3: AGENT ARCHITECTURE

AI6125: Intelligent Agents

Assoc Prof Zhang Jie

Based on "An Introduction to MultiAgent Systems" by Michael Wooldridge, John Wiley & Sons, 2002/2009

Agent Architecture

- An *agent architecture* is a *software design* for an agent
- We have already seen a top-level decomposition (abstract), into:

  perception – state – decision – action
- An agent architecture defines (concrete):
  - Key data structures;
  - Operations on data structures;
  - Control flow between operations

History of Agent Architectures

- Originally (1956-1985), pretty much all agents designed within AI were ***symbolic reasoning* agents**
- Its purest expression proposes that agents use *explicit logical reasoning* in order to decide what to do
- Problems with symbolic reasoning led to a reaction against this — the so-called ***reactive agents*** movement, 1985–present

  no long-term planning.
- From 1990-present, a number of alternatives proposed: ***hybrid* architectures**, which attempt to combine the best of reasoning and reactive architectures

Symbolic Reasoning Agents

- The classical approach to building agents is to view them as a particular type of knowledge-based system, and bring all the associated (discredited?!) methodologies of such systems to bear
- This paradigm is known as *symbolic AI*
- We define a ***deliberative*** agent or agent architecture to be one that:
  - contains an explicitly represented, symbolic model of the world
  - makes decisions (for example about what actions to perform) via symbolic reasoning

Symbolic Reasoning Agents

- If we aim to build an agent in this way, there are two key problems to be solved:

1. *The transduction problem*:
   that of translating the real world into an accurate, adequate symbolic description, in time for that description to be useful…vision, speech understanding, learning
2. *The representation/reasoning problem*:
   that of how to symbolically represent information about complex real-world entities and processes, and how to get agents to reason with this information in time for the results to be useful…knowledge representation, automated reasoning, automatic planning

Symbolic Reasoning Agents

- Most researchers accept that neither problem is anywhere near solved
- Underlying problem lies with the complexity of symbol manipulation algorithms in general: many (most) search-based symbol manipulation algorithms of interest are *highly intractable*
- Because of these problems, some researchers have looked to alternative techniques for building agents; we look at these later

Deductive Reasoning Agents

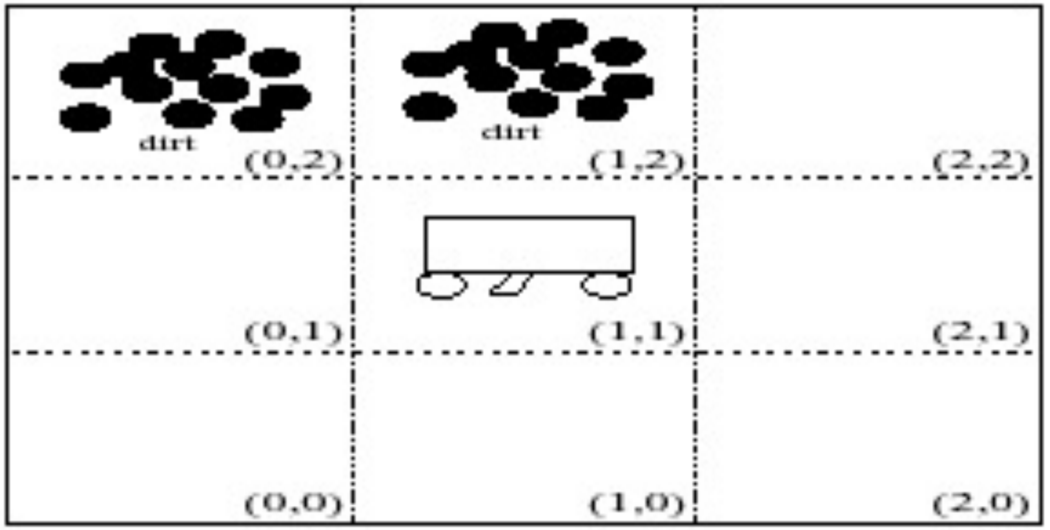
- How can an agent decide what to do using theorem proving?
- Basic idea is to use logic to encode a theory stating the *best* action to perform in any given situation
- Let:
  - ρ be this theory (typically a set of rules) facts
  - Δ be a logical database that describes the current state of the world
  - Ac be the set of actions the agent can perform
  - $\Delta \vdash_{\rho} \phi$ mean that ϕ can be proved from Δ using ρ

Deductive Reasoning Agents

/* *try to find an action explicitly prescribed* */
for each $a \in Ac$ do
    if $\Delta \vdash_\rho Do(a)$ then
        return a
    end-if
end-for
/* *try to find an action not excluded* */
for each $a \in Ac$ do
    if $\Delta \nvdash_\rho \neg Do(a)$ then
        return a
    end-if
end-for
return $null$ /* *no action found* */

Deductive Reasoning Agents

- An example: The Vacuum World
- Goal is for the robot to clear up all dirt



Deductive Reasoning Agents

- Use 3 *domain predicates* to solve problem:

$In(x, y)$	agent is at (x, y)
$Dirt(x, y)$	there is dirt at (x, y)
$Facing(d)$	the agent is facing direction d

- Possible actions:

$$Ac = \{turn, forward, suck\}$$

P.S. *turn* means "turn right"

Deductive Reasoning Agents

- Rules ρ for determining what to do:

$$
\begin{array}{rcl}
In(0,0) \wedge Facing(north) \wedge \neg Dirt(0,0) & \longrightarrow & Do(forward) \\
In(0,1) \wedge Facing(north) \wedge \neg Dirt(0,1) & \longrightarrow & Do(forward) \\
In(0,2) \wedge Facing(north) \wedge \neg Dirt(0,2) & \longrightarrow & Do(turn) \\
In(0,2) \wedge Facing(east) & \longrightarrow & Do(forward)
\end{array}
$$

- …and so on!
- Using these rules (+ other obvious ones), starting at $(0, 0)$ the robot will clear up dirt

Deductive Reasoning Agents

- Problems:
  - How to convert video camera input to $Dirt(0, 1)$?
  - decision making assumes a *static* environment: *calculative* rationality
  - decision making using first-order logic is *undecidable*!
- Typical solutions:
  - weaken the logic
  - use symbolic, non-logical representations
  - shift the emphasis of reasoning from *run time* to *design time*

!

NP-hard.

