DEDUCTIVE REASONING AGENTS

Agent Architectures

- An *agent* is a computer system capable of *flexible autonomous action*…
- Three types of agent *architecture*:
  - symbolic/logical
  - reactive
  - hybrid

Agent Architectures

- We want to build agents, that enjoy the properties of autonomy, reactiveness, pro-activeness, and social ability that we talked about earlier
- This is the area of *agent architectures*
- Maes defines an agent architecture as:
  '[A] particular methodology for building [agents]. It specifies how… the agent can be decomposed into the construction of a set of component modules and how these modules should be made to interact. The total set of modules and their interactions has to provide an answer to the question of how the sensor data and the current internal state of the agent determine the actions… and future internal state of the agent. An architecture encompasses techniques and algorithms that support this methodology.'

Agent Architectures

- Kaelbling considers an agent architecture to be:

  ‘[A] specific collection of software (or hardware) modules, typically designated by boxes with arrows indicating the data and control flow among the modules. A more abstract view of an architecture is as a general methodology for designing particular modular decompositions for particular tasks.’

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–
- From, 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 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

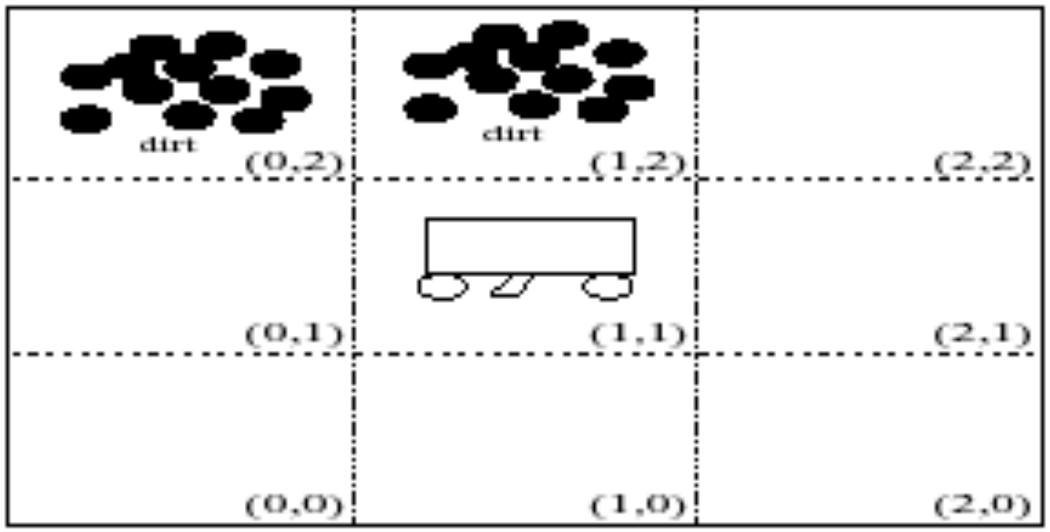
- How can an agent decide what to do using theorem proving?
- Basic idea is to use logic to encode a theory stating the *best* action to perform in any given situation
- Let:
  - ρ be this theory (typically a set of deduction rules)
  - Δ be a logical database that describes the current state of the 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 $\Delta \not\vdash_\rho \neg Do(a)$ 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:

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

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

Deductive Reasoning Agents

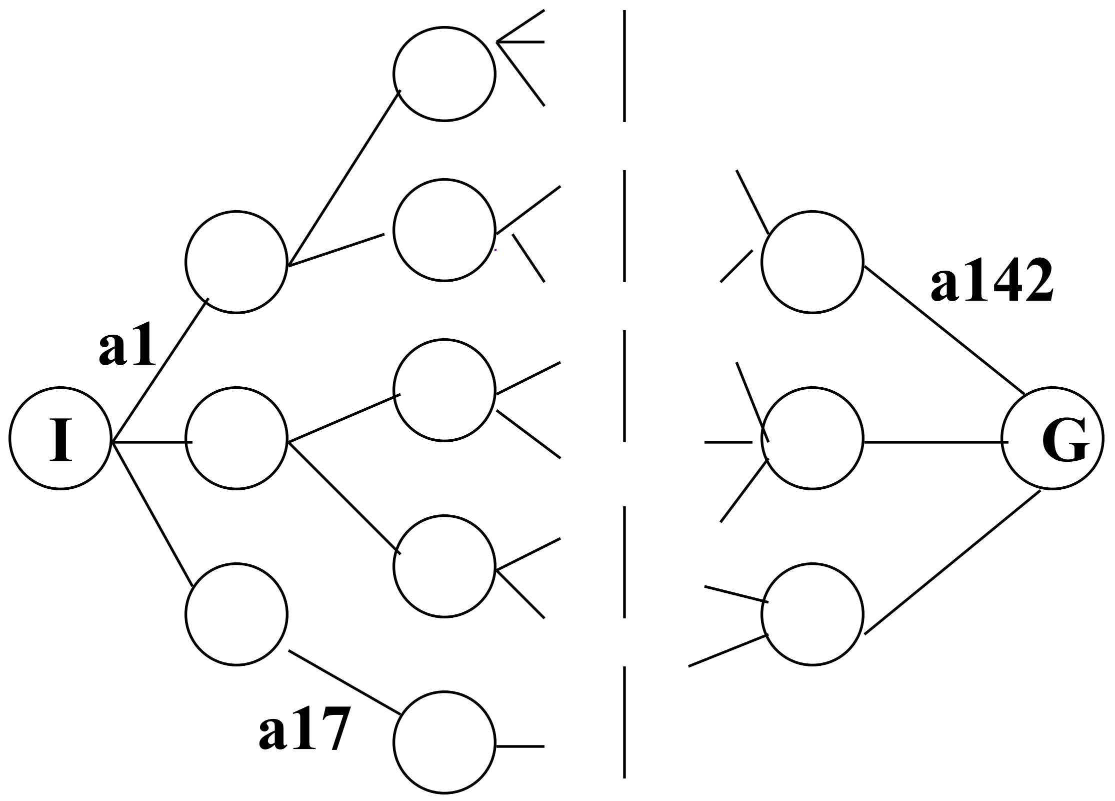
- Problems:
  - How to convert video camera input to *Dirt*(0, 1)?
  - decision making assumes a *static* environment: *calculative* rationality (capacity to compute a perfectly rational decision with the information initially available)
  - decision making using first-order logic is *undecidable*!
- Even where we use *propositional* logic, decision making in the worst case means solving co-NP-complete problems (PS: co-NP-complete = bad news!)
- Typical solutions:
  - weaken the logic
  - use symbolic, non-logical representations
  - shift the emphasis of reasoning from *run time* to *design time*

More Problems…

- The “logical approach” that was presented implies adding and removing things from a database
- That’s not pure logic
- Early attempts at creating a “planning agent” tried to use true logical deduction to the solve the problem

Planning Systems (in general)

- Planning systems find a sequence of actions that transforms an initial state into a goal state



Planning

- Planning involves issues of both Search and Knowledge Rrepresentation
- Sample planning systems:
  - Robot Planning
  - Planning of biological experiments
  - Planning of speech acts
- For purposes of exposition, we use a simple domain – The Blocks World

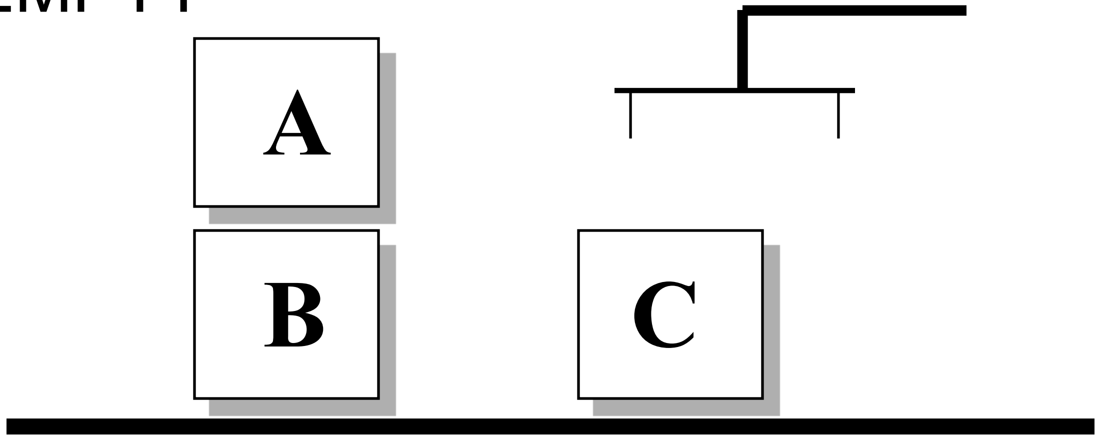
The Blocks World

- The Blocks World (today) consists of equal sized blocks on a table
- A robot arm can manipulate the blocks using the actions:
  - UNSTACK(a, b)
  - STACK(a, b)
  - PICKUP(a)
  - PUTDOWN(a)

The Blocks World

- We also use predicates to describe the world:
  - ON(A,B)
  - ONTABLE(B)
  - ONTABLE(C)
  - CLEAR(A)
  - CLEAR(C)
  - ARMEMPTY

In general:
ON(a,b)
HOLDING(a)
ONTABLE(a)
ARMEMPTY
CLEAR(a)



Logical Formulas to Describe Facts Always True of the World

- And of course we can write general logical truths relating the predicates:

$[\exists x\ \mathrm{HOLDING}(x)] \rightarrow \neg\,\mathrm{ARMEMPTY}$

$\forall x\ [\mathrm{ONTABLE}(x) \rightarrow \neg\ \exists y\ [\mathrm{ON}(x,y)]]$

$\forall x\ [\neg \exists y\ [\mathrm{ON}(y, x)] \rightarrow \mathrm{CLEAR}(x)]$

So…how do we use theorem-proving techniques to construct plans?

