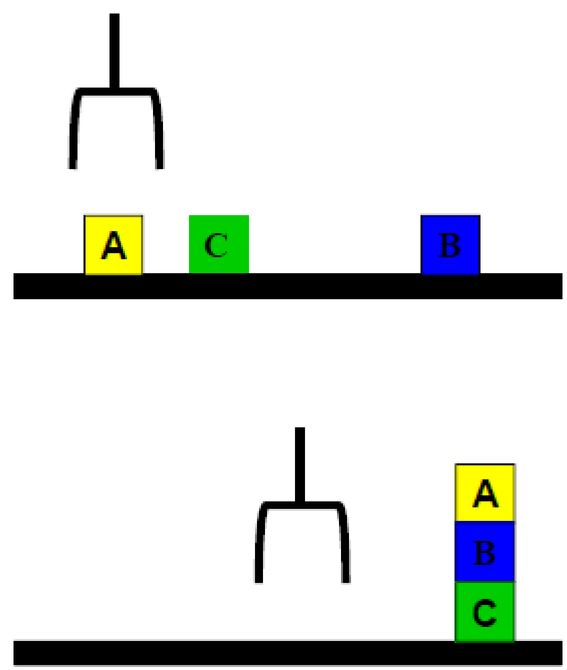
state ← UPDATE-STATE (*state*, *percept*)
rule ← RULE-MATCH (*state*, *rules*)
action ← RULE-ACTION [*rule*]
state ← <u>UPDATE-STATE</u> (*state*, *action*) // predict, remember
return *action*

Agent Architectures

- Knowledge-based Agents
  - knowledge base containing logical rules for:
    - inferring unobservable aspects of state
    - inferring effects of actions
    - inferring what is likely to happen
  - Proactive agents - reason about what is going to happen
  - use inference algorithm to decide what to do next, given state and goals
    - use forward/backward chaining, natural deduction, resolution...
    - prove: $\text{Percepts} \cup \text{KB} \cup \text{Goals} \models do(\alpha_i)$ for some action α_i (remember SatPlan?)

