LECTURE 3b: Practical Reasoning Agents

Introduction to Multi-Agent Systems (MESIIA, MIA)

URV

Classes of Architecture

- 1956–present: Symbolic Reasoning Agents
  - Agents make decisions about what to do via symbol manipulation.
  - Its purest expression, proposes that agents use explicit logical reasoning in order to decide what to do.

- 1985–present: Reactive Agents
  - Problems with symbolic reasoning led to a reaction against this
  - led to the reactive agents movement

- 1990-present: Hybrid Agents
  - Hybrid architectures attempt to combine the best of reasoning and reactive architectures

What is Practical Reasoning?

- Practical reasoning is reasoning directed towards actions — the process of figuring out what to do:

Bratman (1990)

"… 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 gent believes…"

- 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 output of deliberation is *intentions*;
  - Means-ends reasoning
    - Deciding *how* to achieve these state of affairs
    - The outputs of this step are *plans*

- Intentions are a key part of this.
  - The interaction between *beliefs*, *desires* and *intentions* defines how the model works.

Intentions in Practical Reasoning

- Future-directed intentions
  - Intentions that an agent has towards some future state affairs.
- Intentions are:
  - *Pro-attitudes*, they tend to lead to action (drive means-ends reasoning).
  - *Persist*, attempt to achieve it (but not for too long).
  - *Consistent*/ constrain the future deliberation while holding some particular intentions, no options that are inconsistent with the intentions can be considered.
  - Influence beliefs upon with future practical reasoning is based.

Intentions in Practical Reasoning

1. Intentions pose problems for agents, who need to determine ways of achieving them. $\text{Type equation here.}$

   If I have an intention to ϕ, you would expect me to devote resources to deciding how to bring about ϕ

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

   If I have an intention to ϕ, you would not expect me to adopt an intention ψ that was incompatible with ϕ.

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

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

Intentions in Practical Reasoning

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 ϕ if I believed I would fail with ϕ.

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

   If I intend ϕ, then I believe that under "normal circumstances" I will succeed with ϕ.

Intentions in Practical Reasoning

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

   f I believe $\phi \Rightarrow \psi$ and I intend that ϕ, I do not necessarily intend ψ also.

8. Intentions are not closed under implication.

   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!

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;
    - (representation of) the environment.
  - and have it generate a *plan* to achieve the goal.

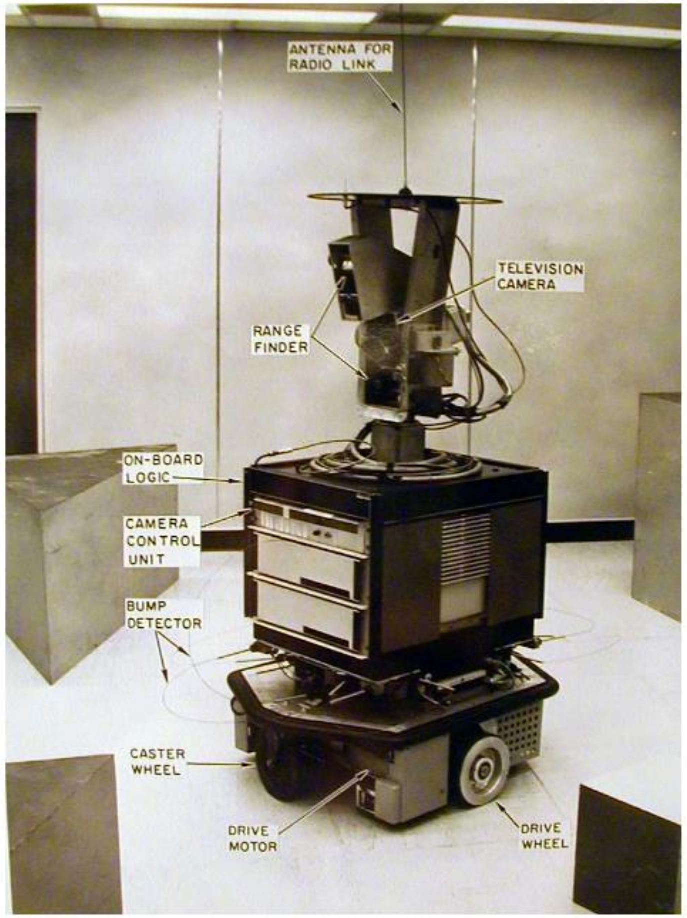
- This is ***automatic programming.***

Means-ends Reasoning/Planning

- Don't have to directly tell the system what to do!
    - Let it *figure out* how to achieve the goal on its own.