AGENT0 and PLACA

- Much of the interest in agents from the AI community has arisen from Shoham's notion of *agent oriented programming* (AOP)
- AOP: 'a new programming paradigm, based on a societal view of computation'
- The key idea that informs AOP is that of directly programming agents in terms of intentional notions like belief, commitment, and intention

AGENT0

- AGENT0 is implemented as an extension to LISP
- Each agent in AGENT0 has 4 components:
  - a set of capabilities (things the agent can do)
  - a set of initial beliefs
  - a set of initial commitments (things the agent will do)
  - a set of *commitment rules* 类似于theroy.
- The key component, which determines how the agent acts, is the commitment rule set

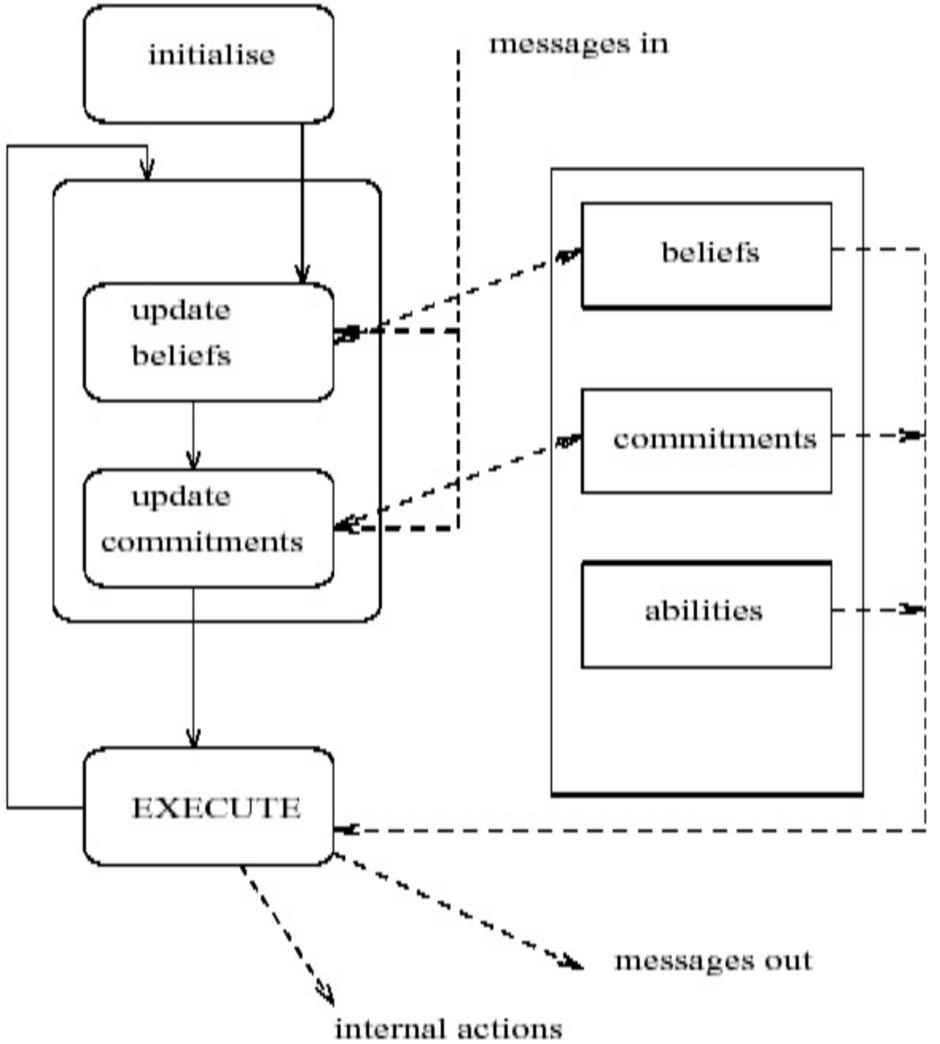
AGENT0

- Each commitment rule contains
  - a *message condition*
  - a *mental condition*
  - an action
- On each 'agent cycle'…
  - The message condition is matched against the messages the agent has received
  - The mental condition is matched against the beliefs of the agent
  - If the rule fires, then the agent becomes committed to the action (the action gets added to the agent's commitment set)

AGENT0

- Actions may be
  - *private*:
    an internally executed computation, or
  - *communicative*:
    sending messages
- Messages are constrained to be one of three types:
  - "requests" to commit to action
  - "unrequests" to refrain from actions
  - "informs" which pass on information

AGENT0


initialise
messages in
beliefs
update
beliefs
commitments
update
commitments
abilities
EXECUTE
messages out
internal actions

AGENT0

- A commitment rule:

```
COMMIT(
    ( agent, REQUEST, DO(time, action)
    ), ;;; msg condition
    ( B,
          [now, Friend agent] AND
          CAN(self, action) AND
          NOT [time, CMT(self, anyaction)]
    ), ;;; mental condition
    self,
    DO(time, action)
)
```

1

2

3

AGENT0

- This rule may be paraphrased as follows:
  if I receive a message from *agent* which requests me to do *action* at *time*, and I believe that:
  - *agent* is currently a friend
  - I can do the action
  - At *time*, I am not committed to doing any other action

  then commit to doing *action* at *time*

AGENT0 and PLACA

- AGENT0 provides support for multiple agents to cooperate and communicate, and provides basic provision for debugging…
- …it is, however, a *prototype*, that was designed to illustrate some principles, rather than be a production language
- A more refined implementation was developed by Thomas, for her 1993 doctoral thesis
- Her Planning Communicating Agents (PLACA) language was intended to address one severe drawback to AGENT0: the inability of agents to plan, and communicate requests for action via high-level goals
- Agents in PLACA are programmed in much the same way as in AGENT0, in terms of *mental change* rules

AGENT0 and PLACA

- An example mental change rule:

```
(((self ?agent REQUEST (?t (xeroxed ?x)))
  (AND (CAN-ACHIEVE (?t xeroxed ?x)))
     (NOT (BEL (*now* shelving)))
     (NOT (BEL (*now* (vip ?agent))))
  ((ADOPT (INTEND (5pm (xeroxed ?x)))))
  ((?agent self INFORM
     (*now* (INTEND (5pm (xeroxed ?x)))))))
```

- Paraphrased:
  if someone asks you to xerox something, and you can, and you don't believe that they're a VIP, or that you're supposed to be shelving books, then
  - adopt the intention to xerox it by 5pm, and
  - inform them of your newly adopted intention

Practical Reasoning

- Practical reasoning is reasoning directed towards actions — the process of figuring out what to do:
    - “Practical reasoning is a matter of weighing conflicting considerations for and against competing options, where the relevant considerations are provided by what the agent desires/values/cares about and what the agent believes.” (Bratman)
- Practical reasoning is distinguished from *theoretical reasoning* – theoretical reasoning is directed towards beliefs

Practical Reasoning

- Human practical reasoning consists of two activities:
  1 - *deliberation*
      deciding *what* state of affairs we want to achieve
  2 - *means-ends reasoning*
      deciding *how* to achieve these states of affairs
- The outputs of deliberation are *intentions*

Intentions in Practical Reasoning

- Notice that intentions are much stronger than mere desires:

