Intelligent Agent

Deductive Reasoning agents
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 (symbolic representation, syntactic manipulation)
- This paradigm is known as *symbolic AI*
- We define an 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:
  – \( \rho \) be this theory (typically a set of rules)
  – \( \Delta \) be a logical database that describes the current state of the world
  – \( Ac \) be the set of actions the agent can perform
  – \( \Delta \models_{\rho} \phi \) mean that \( \phi \) can be proved from \( \Delta \) using \( \rho \)
Deductive Reasoning Agents

/* try to find an action explicitly prescribed */
for each $a \in Ac$ do
  if $\Delta \models_\rho Do(a)$ then
    return $a$
  end-if
end-for

/* try to find an action not excluded */
for each $a \in Ac$ do
  if $\Delta \not\models_\rho \neg Do(a)$ 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:
  \( Ac = \{\text{turn, forward, suck}\} \)

P.S. \textit{turn} means “turn right”
Deductive Reasoning Agents

• Highest priority rule:
  \[ In(x,y) \land Dirt(x,y) \rightarrow Do(suck) \]

• Rules \( \rho \) for determining what to do:

  \[
  \begin{align*}
  In(0, 0) \land Facing(north) \land \neg Dirt(0, 0) & \rightarrow Do(forward) \\
  In(0, 1) \land Facing(north) \land \neg Dirt(0, 1) & \rightarrow Do(forward) \\
  In(0, 2) \land Facing(north) \land \neg Dirt(0, 2) & \rightarrow Do(turn) \\
  In(0, 2) \land Facing(east) & \rightarrow Do(forward)
  \end{align*}
  \]

• …and so on!

• Using these rules, starting at \((0, 0)\) the robot will clear up dirt
Deductive Reasoning Agents

- Problems:
  - How to convert video camera input to \( \text{Dirt}(0, 1) \)?
  - decision making assumes a *static* environment: *calculative* rationality
  - 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 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 Representation

• Sample planning systems:
  – Robot Planning (STRIPS)
  – Planning of biological experiments (MOLGEN)
  – 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:

\[
\exists x \text{ HOLDING}(x) \rightarrow \neg \text{ ARMEMPTY}
\]

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

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

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

   \[ \text{DO: } A \times S \rightarrow S \]

• Example:

   \[ \text{DO(UNSTACK(x, y), S) is a new state} \]
UNSTACK

• So to characterize the action UNSTACK we could write:
  \[ [ \text{CLEAR}(x, s) \land \text{ON}(x, y, s) ] \rightarrow [\text{HOLDING}(x, \text{DO(UNSTACK}(x,y),s))] \land \text{CLEAR}(y, \text{DO(UNSTACK}(x,y),s))] \]

• We can prove that if S0 is
  \[ \text{ON}(A,B,S0) \land \text{ONTABLE}(B,S0) \land \text{CLEAR}(A, S0) \]
  then

  \[ \text{HOLDING}(A, \text{DO(UNSTACK}(A,B),S0)) \land \text{CLEAR}(B, \text{DO(UNSTACK}(A,B),S0)) \]
More Proving

• The proof could proceed further; if we characterize PUTDOWN:  
  \( \text{HOLDING}(x,s) \rightarrow \text{ONTABLE}(x, \text{DO}(\text{PUTDOWN}(x),s)) \)

• Then we could prove:  
  \( \text{ONTABLE}(A, \text{DO}(\text{PUTDOWN}(A), \text{DO}(\text{UNSTACK}(A,B), S0))) \)

• The nested actions in this constructive proof give you the plan:  
  1. \text{UNSTACK}(A,B); 2. \text{PUTDOWN}(A)
More Proving

- So if we have in our database:
  \[ \text{ON}(A,B,S0) \land \text{ONTABLE}(B,S0) \land \text{CLEAR}(A,S0) \]
  and our goal is
  \[ \exists s(\text{ONTABLE}(A, s)) \]
  we could use theorem proving to find the plan

- But could I prove:
  \[ \text{ONTABLE}(B, \text{DO}(\text{PUTDOWN}(A), \text{DO}(\text{UNSTACK}(A,B), S0)))) \]
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. \( \text{ONTABLE}(z, s) \rightarrow \text{ONTABLE}(z, \text{DO} (\text{UNSTACK}(x,y), s)) \)
  2. \( [\text{ON}(m, n, s) \land \text{DIFF}(m, x)] \rightarrow \text{ON}(m, n, \text{DO} (\text{UNSTACK}(x,y), s)) \)
Frame Axioms

• Problem: Unless we go to a higher-order logic, Green’s method forces us to write many frame axioms
• Example:
  \[ \text{COLOR}(x, c, s) \rightarrow \text{COLOR}(x,c,\text{DO}(\text{UNSTACK}(y,z),s)) \]
• We want to avoid this…other approaches are needed…
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
• The motivation behind such a proposal is that, as we humans use the intentional stance as an *abstraction* mechanism for representing the properties of complex systems. *In the same way that we use the intentional stance to describe humans, it might be useful to use the intentional stance to program machines.*
AGENT0

• Shoham suggested that a complete AOP system will have 3 components:
  – a logic for specifying agents and describing their mental states
  – an interpreted programming language for programming agents
  – an ‘agentification’ process, for converting ‘neutral applications’ (e.g., databases) into agents
• Results only reported on first two components.
• Relationship between logic and programming language is *semantics*
• We will skip over the logic(!), and consider the first AOP language, AGENT0
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
• 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

• A commitment rule:

```commit
COMMITS

  ( 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)

)```

Phạm Bảo Sơn - Intelligent Agent
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

Phạm Bảo Sơn - Intelligent Agent 32