# LECTURE 5: PRACTICAL REASONING AGENTS

An Introduction to Multiagent Systems

CIS 716.5, Spring 2005



## 1 What is 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)

- Distinguish practical reasoning from *theoretical reasoning*.

  Theoretical reasoning is directed towards beliefs.






## The Components of Practical Reasoning

- Human practical reasoning consists of two activities:
  - *deliberation*
    deciding *what* state of affairs we want to achieve
    — the outputs of deliberation are *intentions*;
  - *means-ends reasoning*
    deciding *how* to achieve these states of affairs
    — the outputs of means-ends reasoning are *plans*.






## 2 Intentions in Practical Reasoning

1. Intentions pose problems for agents, who need to determine ways of achieving them.

   *If I have an intention to $\phi$, you would expect me to devote resources to deciding how to bring about $\phi$.*

2. Intentions provide a "filter" for adopting other intentions, which must not conflict.

   *If I have an intention to $\phi$, you would not expect me to adopt an intention $\psi$ that was incompatible with $\phi$.*

3. Agents track the success of their intentions, and are inclined to try again if their attempts fail.

   *If an agent's first attempt to achieve $\phi$ fails, then all other things being equal, it will try an alternative plan to achieve $\phi$.*

4. Agents believe their intentions are possible.

   *That is, they believe there is at least some way that the intentions could be brought about.*

5. Agents do not believe they will not bring about their intentions.

   *It would not be rational of me to adopt an intention to $\phi$ if I believed I would fail with $\phi$.*

6. Under certain circumstances, agents believe they will bring about their intentions.

   *If I intend $\phi$, then I believe that under "normal circumstances" I will succeed with $\phi$.*



7. Agents need not intend all the expected side effects of their intentions.

   *If I believe $\phi \Rightarrow \psi$ and I intend that $\phi$, I do not necessarily intend $\psi$ also. (Intentions are not closed under implication.)*

   This last problem is known as the *side effect* or *package deal* problem.

   I may believe that going to the dentist involves pain, and I may also intend to go to the dentist — but this does not imply that I intend to suffer pain!



## Intentions are Stronger than 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)

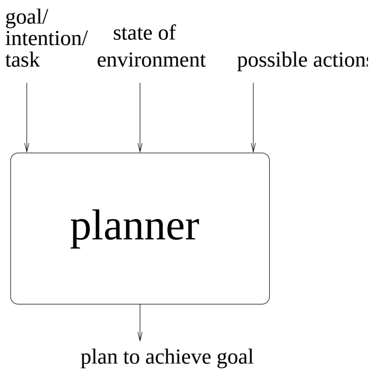


## Means-ends Reasoning/Planning

- Planning is the design of a course of action that will achieve some desired goal.
- Basic idea is to give a planning system:
  - (representation of) goal/intention to achieve;
  - (representation of) actions it can perform; and
  - (representation of) the environment;

  and have it generate a *plan* to achieve the goal.
- This is *automatic programming*.





## Representations

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



- We'll illustrate the techniques with reference to the *blocks world*.
- Contains a robot arm, 2 blocks (A and B) of equal size, and a table-top.



- 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 $x$ |

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

$$\begin{array}{l} Clear(A) \\ On(A,B) \\ OnTable(B) \\ OnTable(C) \end{array}$$

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






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

$$\{OnTable(A), \quad OnTable(B), \quad OnTable(C)\}$$






- *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*.






- Example 1:

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

$$\begin{array}{ll} & Stack(x,y) \\ \text{pre} & Clear(y) \wedge Holding(x) \\ \text{del} & Clear(y) \wedge Holding(x) \\ \text{add} & ArmEmpty \wedge On(x,y) \end{array}$$

- Example 2:

  The *unstack* action occurs when the robot arm picks an object $x$ up from on top of another object $y$.

$$\begin{array}{ll} & UnStack(x,y) \\ \text{pre} & On(x,y) \wedge Clear(x) \wedge ArmEmpty \\ \text{del} & On(x,y) \wedge ArmEmpty \\ \text{add} & Holding(x) \wedge Clear(y) \end{array}$$

  Stack and UnStack are *inverses* of one-another.