STRIPS Planner

- The **S**tanford **R**esearch **I**nstitute **P**roblem **S**olver
- Used by Shakey, the robot developed by Richard Fikes and Nils Nilsson in 1971 at SRI International.



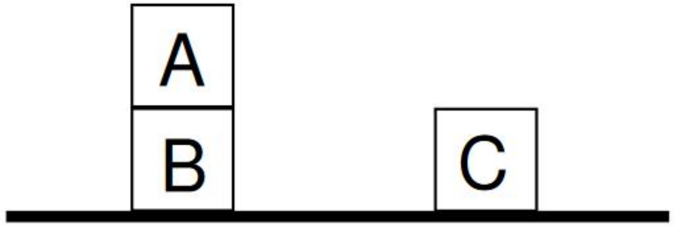
Representations

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

- Answer: We use *logic*, or something that looks a lot like logic.

Blocksworld

- We’ll illustrate the techniques with reference to the blocks world.
  - A simple (toy) world, in this case one where we consider toys.
- The blocks world contains a robot arm, 3 blocks (A, B and C) of equal size, and a table-top.
- The aim is to generate a plan for the robot arm to build towers out of blocks.



Blocksworld

- The environment is represented by an *ontology*.
- The closed world assumption is used
  - Anything not stated is assumed to be false.
- A *goal* is represented as a set of formulae

Blocksworld Ontology

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

Representation of the following blocks

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

The goal:

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

Blocksworld Actions

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

Blocksworld Actions

$Stack(x, y)$

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

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

$Pickup(x)$

pre	$Clear(x) \wedge OnTable(x) \wedge ArmEmpty$
del	$OnTable(x) \wedge ArmEmpty$
add	$Holding(x)$

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

$UnStack(x, y)$

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

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

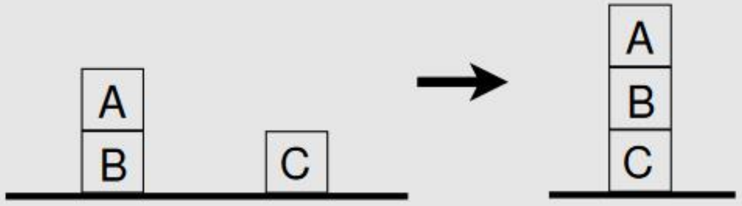
$PutDown(x)$

pre	$Holding(x)$
del	$Holding(x)$
add	$OnTable(x) \wedge ArmEmpty \wedge Clear(x)$

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

Using Plans

To get from here (left) to here (right)...



...we need this set of actions:

UnStack(A,B)

Putdown(A)

Pickup(B)

Stack(B, C)

Pickup(A)

Stack(A, B)

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

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

	$Pickup(x)$
pre	$Clear(x) \wedge OnTable(x) \wedge ArmEmpty$
del	$OnTable(x) \wedge ArmEmpty$
add	$Holding(x)$

	$PutDown(x)$
pre	$Holding(x)$
del	$Holding(x)$
add	$OnTable(x) \wedge ArmEmpty \wedge Clear(x)$

Plan Validity

- Thus, a plan is simply a sequence of steps
- However, how can we:
  - Generate the plan?
  - Ensure that it is correct?

Formal Representation

- As before we assume that the agent has a set of actions Ac, and we will write individual actions as α_1, α_2 and so on.
- Now, we define a new component called action *descriptor*.

Formal Representation

- A descriptor for an action $\alpha \in Ac$ is a triple $\langle P_\alpha, D_\alpha, A_\alpha \rangle$ where
  - P_α is the set of formulae of first-order logic that describes the *preconditions* of action α.
  - D_α is the set of formulae of first-order logic thar represents those facts will be *false* by applying the action α (the *delete* list).
  - A_α is the set of formulae of first-order logic thar represents those facts will be *true* by applying the action α(t he *add* list).

- For simplicity, we assume these lists constrained to only contain *ground atoms*.

Formal Representation

- A *planning problem* is determined by a triple $\langle \Delta, \mathrm{O}, \gamma \rangle$ where
  - Δ is the beliefs of the agent about the *initial* state of the world.
  - $\mathrm{O} = \{\langle P_\alpha, D_\alpha, A_\alpha \rangle \mid \alpha \in Ac\}$ is an indexed set of operator descriptors, one for each available action \alpha.
  - γ is a set of formulae of first-order logic, representing the *goal/task/intention* to be achieved.