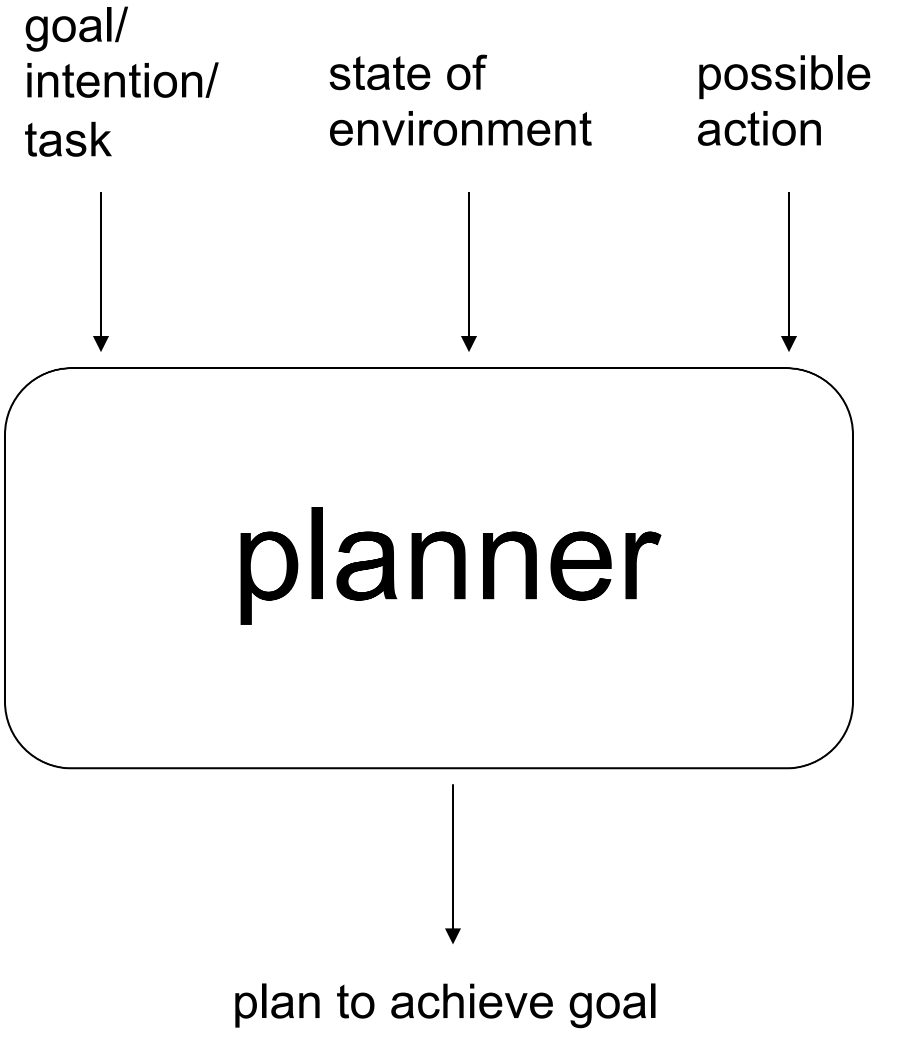
  "My desire to play basketball this afternoon is merely a potential influencer of my conduct this afternoon. It must vie with my other relevant desires [. . .] before it is settled what I will do. In contrast, once I intend to play basketball this afternoon, the matter is settled: I normally need not continue to weigh the pros and cons. When the afternoon arrives, I will normally just proceed to execute my intentions." (Bratman, 1990)

What is Means-End Reasoning?

- Basic idea is to give an agent:
  - representation of goal/intention to achieve
  - representation of actions it can perform
  - representation of the environment

  and have it generate a *plan* to achieve the goal
- Essentially, this is

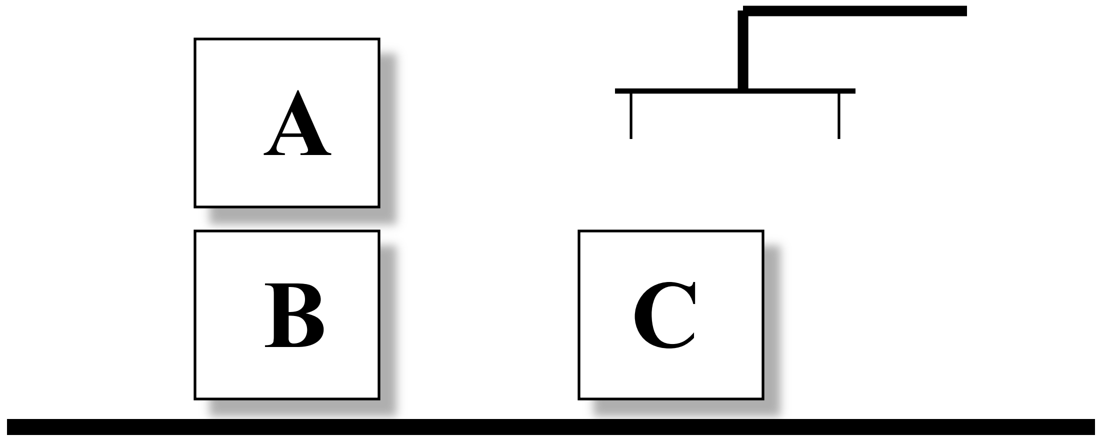
  automatic programming


goal/
intention/
task
state of
environment
possible
action
planner
plan to achieve goal

Planning

- Question: How do we *represent*. . .
  - goal to be achieved
  - state of environment
  - actions available to agent
  - plan itself

The Blocks World



- We'll illustrate the techniques with reference to the *blocks world* (like previous module)
- Contains a robot arm, 3 blocks (A, B, and C) of equal size, and a table-top

The Blocks World Ontology

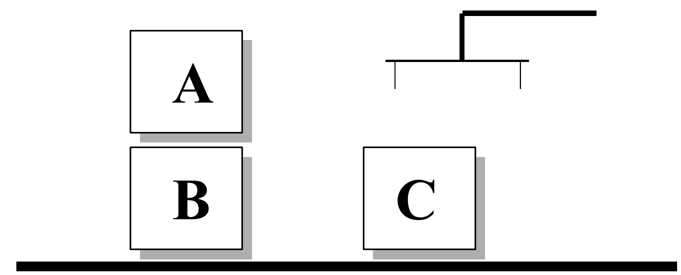
- To represent this environment, need an *ontology*

$On(x, y)$	obj x on top of obj y
$OnTable(x)$	obj x is on the table
$Clear(x)$	nothing is on top of obj x
$Holding(x)$	arm is holding x

The Blocks World

- Here is a representation of the blocks world described above:

  Clear(A)
  On(A, B)
  OnTable(B)
  OnTable(C)

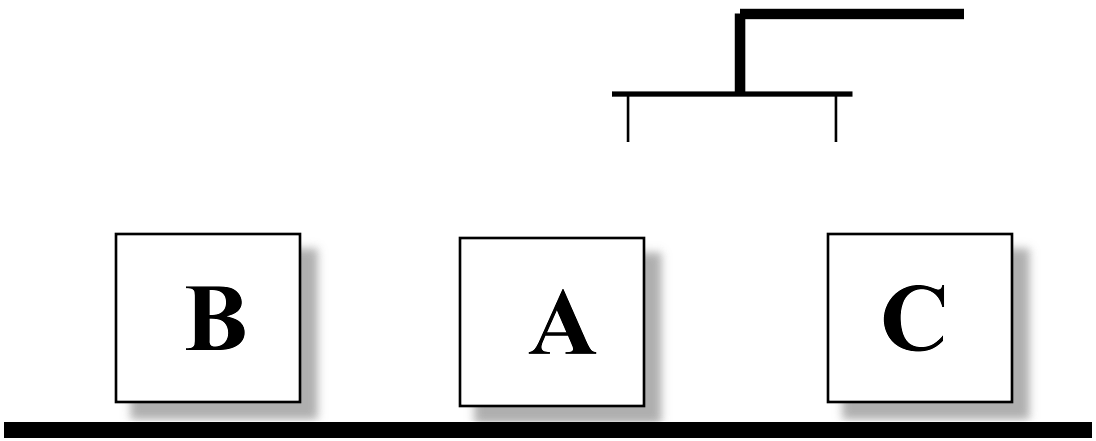


- Use the *closed world assumption*: anything not stated is assumed to be *false*