Green’s Method

- Add state variables to the predicates, and use a function DO that maps actions and states into new states

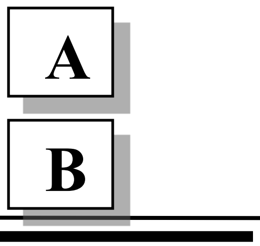
  DO: $A \times S \rightarrow S$

- Example:
  DO(UNSTACK(x, y), S) is a new state

UNSTACK

- So to characterize the action UNSTACK we could write:
  $[\mathrm{CLEAR}(x, s) \wedge \mathrm{ON}(x, y, s)] \rightarrow$
  $[\mathrm{HOLDING}(x, \mathrm{DO}(\mathrm{UNSTACK}(x,y),s)) \wedge$
  $\mathrm{CLEAR}(y, \mathrm{DO}(\mathrm{UNSTACK}(x,y),s))]$
- We can prove that if S0 is
  $\mathrm{ON}(A,B,S0) \wedge \mathrm{ONTABLE}(B,S0) \wedge \mathrm{CLEAR}(A, S0)$
  then

$$\mathrm{HOLDING}(A,\overbrace{\mathrm{DO}(\mathrm{UNSTACK}(A,B),S0)}^{S1}) \wedge$$
$$\mathrm{CLEAR}(B,\underbrace{\mathrm{DO}(\mathrm{UNSTACK}(A,B),S0)}_{S1})$$



More Proving

- The proof could proceed further; if we characterize PUTDOWN:

  HOLDING(x,s) → ONTABLE(x,DO(PUTDOWN(x),s))

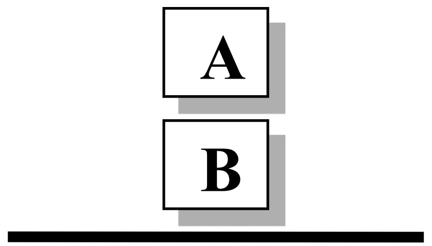
- Then we could prove:

$$\text{ONTABLE}(A,\ \underbrace{\text{DO}(\text{PUTDOWN}(A),\ \underbrace{\text{DO}(\text{UNSTACK}(A,B),\ S0)}_{S1})}_{S2})$$

- The nested actions in this constructive proof give you the plan:

1. UNSTACK(A,B); 2. PUTDOWN(A)

More Proving



- So if we have in our database:
  ON(A,B,S0) ∧ ONTABLE(B,S0) ∧ CLEAR(A,S0)
  and our goal is
  ∃ s(ONTABLE(A, s))
  we could use theorem proving to find the plan
- But could I prove:

$$\text{ONTABLE}(B,\ \underbrace{\text{DO}(\text{PUTDOWN}(A),\ \underbrace{\text{DO}(\text{UNSTACK}(A,B),\ S0)}_{S1})}_{S2})\qquad \boxed{?}$$

The Frame Problem

- How do you determine *what changes* and *what doesn't change* when an action is performed?
- One solution: "Frame axioms" that specify how predicates can remain unchanged after an action
- Example:
  1. $\mathrm{ONTABLE}(z, s) \rightarrow$
     $\mathrm{ONTABLE}(z, \mathrm{DO}(\mathrm{UNSTACK}(x,y),s))$
  2. $[\mathrm{ON}(m, n, s) \wedge \mathrm{DIFF}(m, x)] \rightarrow$
     $\mathrm{ON}(m,n,\mathrm{DO}(\mathrm{UNSTACK}(x,y),s))$

Frame Axioms

- Problem: Unless we go to a higher-order logic, Green’s method forces us to write many frame axioms
- Example:
  $$\mathrm{COLOR}(x, c, s) \rightarrow \mathrm{COLOR}(x,c,\mathrm{DO}(\mathrm{UNSTACK}(y,z),s))$$
- We want to avoid this…other approaches are needed…

Planning Agents

- Since the early 1970s, the AI planning community has been closely concerned with the design of artificial agents
- Planning is essentially automatic programming: the design of a course of action that will achieve some desired goal
- Within the symbolic AI community, it has long been assumed that some form of AI planning system will be a central component of any artificial agent
- Building largely on the early work of Fikes & Nilsson, many planning algorithms have been proposed, and the theory of planning has been well-developed
- But in the mid 1980s, Chapman established some theoretical results which indicate that AI planners will ultimately turn out to be unusable in any time-constrained system