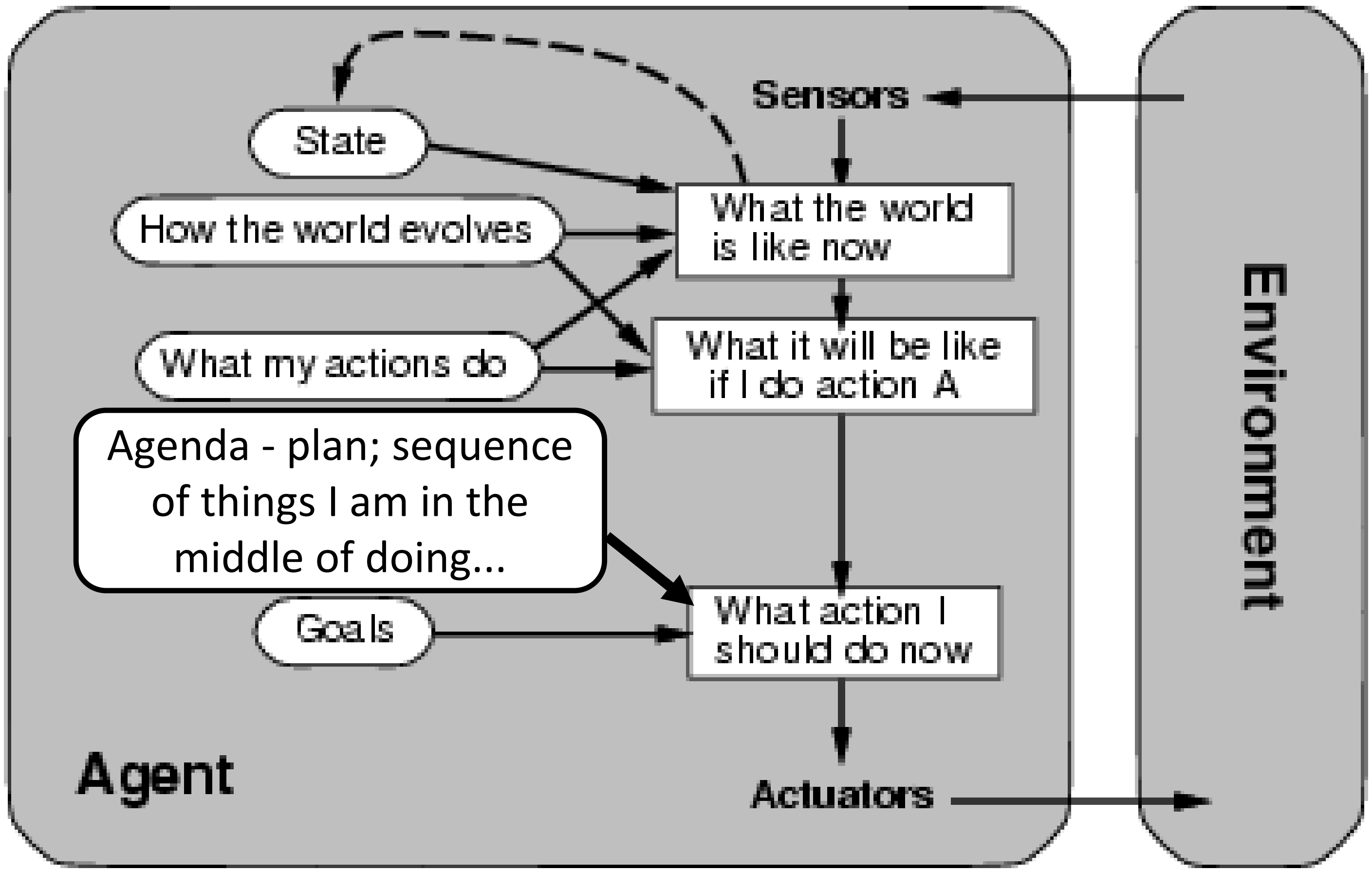
Agent Architectures

- Goal-based Agents (Planners)
  - search for plan (sequence of actions) that will transform S_{init} into S_{goal}
- state-space search (forward from S_{init}, e.g. using A*)
- goal-regression (backward from S_{goal})
  - reason about effects of actions
- SATplan, GraphPlan, PartialOrderPlan...



A plan:
pickup(b)
stack(b,c)
pickup(a)
stack(a,b)

Goal-based agents



note: plans must be maintained on an <u>agenda</u> and carried out over time - these are <u>intentions</u>

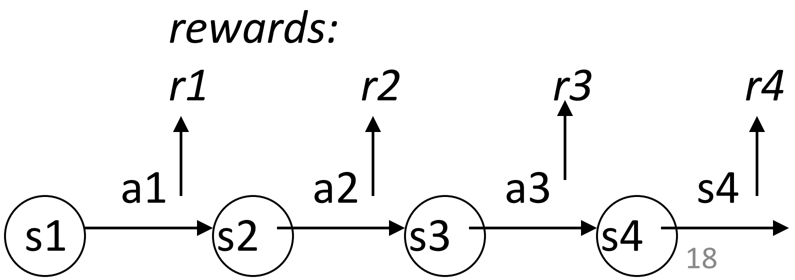
Agent Architectures

- Utility-based Agents
  - utility function: maps states to real values, quantifies "goodness" of states, u(s)$\rightarrow\Re$
  - agents select actions to maximize utility
    - sometimes payoffs are immediate (think "reactive")
    - othertimes payoffs are delayed:
      - <u>Sequential Decision Problems</u>
      - maximize long-term reward

Markov Decision Problems (MDPs)

- transition function: T(s,a)$\rightarrow$S
  - outcomes of actions
  - could be probabilistic (distribution over successors states)
- reward/cost function: R(s,a)$\rightarrow\Re$
- "plans" are encoded in *policies*
  - mappings from states to actions: π:S$\rightarrow$A
  - Markov property: probabilities only depend on current state
- the goal: maximize reward over time
  - long-term discounted reward

$$\sum_{t=0}^{\infty} \gamma^t R_{a_t}(s_t, s_{t+1})$$



Multi-Agent Systems

- Collaborative Agents
  - competition (Minimax) vs. collaboration
  - collaboration: is there a way agents can work together so they mutually benefit?
  - "open" agent environment: assume all agents are self-interested (have their own utility function)

Market-based methods for Multi-Agent Systems

- mechanisms to incentivize collaboration
  - contract networks - agents make bids to do tasks for each other, negotiate price, make commitments
  - auctions - agents bid on resources
    - first-price, second-price, open vs sealed bid, asc vs descending
    - strategy to maximize utility?
    - bidding on *combinations* of resources is more complicated
  - consensus algorithms - voting (weight choices by utility)
  - do these mechanisms incentivize agents to be rational and bid their true values; free of exploits and manipulation?
  - efficiency: do these mechanisms maximize social benefit? (sum of utility of outcomes over all agents)

Methods for Collaborative Agents

- Agent Teamwork
  - shared goals, joint intentions
    - assume teammates are not just self-interested
    - teammates can compensate for each other if a team goal is at risk
  - well-defined roles, responsibilities
  - communication among teammates is key
- BDI - *modal logic* for representing **B**eliefs, **D**esires (goals), and **I**ntentions (actions) of other agents
  - **Bel**(self,empty(ammo))
    ∧**Bel**(teammate,¬empty(ammo))
    ∧**Goal**(teammate,shoot(gun))
    → **Tell**(teammate,empty(ammo))
  - *intentions* are actions that we select and commit to, which means we plan to do them (or keep trying till we succeed)
  - model operators go beyond FOL: Bel(<agt>,<sentence>)