The Blocks World

- A *goal* is represented as a set of formulae
- Here is a goal:

$$OnTable(A) \land OnTable(B) \land OnTable(C)$$

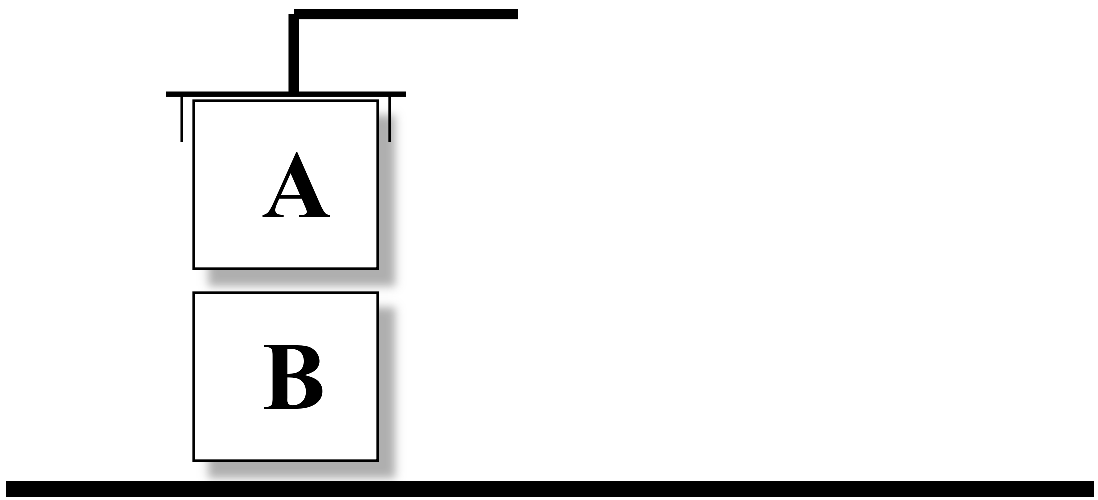


The Blocks World

- *Actions* are represented using a technique that was developed in the STRIPS planner
- Each action has:
  - a *name*
    which may have arguments
  - a *pre-condition list*
    list of facts which must be true for action to be executed
  - a *delete list*
    list of facts that are no longer true after action is performed
  - an *add list*
    list of facts made true by executing the action

Each of these may contain *variables*

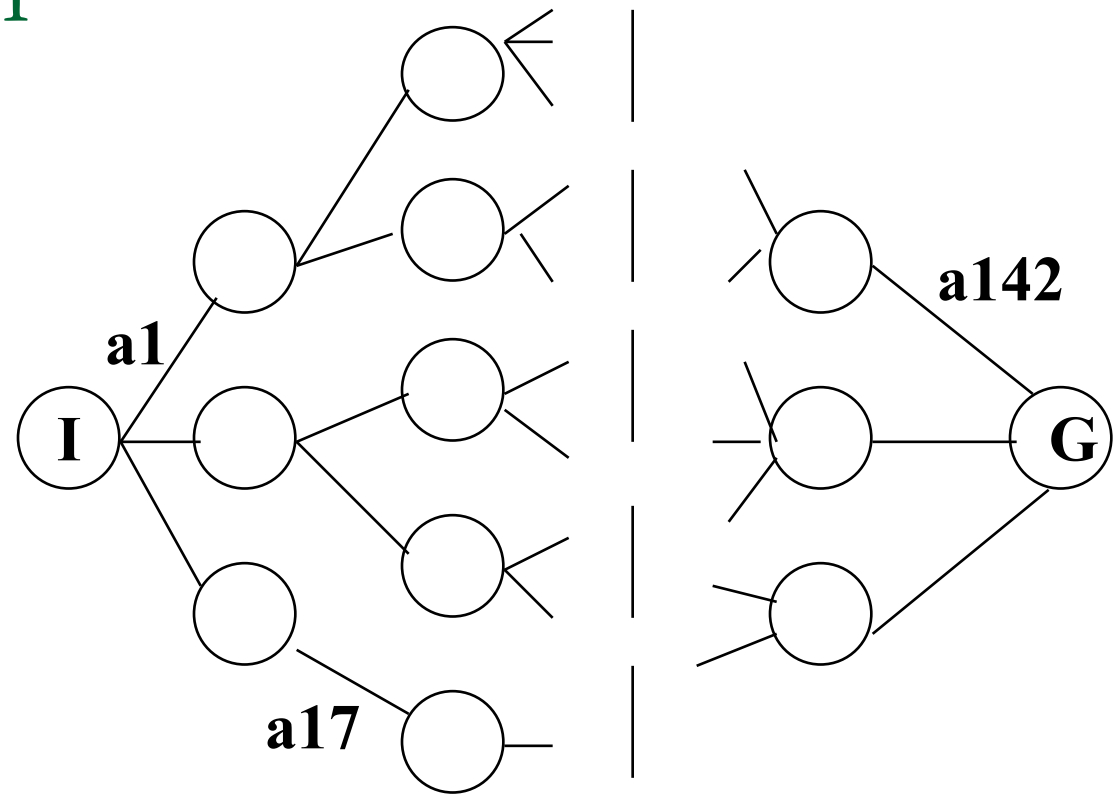
The Blocks World Operators



- Example:
  The *stack* action occurs when the robot arm places the object x it is holding is placed on top of object y.

	$Stack(x, y)$
pre	$Clear(y) \wedge Holding(x)$
del	$Clear(y) \wedge Holding(x)$
add	$ArmEmpty \wedge On(x, y)$

A Plan



- What is a plan?
  A sequence (list) of actions, with variables replaced by constants.

Implementing Practical Reasoning Agents

- A first pass at an implementation of a practical reasoning agent:

```
Agent Control Loop Version 1
1. while true
2.      observe the world;
3.      update internal world model;
4.      deliberate about what intention to achieve next;
5.      use means-ends reasoning to get a plan for the intention;
6.      execute the plan
7. end while
```

- (We will not be concerned with stages (2) or (3))

Implementing Practical Reasoning Agents

- Let’s make the algorithm more formal:

Agent Control Loop Version 2

$B := B_0$; /* initial beliefs */
while true do
    get next percept ρ;
    $B := brf(B, \rho)$;
    $I := deliberate(B)$;
    $\pi := plan(B, I)$;
    $execute(\pi)$
end while

Deliberation

- How does an agent deliberate?
  - begin by trying to understand what the *options* available to you are
  - *choose between them*, and *commit* to some
- Chosen options are then intentions

Deliberation

- The *deliberate* function can be decomposed into two distinct functional components:
  1 - *option generation*
    in which the agent generates a set of possible alternatives;
    Represent option generation via a function, *options*, which takes the agent's current beliefs and current intentions, and from them determines a set of options (= *desires*)
  2 - *filtering*
    in which the agent chooses between competing alternatives, and commits to achieving them.
    In order to select between competing options, an agent uses a *filter* function.