- So, a *plan* π is a sequence of actions $\pi = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ where α_i is a member of Ac.

Formal Representation

- With respect to a planning problem $\langle \Delta, O, \gamma \rangle$ a plan $\pi = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ determines a sequence of $n + 1$ belief databases $\Delta_0, \Delta_1, \ldots, \Delta_n$, where $\Delta_0 = \Delta$, and

$$\Delta_i = \left(\Delta_{i-1} \backslash \mathrm{D}_{\alpha_i}\right) \cup A_{\alpha_i} \text{ for } 1 \leq i \leq n$$

Formal Representation

- Acceptable Plan:

  A (linear) plan $\pi = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ is said to be acceptable w.r.t the problem $\langle \Delta, O, \gamma \rangle$ iff the preconditions of every action is satisfied in preceding belief database, i.e., if $\Delta_{i-1} \vDash P_{\alpha_i}$, for all $1 \leq i \leq n$.

- Correct Plan:

  A plan $\pi = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ is correct w.r.t $\langle \Delta, O, \gamma \rangle$iff:

  1. It is acceptable
  2. $\Delta_n \vDash \gamma$ (i.e., if the goal is achieved in the final belief database generated by π)

Action Definitions are important!!!

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

	UnStack(x, y)
pre	$On(x, y)$ & $Clear(x)$ & $ArmEmpty$
del	$On(x, y)$& $ArmEmpty$
add	$Holding(x)$ & $Clear(y)$

	Pickup(x)
pre	$Clear(x)$ & $OnTable(x)$ & $ArmEmpty$
del	$OnTable(x)$ & $ArmEmpty$
add	$Holding(x)$

	Release(x)
pre	$Holding(x)$
del	$Holding(x)$
add	$OnTable(x)$ & $ArmEmpty$

One of these definitions works. The other doesn't!

	Grap(x)
pre	$Clear(x)$ & $OnTable(x)$ & $ArmEmpty$
del	$OnTable(x)$ & $ArmEmpty$
add	$Holding(x)$

	Build(x, y)
pre	$Clear(y)$ & $Holding(x)$
del	$Clear(y)$ & $Holding(x)$
add	$ArmEmpty$ & $On(x, y)$

	Drop(y)
pre	$Holding(y)$
del	$Holding(y)$
add	$OnTable(y)$ & $ArmEmpty$ & $Clear(y)$

	Demolish(x, y)
pre	$On(x, y)$ & $Clear(y)$ & $ArmEmpty$
del	$On(x, y)$ & $ArmEmpty$
add	$Holding(x)$ & $Clear(y)$

Action Definitions are important!!!

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

	UnStack(x, y)
pre	$On(x, y)$ & $Clear(x)$ & $ArmEmpty$
del	$On(x, y)$& $ArmEmpty$
add	$Holding(x)$ & $Clear(y)$

	Pickup(x)
pre	$Clear(x)$ & $OnTable(x)$ & $ArmEmpty$
del	$OnTable(x)$ & $ArmEmpty$
add	$Holding(x)$

	Release(x)
pre	$Holding(x)$
del	$Holding(x)$
add	$OnTable(x)$ & $ArmEmpty$ & $Clear(x)$

One of these definitions works. The other doesn't!

	Grap(x)
pre	$Clear(x)$ & $OnTable(x)$ & $ArmEmpty$
del	$OnTable(x)$ & $ArmEmpty$
add	$Holding(x)$

	Build(x, y)
pre	$Clear(y)$ & $Holding(x)$
del	$Clear(y)$ & $Holding(x)$
add	$ArmEmpty$ & $On(x, y)$

	Drop(y)
pre	$Holding(y)$
del	$Holding(y)$
add	$OnTable(y)$ & $ArmEmpty$ & $Clear(y)$

	Demolish(x, y)
pre	$On(x, y)$ & $Clear(y)$ & $ArmEmpty$
del	$On(x, y)$ & $ArmEmpty$
add	$Holding(x)$ & $Clear(y)$

Implementing Practical Reasoning Agents

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

- For now, we will not be concerned with stages 2 or 3.
  - These are related to the functions ***see*** and ***next***

```
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
```