- Example 3:

  The *pickup* action occurs when the arm picks up an object $x$ from the table.

$$\begin{array}{ll} & Pickup(x) \\ \text{pre} & Clear(x) \wedge OnTable(x) \wedge ArmEmpty \\ \text{del} & OnTable(x) \wedge ArmEmpty \\ \text{add} & Holding(x) \end{array}$$






- Example 4:

  The *putdown* action occurs when the arm places the object $x$ onto the table.

$$\begin{array}{ll} & PutDown(x) \\ \text{pre} & Holding(x) \\ \text{del} & Holding(x) \\ \text{add} & Holding(x) \wedge ArmEmpty \end{array}$$

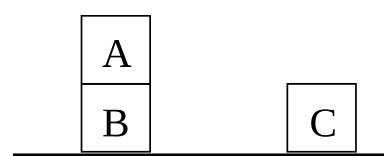




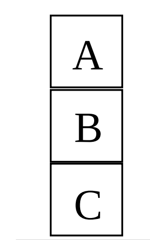

- What is a plan?

  A sequence (list) of actions, with variables replaced by constants.

- So, to get from:




to



- We need the set of actions:

$$\begin{array}{c} \mathit{Unstack}(A) \\ \mathit{Putdown}(A) \\ \mathit{Pickup}(B) \\ \mathit{Stack}(B,C) \\ \mathit{Pickup}(A) \\ \mathit{Stack}(A,B) \end{array}$$





## 3 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).)





- Problem: deliberation and means-ends reasoning processes are not instantaneous.

  They have a *time cost*.
- Suppose that deliberation is *optimal* in that if it selects some intention to achieve, then this is the best thing for the agent. (Maximises expected utility.)
- So the agent has selects an intention to achieve that would have been optimal *at the time it observed the world*.

  This is *calculative rationality*.

  The world *may change*.
- Deliberation is only half of the problem: the agent still has to determine *how* to achieve the intention.





- Let's make the algorithm more formal.

Agent Control Loop Version 2  
1. $B := B_0$; /* initial beliefs */  
2. while true do  
3. get next percept $\rho$;  
4. $B := \mathit{brf}(B, \rho)$;  
5. $I := \mathit{deliberate}(B)$;  
6. $\pi := \mathit{plan}(B, I)$;  
7. $\mathit{execute}(\pi)$  
8. end while

## 4 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.





- The *deliberate* function can be decomposed into two distinct functional components:
  - *option generation*
    in which the agent generates a set of possible alternatives; and
    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*).
  - *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.





```
Agent Control Loop Version 3

    B := B_0;
    I := I_0;
    while true do
        get next percept ρ;
        B := brf(B, ρ);
        D := options(B, I);
        I := filter(B, D, I);
        π := plan(B, I);
        execute(π)
    end while
```





## 5 Commitment Strategies

Some time in the not-so-distant future, you are having trouble with your new household robot. You say "Willie, bring me a beer." The robot replies "OK boss." Twenty minutes later, you screech "Willie, why didn't you bring me that beer?" It answers "Well, I intended to get you the beer, but I decided to do something else." Miffed, you send the wise guy back to the manufacturer, complaining about a lack of commitment. After retrofitting, Willie is returned, marked "Model C: The Committed Assistant." Again, you ask Willie to bring you a beer. Again, it accedes, replying "Sure thing." Then you ask: "What kind of beer did you buy?" It answers: "Genessee." You say "Never mind." One minute later, Willie trundles over with a Genessee in its gripper. [...] After still more tinkering, the manufacturer sends Willie back, promising no more problems with its commitments. So, being a somewhat trusting customer, you accept the rascal back into your household, but as a test, you ask it to bring you your last beer. [...] The robot gets the beer and starts towards you. As it approaches, it lifts its arm, wheels around, deliberately smashes the bottle, and trundles off. Back at the plant, when interrogated by customer service as to why it had abandoned its commitments, the robot replies that according to its specifications, it kept its commitments as long as required — commitments must be dropped when fulfilled or impossible to achieve. By smashing the bottle, the commitment became unachievable.

## Degrees of Commitment

- *Blind commitment*

  A blindly committed agent will continue to maintain an intention until it believes the intention has actually been achieved. Blind commitment is also sometimes referred to as *fanatical* commitment.
- *Single-minded commitment*

  A single-minded agent will continue to maintain an intention until it believes that either the intention has been achieved, or else that it is no longer possible to achieve the intention.






- An agent has commitment both to *ends* (i.e., the state of affairs it wishes to bring about), and *means* (i.e., the mechanism via which the agent wishes to achieve the state of affairs).
- Currently, our agent control loop is overcommitted, both to means and ends.