Deliberation

BDI agents

Agent Control Loop Version 3

$B := B_0$; belief

$I := I_0$; intention

while true do

    get next percept ρ;

    $B := brf(B, \rho)$;

    $D := options(B, I)$;

    $I := filter(B, D, I)$;

    $\pi := plan(B, I)$;

    $execute(\pi)$

end while

Reactive Architectures

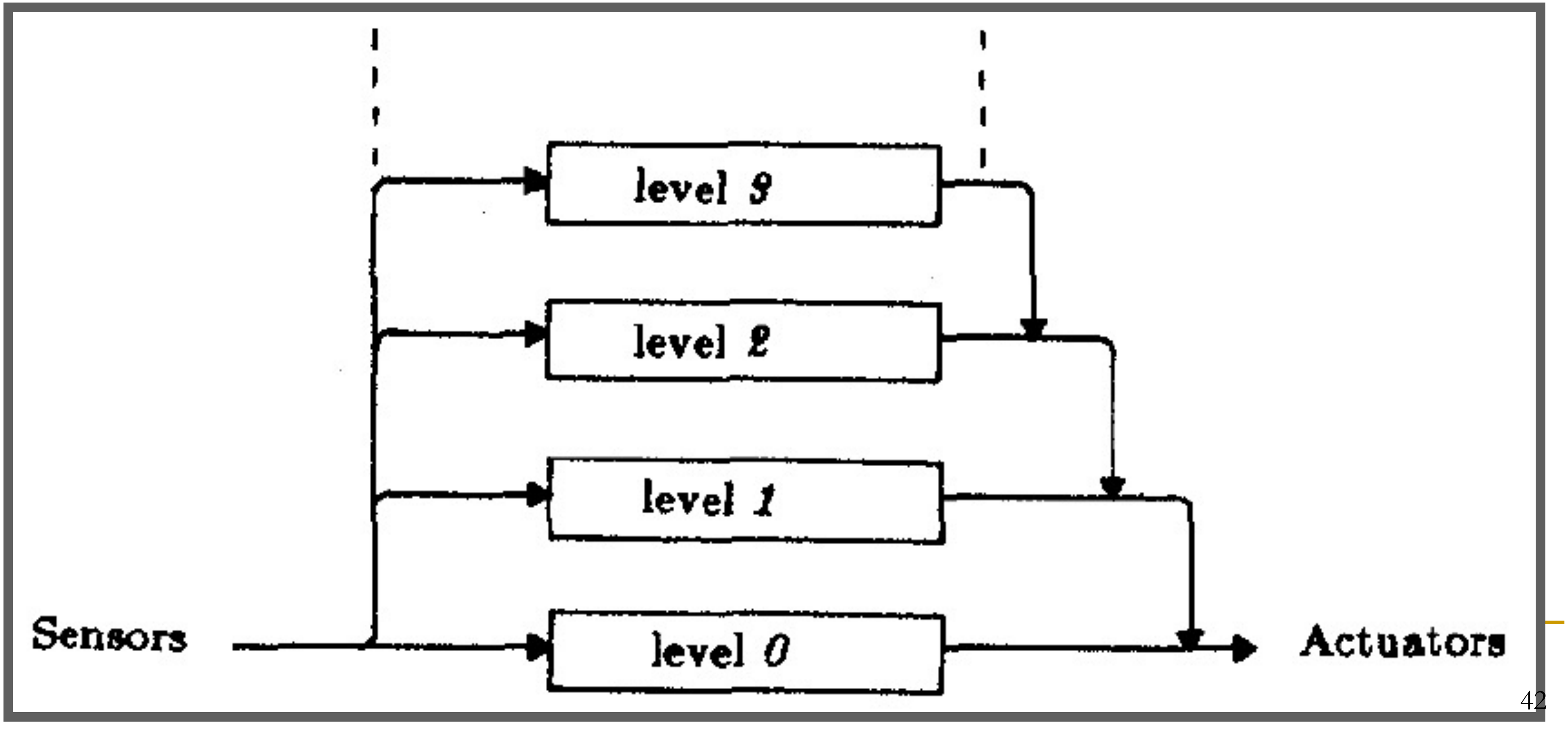
- There are many unsolved (some would say insoluble) problems associated with symbolic AI
- These problems have led some researchers to question the viability of the whole paradigm, and to the development of *reactive* architectures
- Although united by a belief that the assumptions underpinning mainstream AI are in some sense wrong, reactive agent researchers use many different techniques
- We start by reviewing the work of one of the most vocal critics of mainstream AI: Rodney Brooks

Rodney A. Brooks

- M.I.T professor
- Member of M.I.T.’s Artificial Intelligence Lab
- Developed the Subsumption Architecture for robot control in 1986
- His goal was to develop artificial, complete creatures capable of inhabiting our world, not a simplified world