Implementing Practical Reasoning Agents

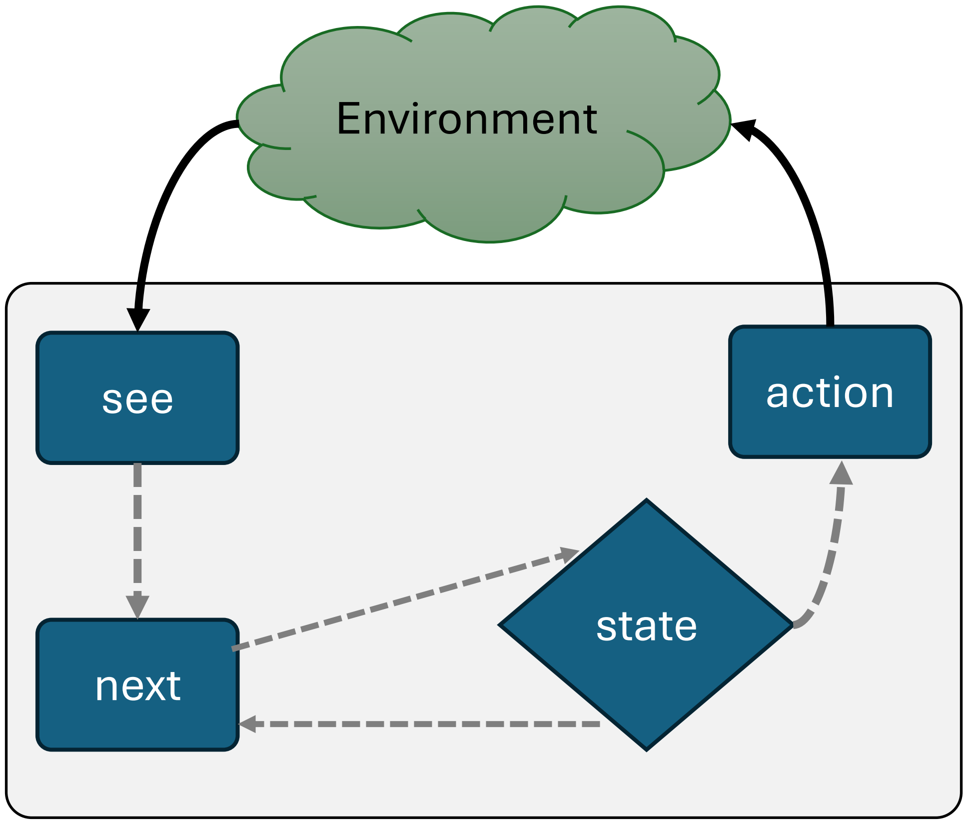
- see is as before

$$see: E \rightarrow Per$$

- Instead of the function next...
  - which takes a percept and used it to update the internal state of an agent
- ...we have a belief revision function:

$$brf: 2^{Bel} \times Per \rightarrow 2^{Bel}$$

Where, Bel is the set of all possible beliefs that an agent might have



Implementing Practical Reasoning Agents

- Problem:
  - deliberation and means-ends reasoning processes are not *instantaneous*.
  - They have a *time cost*.
- Suppose that deliberation is *optimal*
  - The agent selects the optimal intention to achieve, then this is the best thing for the agent.
  - i.e., it maximises the expected utility.
- So, the agent 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* in the meantime.
  - Even if the agent can compute the right thing to do, it may not do the right thing.
  - Optimality is hard.

Implementing Practical Reasoning Agents

- Let's make the algorithm more formal
  - The term $I \subseteq Int$, the set of intentions
  - Plan() is the planning function,
  - brf() is s the belief revision function
  - and execute() is a function that executes each action in a plan.

- How might we implement these functions?

```
Agent Control Loop Version 2
1.   B := B_0; /* initial beliefs */
2.   while true do
3.        get next percept ρ;
4.        B := brf(B, ρ);
5.        I := deliberate(B);
6.        π := plan(B, I);
7.        execute(π)
8.   end while
```

Deliberation

- How does an agent *deliberate*?
  - begin by trying to understand what the *options* available to it 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***; and
  - ***filtering***

Option Generation and Filtering

- Option Generation
  - 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***

$$options : \mathcal{P}(Bel) \times \mathcal{P}(Int) \rightarrow \mathcal{P}(Des)$$

- Filtering
  - the agent chooses between competing alternatives, and commits to achieving them.
  - In order to select between competing options, an agent uses a **filter**() function.
  - ***Intentions***

$$filter : \mathcal{P}(Bel) \times \mathcal{P}(Des) \times \mathcal{P}(Int) \rightarrow \mathcal{P}(Int)$$

Implementing Practical Reasoning Agents

Agent Control Loop Version 3

$B := B_0$;
$I := I_0$;
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

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.