  Modification: *replan* if ever a plan goes wrong.






```
Agent Control Loop Version 4

B := B_0;
I := I_0;
while true do
    get next percept ρ;
    B := brf(B, ρ);
    D := options(B, I);
    I := filter(B, D, I);
    π := plan(B, I);

⋮
```






```
Agent Control Loop Version 4 (cont)
⋮
    while not empty(π) do
        α := hd(π);
        execute(α);
        π := tail(π);
        get next percept ρ;
        B := brf(B, ρ);
        if not sound(π, I, B) then
            π := plan(B, I)
        end-if
    end-while
end-while
```

- Still overcommitted to intentions: Never stops to consider whether or not its intentions are appropriate.
- Modification: stop to determine whether intentions have succeeded or whether they are impossible:

*single-minded commitment*.





Agent Control Loop Version 5

$B := B_0$;
$I := I_0$;
while true do
    get next percept $\rho$;
    $B := brf(B, \rho)$;
    $D := options(B, I)$;
    $I := filter(B, D, I)$;
    $\pi := plan(B, I)$;
⋮





⋮
    while not $empty(\pi)$
        or $succeeded(I, B)$
        or $impossible(I, B)$) do
        $\alpha := hd(\pi)$;
        $execute(\alpha)$;
        $\pi := tail(\pi)$;
        get next percept $\rho$;
        $B := brf(B, \rho)$;
        if not $sound(\pi, I, B)$ then
            $\pi := plan(B, I)$
        end-if
    end-while
end-while





## 6 Intention Reconsideration

- Our agent gets to reconsider its intentions once every time around the outer control loop, i.e., when:
  - it has completely executed a plan to achieve its current intentions; or
  - it believes it has achieved its current intentions; or
  - it believes its current intentions are no longer possible.
- This is limited in the way that it permits an agent to *reconsider* its intentions.
- Modification: Reconsider intentions after executing every action.

Agent Control Loop Version 6

```
⋮
10.     while not (empty(π) or succeeded(I,B)
                 or impossible(I,B)) do
11.         α := hd(π);
12.         execute(α);
13.         π := tail(π);
14.         get next percept ρ;
15.         B := brf(B,ρ);
16.         D := options(B,I);
17.         I := filter(B,D,I);
18.         if not sound(π,I,B) then
19.             π := plan(B,I)
⋮
```



- But intention reconsideration is *costly*!

  A dilemma:

  – an agent that does not stop to reconsider its intentions sufficiently often will continue attempting to achieve its intentions even after it is clear that they cannot be achieved, or that there is no longer any reason for achieving them;

  – an agent that *constantly* reconsiders its attentions may spend insufficient time actually working to achieve them, and hence runs the risk of never actually achieving them.

- Solution: incorporate an explicit *meta-level control* component, that decides whether or not to reconsider.



Agent Control Loop Version 7

```
⋮
10.     while not (empty(π) or succeeded(I,B)
                 or impossible(I,B)) do
11.         α := hd(π);
12.         execute(α);
13.         π := tail(π);
14.         get next percept ρ;
15.         B := brf(B,ρ);
16.         if reconsider(I,B) then
17.             D := options(B,I);
18.             I := filter(B,D,I);
19.         if not sound(π,I,B) then π := plan(B,I)
⋮
```



- The possible interactions between meta-level control and deliberation are:

| Situation number | Chose to deliberate? | Changed intentions? | Would have changed intentions? | $reconsider(\ldots)$ optimal? |
|---|---|---|---|---|
| 1 | No | — | No | Yes |
| 2 | No | — | Yes | No |
| 3 | Yes | No | — | No |
| 4 | Yes | Yes | — | Yes |

- An important assumption: cost of $reconsider(\ldots)$ is *much* less than the cost of the deliberation process itself.

- In situation (1), the agent did not choose to deliberate, and as a consequence, did not choose to change intentions. Moreover, if it *had* chosen to deliberate, it would not have changed intentions. In this situation, the $reconsider(\ldots)$ function is behaving optimally.
- In situation (2), the agent did not choose to deliberate, but if it had done so, it *would* have changed intentions. In this situation, the $reconsider(\ldots)$ function is not behaving optimally.
- In situation (3), the agent chose to deliberate, but did not change intentions. In this situation, the $reconsider(\ldots)$ function is not behaving optimally.
- In situation (4), the agent chose to deliberate, and did change intentions. In this situation, the $reconsider(\ldots)$ function is behaving optimally.



## 7 Optimal Intention Reconsideration

- Kinny and Georgeff's experimentally investigated effectiveness of intention reconsideration strategies.
- Two different types of reconsideration strategy were used:
  - *bold* agents
    never pause to reconsider intentions, and
  - *cautious* agents
    stop to reconsider after every action.
- *Dynamism* in the environment is represented by the *rate of world change*, $\gamma$.