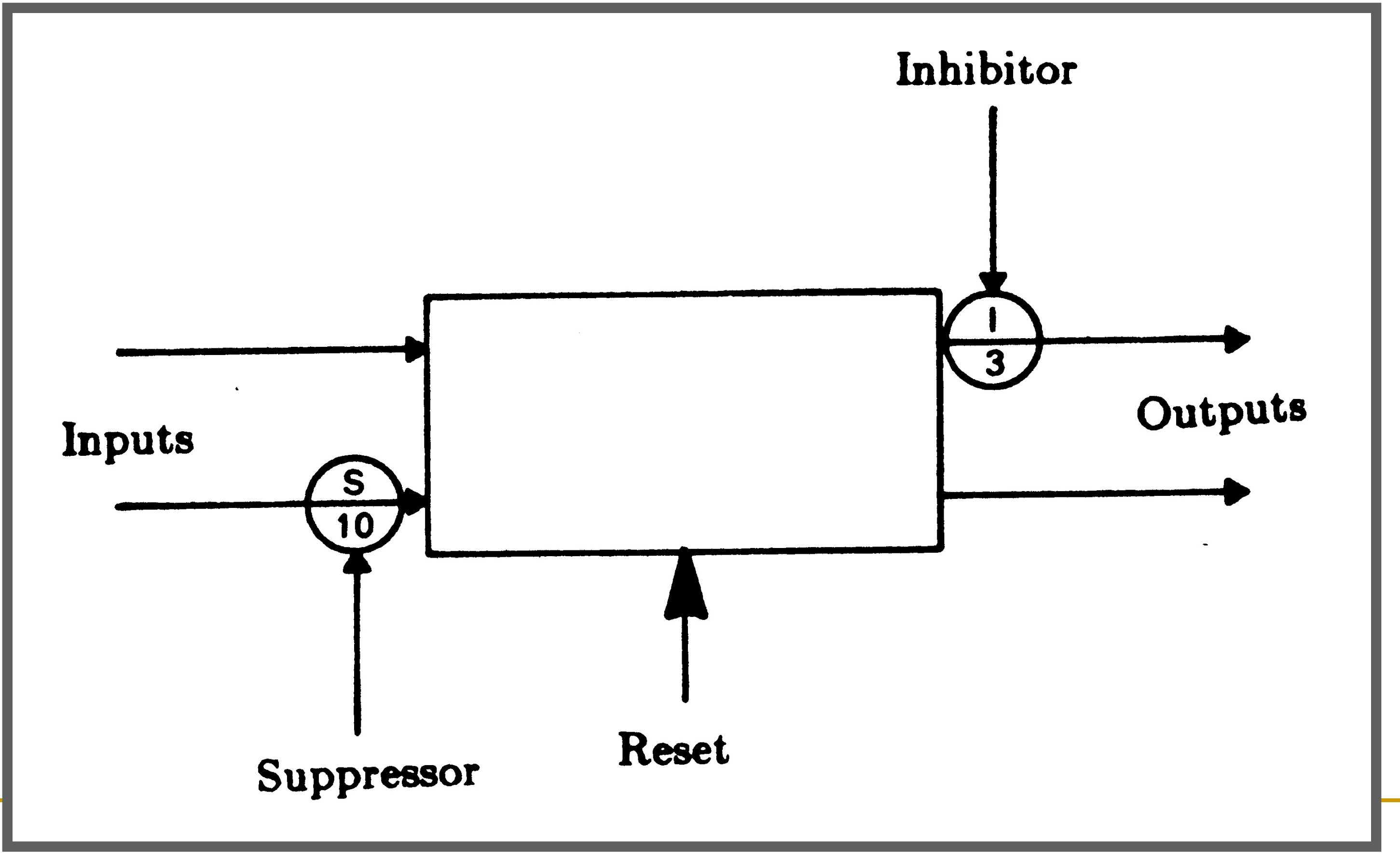
The Subsumption Architecture

- The Subsumption Architecture is:
  - A layering methodology for robot control systems
  - A parallel and distributed method for connecting sensors and actuators in robots



The Subsumption Architecture

- Each layer is made up of connected, simple processors: Augmented Finite State Machines

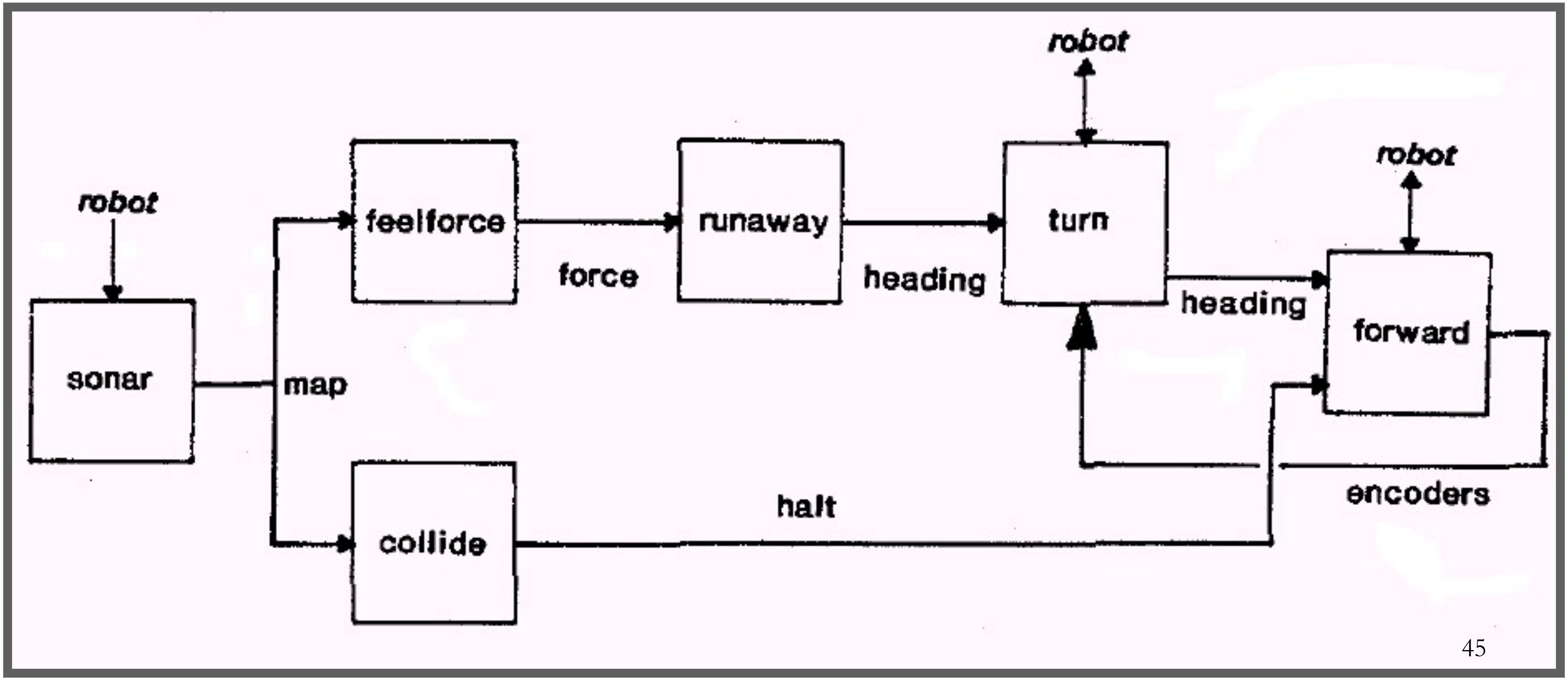


The Subsumption Architecture

- The most important aspect of these FSMs
  - Outputs are simple functions of inputs and local variables
  - Inputs can be suppressed and outputs can be inhibitated
    - This function allows higher levels to subsume the function of lower levels
    - Lower, therefore, still function as they would without the higher levels

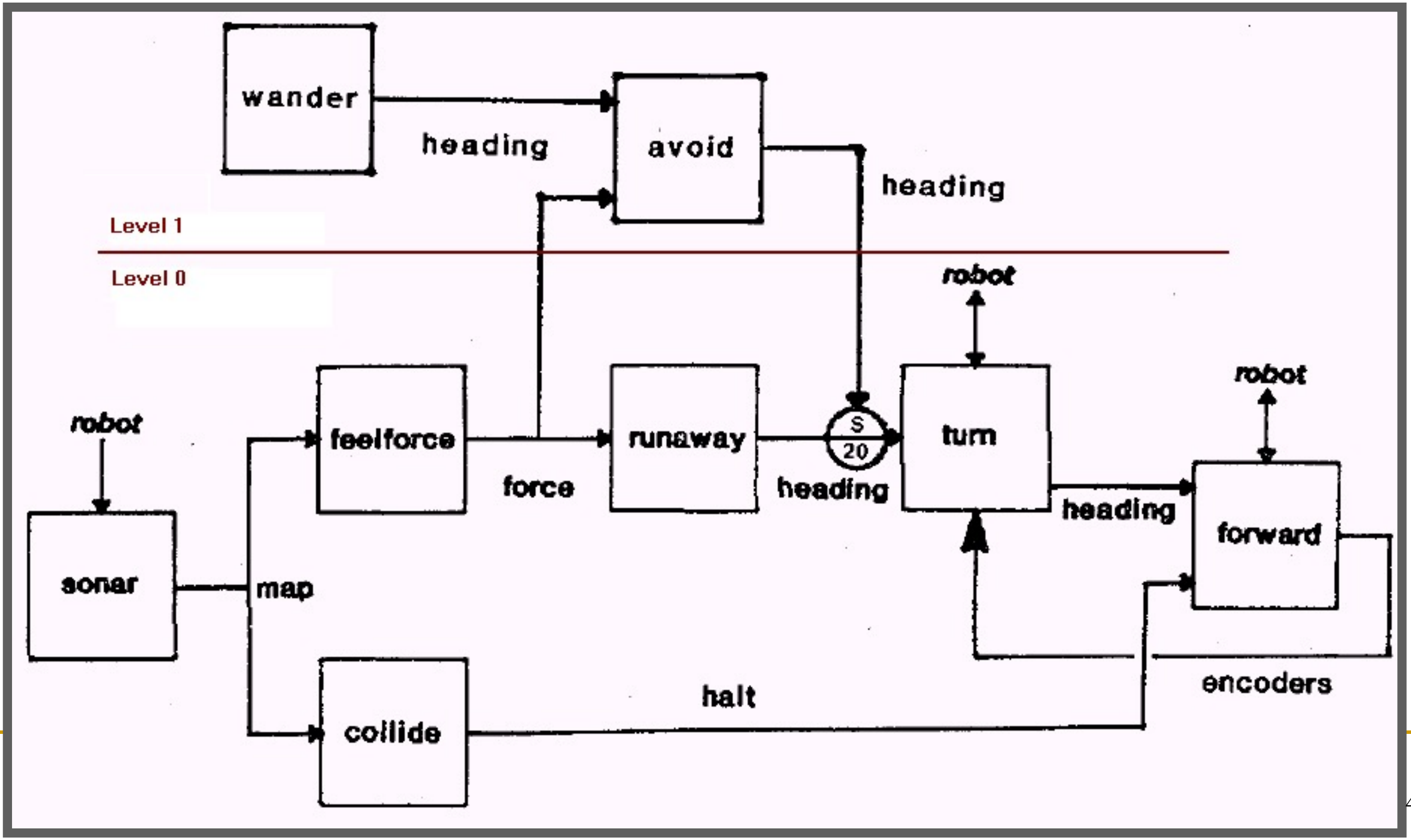
An Example: Allen

- Brooks' first Subsumption robot
- Level 0: Runs away if approached, avoids objects



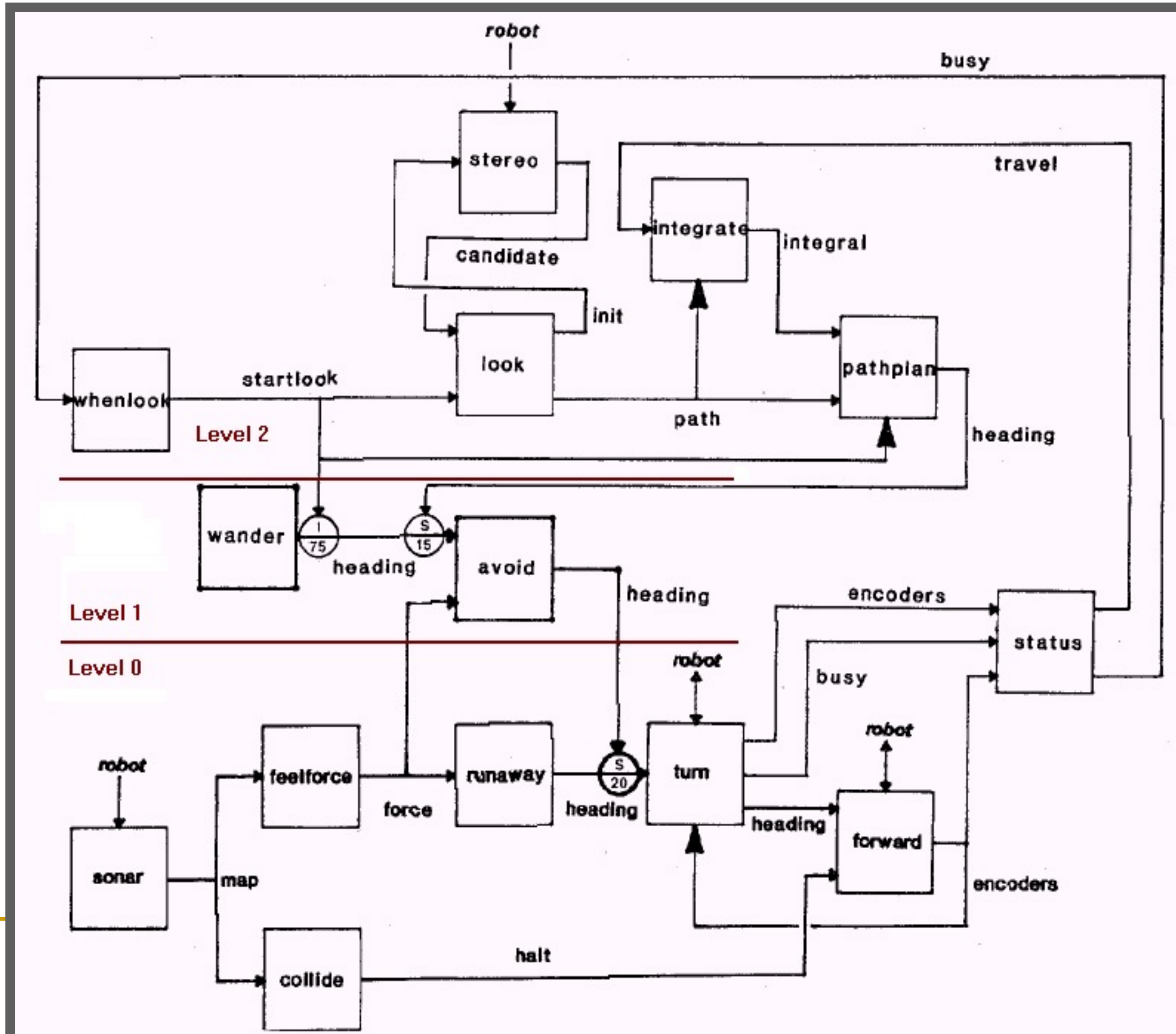
An Example: Allen

- Levels 1 and 0: Adds wandering



An Example: Allen

- Levels 2, 1, and 0: Adds hallway following



Programming Characteristics of Subsumption

- No internal model of the real world because:
  - No free communication
  - No shared memory
- So, use real world as the model
  - "The world really is a rather good model of itself"[1]
  - Very accurate
  - Never out of date
  - No computation needed to keep model up to date
- Real world used for sub-system communication
  - Instead of direct communication, sub-systems just sense the real world

Brooks – behavior languages

- Brooks has put forward three theses:
  1. Intelligent behavior can be generated *without* explicit representations of the kind that symbolic AI proposes
  2. Intelligent behavior can be generated *without* explicit abstract reasoning of the kind that symbolic AI proposes
  3. Intelligence is an *emergent* property of certain complex systems

Brooks – behavior languages

- He identifies two key ideas that have informed his research:
  1. Situatedness and embodiment: 'Real' intelligence is situated in the world, not in disembodied systems such as theorem provers or expert systems
  2. Intelligence and emergence: 'Intelligent' behavior arises as a result of an agent's interaction with its environment. Also, intelligence is 'in the eye of the beholder'; it is not an innate, isolated property