- Results:
  - If $\gamma$ is low (i.e., the environment does not change quickly), then bold agents do well compared to cautious ones. This is because cautious ones waste time reconsidering their commitments while bold agents are busy working towards — and achieving — their intentions.
  - If $\gamma$ is high (i.e., the environment changes frequently), then cautious agents tend to outperform bold agents. This is because they are able to recognize when intentions are doomed, and also to take advantage of serendipitous situations and new opportunities when they arise.

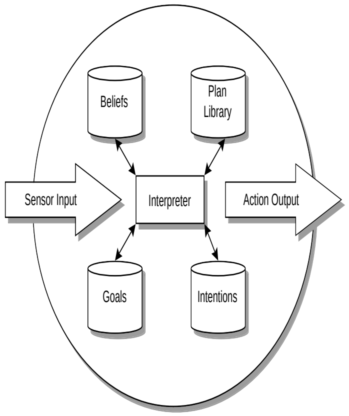
 



## 8 Implemented BDI Agents: PRS

- We now make the discussion even more concrete by introducing an actual agent architecture: the PRS.
- In the PRS, each agent is equipped with a *plan library*, representing that agent's *procedural knowledge*: knowledge about the mechanisms that can be used by the agent in order to realise its intentions.
- The options available to an agent are directly determined by the plans an agent has: an agent with no plans has no options.
- In addition, PRS agents have explicit representations of beliefs, desires, and intentions, as above.








## Example PRS (JAM) System

```
GOALS:

    ACHIEVE blocks_stacked;

FACTS:

// Block1 on Block2 initially so need to clear Block2 before stacking.

    FACT ON "Block1" "Block2";
    FACT ON "Block2" "Table";
    FACT ON "Block3" "Table";
    FACT CLEAR "Block1";
    FACT CLEAR "Block3";
    FACT CLEAR "Table";
    FACT initialized "False";
```





```
Plan: {
NAME: "Top-level plan"
DOCUMENTATION:
    "Establish Block1 on Block2 on Block3."
GOAL:
    ACHIEVE blocks_stacked;
CONTEXT:
BODY:
    EXECUTE print "Goal is Block1 on Block2 on Block2 on Table.\n";
    EXECUTE print "World Model at start is:\n";
    EXECUTE printWorldModel;
    EXECUTE print "ACHIEVEing Block3 on Table.\n";
    ACHIEVE ON "Block3" "Table";
    EXECUTE print "ACHIEVEing Block2 on Block3.\n";
    ACHIEVE ON "Block2" "Block3";
    EXECUTE print "ACHIEVEing Block1 on Block2.\n";
    ACHIEVE ON "Block1" "Block2";
    EXECUTE print "World Model at end is:\n";
    EXECUTE printWorldModel;
}
```





```
Plan: {
NAME: "Stack blocks that are already clear"
GOAL:
    ACHIEVE ON $OBJ1 $OBJ2;
CONTEXT:
BODY:
    EXECUTE print "Making sure " $OBJ1 " is clear\n";
    ACHIEVE CLEAR $OBJ1;
    EXECUTE print "Making sure " $OBJ2 " is clear.\n";
    ACHIEVE CLEAR $OBJ2;
    EXECUTE print "Moving " $OBJ1 " on top of " $OBJ2 ".\n";
    PERFORM move $OBJ1 $OBJ2;
UTILITY: 10;

FAILURE:
    EXECUTE print "\n\nStack blocks failed!\n\n";
}
```

```
Plan: {
NAME: "Clear a block"
GOAL:
    ACHIEVE CLEAR $OBJ;
CONTEXT:
    FACT ON $OBJ2 $OBJ;
BODY:
    EXECUTE print "Clearing " $OBJ2 " from on top of " $OBJ "\n";
    EXECUTE print "Moving " $OBJ2 " to table.\n";
    ACHIEVE ON $OBJ2 "Table";

EFFECTS:
    EXECUTE print "CLEAR: Retracting ON " $OBJ2 " " $OBJ "\n";
    RETRACT ON $OBJ1 $OBJ;

FAILURE:
    EXECUTE print "\n\nClearing block " $OBJ " failed!\n\n";
}
```






```
Plan: {
NAME: "Move a block onto another object"
GOAL:
    PERFORM move $OBJ1 $OBJ2;
CONTEXT:
    FACT CLEAR $OBJ1;
    FACT CLEAR $OBJ2;
BODY:
    EXECUTE print "Performing low-level move action"
    EXECUTE print " of " $OBJ1 " to " $OBJ2 ".\n";

EFFECTS:
    WHEN : TEST (!= $OBJ2 "Table") {
EXECUTE print "    Retracting CLEAR " $OBJ2 "\n";
RETRACT CLEAR $OBJ2;
    };
    FACT ON $OBJ1 $OBJ3;
    EXECUTE print "    move: Retracting ON " $OBJ1 " " $OBJ3 "\n";
    RETRACT ON $OBJ1 $OBJ3;
    EXECUTE print "    move: Asserting CLEAR " $OBJ3 "\n";
    ASSERT CLEAR $OBJ3;
    EXECUTE print "    move: Asserting ON " $OBJ1 " " $OBJ2 "\n\n";
    ASSERT ON $OBJ1 $OBJ2;

FAILURE:
    EXECUTE print "\n\nMove failed!\n\n";
$}
```






## 9 Implemented BDI Agents: IRMA

- IRMA has four key symbolic data structures:
  - a *plan library*, and
  - explicit representations of
    - *beliefs*: information available to the agent — may be represented symbolically, but may be as simple as PASCAL variables;
    - *desires*: those things the agent would *like* to make true — think of desires as *tasks* that the agent has been allocated; in humans, not necessarily logically consistent, but our agents will be! (goals);
    - *intentions*: desires that the agent has *chosen* and *committed to*.

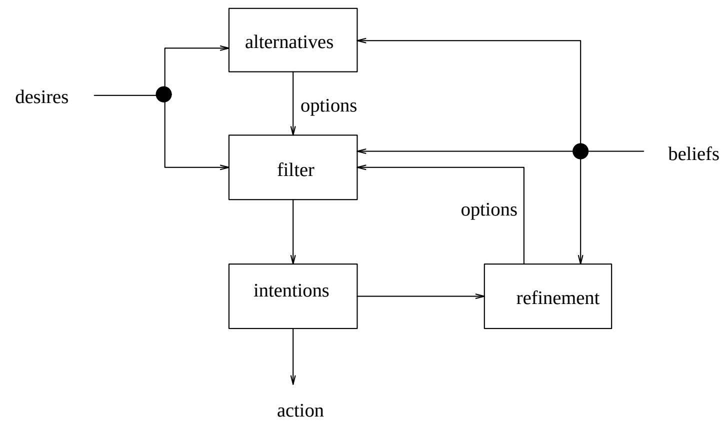





- Additionally, the architecture has:
  - a *reasoner* for reasoning about the world; an inference engine;
  - a *means-ends analyzer* determines which plans might be used to achieve intentions;
  - an *opportunity analyzer* monitors the environment, and as a result of changes, generates new options;
  - a *filtering process* determines which options are compatible with current intentions; and
  - a *deliberation process* responsible for deciding upon the 'best' intentions to adopt.





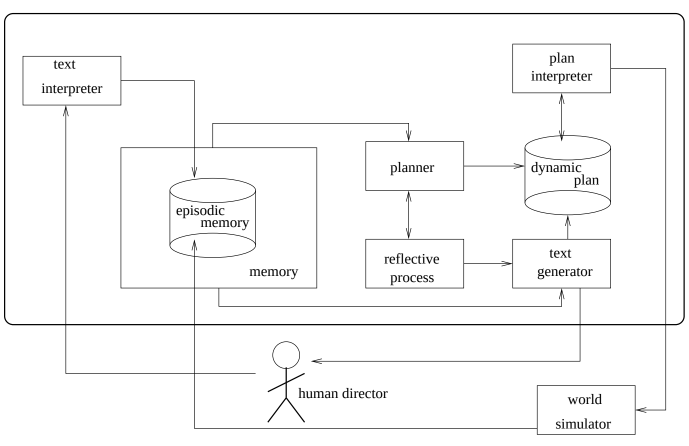


## HOMER — An Agent That Plans

- Vere & Bickmore developed HOMER: a simulated robot submarine, in a two-dimensional 'Seaworld'.
- HOMER takes instructions from a user in a subset of English with about an 800 word vocabulary.
- HOMER can plan how to achieve its instructions, (which usually relate to collecting and moving items around the Seaworld), and can then execute its plans, modifying them as required during execution.
- HOMER has a limited *episodic memory*, and using this, is able to answer questions about its past experiences.