Brooks – behavior languages

- To illustrate his ideas, Brooks built some robots based on his *subsumption architecture*
- A subsumption architecture is a hierarchy of task-accomplishing *behaviors*
- Each behavior is a rather simple rule-like structure
- Each behavior ‘competes’ with others to exercise control over the agent
- Lower layers represent more primitive kinds of behavior (such as avoiding obstacles), and have precedence over layers further up the hierarchy
- The resulting systems are, in terms of the amount of computation they do, *extremely* simple
- Some of the robots do tasks that would be impressive if they were accomplished by symbolic AI systems

Steels' Mars Explorer

- Steels' Mars explorer system, using the subsumption architecture, achieves near-optimal cooperative performance in simulated 'rock gathering on Mars' domain:
  The objective is to explore a distant planet, and in particular, to collect sample of a precious rock. The location of the samples is not known in advance, but it is *known that they tend to be clustered.*

Steels' Mars Explorer Rules

- For individual (non-cooperative) agents, the lowest-level behavior, (and hence the behavior with the highest "priority") is obstacle avoidance:

  if detect an obstacle *then* change direction (1)

- Any samples carried by agents are dropped back at the mother-ship:

  if carrying samples *and* at the base
  then drop samples (2)

- Agents carrying samples will return to the mother-ship:

  if carrying samples and *not* at the base
  then travel up gradient (3)

Steels' Mars Explorer Rules

- Agents will collect samples they find:
  if detect a sample *then* pick sample up (4)
- An agent with "nothing better to do" will explore randomly:
  if true *then* move randomly (5)

Advantages of Reactive Agents

- Simplicity
- Economy
- Computational tractability
- Robustness against failure
- Elegance

Limitations of Reactive Agents

- Agents without environment models must have sufficient information available from local environment
- If decisions are based on *local* environment, how does it take into account *non-local* information (i.e., it has a "short-term" view)
- Difficult to make reactive agents that learn
- Since behavior emerges from component interactions plus environment, it is hard to see how to *engineer* specific agents (no principled methodology exists)
- It is hard to engineer agents with large numbers of behaviors (dynamics of interactions become too complex to understand)

Hybrid Architectures

- Many researchers have argued that neither a completely deliberative nor completely reactive approach is suitable for building agents
- They have suggested using *hybrid* systems, which attempt to marry classical and alternative approaches
- An obvious approach is to build an agent out of two (or more) subsystems:
  - a *deliberative* one, containing a symbolic world model, which develops plans and makes decisions in the way proposed by symbolic AI
  - a *reactive* one, which is capable of reacting to events without complex reasoning

Hybrid Architectures

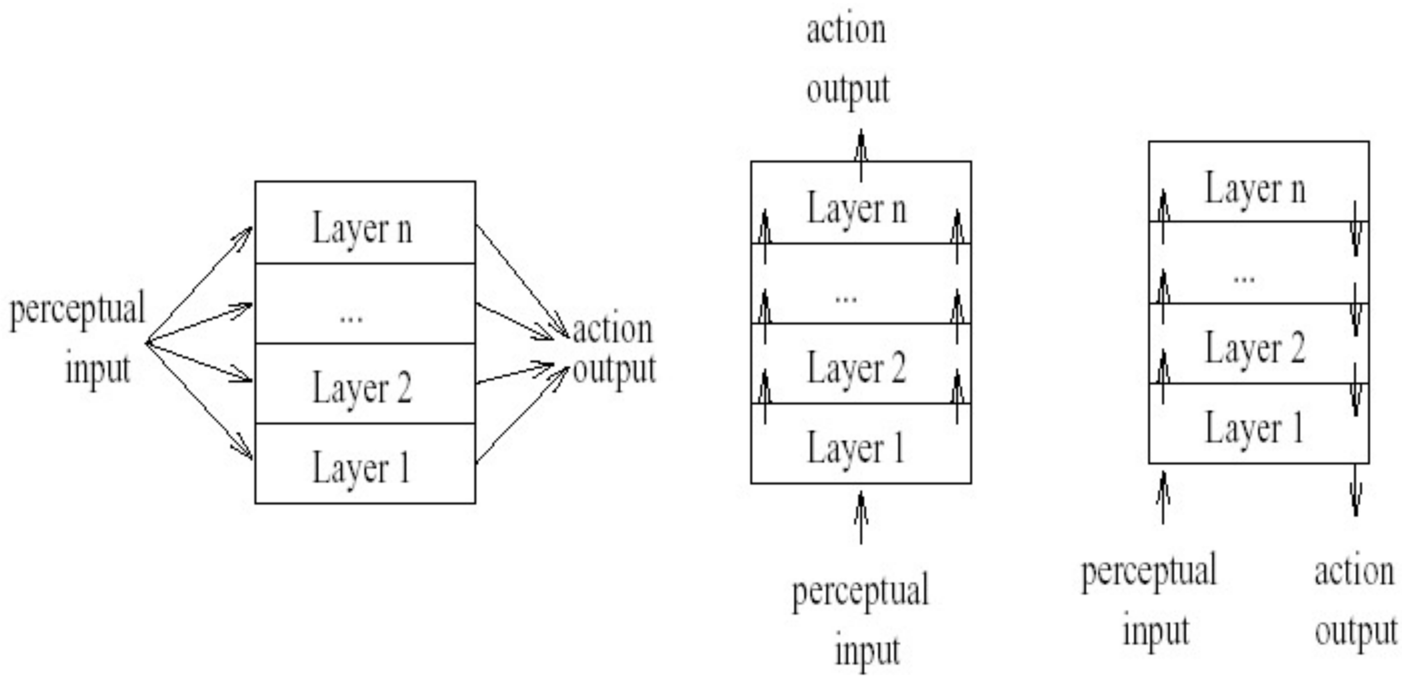
- Often, the reactive component is given some kind of precedence over the deliberative one
- This kind of structuring leads naturally to the idea of a *layered* architecture, of which TOURINGMACHINES and INTERRAP are examples
- In such an architecture, an agent's control subsystems are arranged into a hierarchy, with higher layers dealing with information at increasing levels of abstraction

Hybrid Architectures

- A key problem in such architectures is what kind of control framework to embed the agent's subsystems in, to manage the interactions between the various layers
- *Horizontal layering*
  Layers are each directly connected to the sensory input and action output.
  In effect, each layer itself acts like an agent, producing suggestions as to what action to perform.
- *Vertical layering*
  Sensory input and action output are each dealt with by at most one layer each

Hybrid Architectures

m possible actions suggested by each layer, n layers



(a) Horizontal layering

(b) Vertical layering (One pass control)

(c) Vertical layering (Two pass control)

m^n interactions

$m^2(n-1)$ interactions

Introduces bottleneck in central control system

Not fault tolerant to layer failure

Summary

- Symbolic Reasoning Agents
  - Deductive Reasoning Agents
  - Agent Oriented Programming (AOP)
  - Practical Reasoning Agents
  - Problems with symbolic reasoning agents
- Reactive Agents
  - Limitations of reactive agents
- Hybrid Architectures