## Dialogues with HOMER

```
EXAMPLE DIALOGUE 1
U:  What is in front of you?
H:  A log.
U:  Do you own the log.
H:  No, I don't.
U:  The log belongs to you.
H:  Oh.
[...]
U:  Turn away from your log.
H:  OK, I am turning.
[LOG IS MOVED]
U:  Turn around.
H:  OK, I'm turning.
H:  I've lost the log!
```






```
EXAMPLE DIALOGUE 2
U:  Drop the package at the barge
    next Saturday at 9pm.
H:  OK.
[HOMER GENERATES PLAN]
U:  Are you going to the barge
    next Saturday?
H:  Yes.
U:  What time?
H:  8.56pm
[USER ADDS CONTRADICTORY INSTRUCTIONS,
 WHICH HOMER DETECTS.]
```






## 10 BDI Theory & Practice

- We now consider the *semantics* of BDI architectures: to what extent does a BDI agent satisfy a *theory of agency*.
- In order to give a semantics to BDI architectures, Rao & Georgeff have developed *BDI logics*: non-classical logics with modal connectives for representing beliefs, desires, and intentions.
- The 'basic BDI logic' of Rao and Georgeff is a quantified extension of the expressive branching time logic CTL$^*$.
- Underlying semantic structure is a *labeled branching time* framework.






## 11 BDI Logic

- From classical logic: $\wedge, \vee, \neg, \ldots$.
- The CTL$^*$ *path quantifiers*:

  A$\phi$ 'on all paths, $\phi$'
  E$\phi$ 'on some paths, $\phi$'

- The BDI connectives:

  $(\mathsf{Bel}\ i\ \phi)$ $i$ believes $\phi$
  $(\mathsf{Des}\ i\ \phi)$ $i$ desires $\phi$
  $(\mathsf{Int}\ i\ \phi)$ $i$ intends $\phi$

- Semantics of B-D-I components are given via accessibility relations over 'worlds', where each world is itself a branching time structure.

  Properties required of accessibility relations ensure belief logic KD45, desire logic KD, intention logic KD.

  (Plus interrelationships. . . )






- Let us now look at some possible axioms of BDI logic, and see to what extent the BDI architecture could be said to satisfy these axioms.
- In what follows, let
  - $\alpha$ be an *O-formula*, i.e., one which contains no positive occurrences of A;
  - $\phi$ be an arbitrary formula.






- *Belief goal compatibility*:

$$(\mathsf{Des}\ \alpha) \Rightarrow (\mathsf{Bel}\ \alpha)$$

  States that if the agent has a goal to optionally achieve something, this thing must be an option.

  This axiom is operationalized in the function *options*: an option should not be produced if it is not believed possible.

- *Goal-intention compatibility*:

$$(\mathsf{Int}\ \alpha) \Rightarrow (\mathsf{Des}\ \alpha)$$

  States that having an intention to optionally achieve something implies having it as a goal (i.e., there are no intentions that are not goals).

  Operationalized in the *deliberate* function.






- *Volitional commitment*:

$$(\mathsf{Int}\ does(a)) \Rightarrow does(a)$$

  If you intend to perform some action $a$ next, then you do $a$ next.

  Operationalized in the *execute* function.

- *Awareness of goals & intentions*:

$$(\mathsf{Des}\ \phi) \Rightarrow (\mathsf{Bel}\ (\mathsf{Des}\ \phi))$$
$$(\mathsf{Int}\ \phi) \Rightarrow (\mathsf{Bel}\ (\mathsf{Int}\ \phi))$$

  Requires that new intentions and goals be posted as events.

- *No unconscious actions*:

$$done(a) \Rightarrow (\mathsf{Bel}\ done(a))$$

  If an agent does some action, then it is aware that it has done the action.

  Operationalized in the *execute* function.

  A stronger requirement would be for the success or failure of the action to be posted.

- *No infinite deferral*:

$$(\mathsf{Int}\ \phi) \Rightarrow \mathsf{A}\Diamond(\neg(\mathsf{Int}\ \phi))$$

  An agent will eventually either act for an intention, or else drop it.






## 12 Summary

- This lecture has covered a lot of groung on practical reasoning.
- We started by discussing what practcial reasoning was, and how it relates to intentions.
- We then looked at planning (how an agent achieves its desires) and how deliberation and means-ends reasoning fit into the basic agent control loop.
- We then refined the agent control loop, considering commitment and intention reconsideration.
- Finally, we looked at some implmented systems, and discussed the connection between practical reasoning and logics of belief, desire and intention.