- Open-minded commitment
  - An open-minded agent will maintain an intention *as long as it is still believed* possible.

Degrees of Commitment

- 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.
  - However, to write the algorithm

  down we *need to refine* our notion of plan execution.

Degrees of Commitment

- The previous version was *blindly committed to its means and its ends*

- If π is a plan, then:
  - $empty(\pi)$ is true if there are no more actions in the plan.
  - $hd(\pi)$ returns the first action in the plan.
  - $tail(\pi)$ returns the plan minus the head of the plan.
  - $sound(\pi, I, B)$ means that π is sound plan for I given B.

```
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);
    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
```

Degrees of Commitment

- Makes the control loop more reactive, able to change intention when the world changes.
  - i.e. it is not committed to its means (line 16)

- Still overcommitted to intentions (ends).
  - Never stops to consider whether or not its intentions are appropriate.

Agent Control Loop Version 4

```
1.  B := B_0;
2.  I := I_0;
3.  while true do
4.      get next percept ρ;
5.      B := brf(B, ρ);
6.      D := options(B, I);
7.      I := filter(B, D, I);
8.      π := plan(B, I);
9.      while not empty(π) do
10.         α := hd(π);
11.         execute(α);
12.         π := tail(π);
13.         get next percept ρ;
14.         B := brf(B, ρ);
15.         if not sound(π, I, B) then
16.             π := plan(B, I)
17.         end-if
18.     end-while
19. end-while
```

Single Minded Commitment

- Modification:
  - stop to determine whether intentions have succeeded or whether they are impossible

- Our agent now gets to reconsider its intentions once every time around the outer control loop (line 9), i.e., after:
  - 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

Agent Control Loop Version 5

1. $B := B_0$;
2. $I := I_0$;
3. while true do
4.   get next percept ρ;
5. $B := brf(B, \rho)$;
6. $D := options(B, I)$;
7. $I := filter(B, D, I)$;
8. $\pi := plan(B, I)$;
9.   while not($empty(\pi)$ or $succeeded(I, B)$ or $impossible(I, B)$) do
10. $\alpha := hd(\pi)$;
11. $execute(\alpha)$;
12. $\pi := tail(\pi)$;
13.   get next percept ρ;
14. $B := brf(B, \rho)$;
15.   if not $sound(\pi, I, B)$ then
16. $\pi := plan(B, I)$
17.   end-if
18.   end-while
19. end-while

Open Minded Commitment

- In the previous version, our agent reconsiders its intentions once every time around the outer control loop.
- In this new version, our agent also reconsiders its intentions after every action (lines 15 & 16)
- But this intention reconsideration is ***costly***!

Agent Control Loop Version 6

1. $B := B_0$;
2. $I := I_0$;
3. while true do
4.   get next percept ρ;
5. $B := brf(B, \rho)$;
6. $D := options(B, I)$;
7. $I := filter(B, D, I)$;
8. $\pi := plan(B, I)$;
9.   while not $(empty(\pi)$ or $succeeded(I, B)$ or $impossible(I, B))$ do
10. $\alpha := hd(\pi)$;
11. $execute(\alpha)$;
12. $\pi := tail(\pi)$;
13.   get next percept ρ;
14. $B := brf(B, \rho)$;
15. $D := options(B, I)$;
16. $I := filter(B, D, I)$;
17.   if not $sound(\pi, I, B)$ then
18. $\pi := plan(B, I)$
19.   end-if
20.   end-while
21. end-while

Intention Reconsideration

- A dilemma:
  - an agent that *does not stop* to reconsider its intentions sufficiently often *will continue to attempt* 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

```
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);
    while not(empty(π)
            or succeeded(I, B)
            or impossible(I, B)) do
        α := hd(π);
        execute(α);
        π := tail(π);
        get next percept ρ;
        B := brf(B, ρ);
        if reconsider(I, B) then
            D := options(B, I);
            I := filter(B, D, I);
        end-if
        if not sound(π, I, B) then
            π := plan(B, I)
        end-if
    end-while
end-while
```

Intention Reconsideration

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

Situation number	Chose to deliberate?	Changed intentions?	Would have changed intentions?	*reconsider*(...) optimal?
1	No	—	No	Yes
2	No	—	Yes	No
3	Yes	No	—	No
4	Yes	Yes	—	Yes

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

Optimal Intention Reconsideration

- Two different types of reconsideration strategy considered:
  - bold agents: never pause to reconsider intentions, and
  - cautious agents: stop to reconsider after every action.

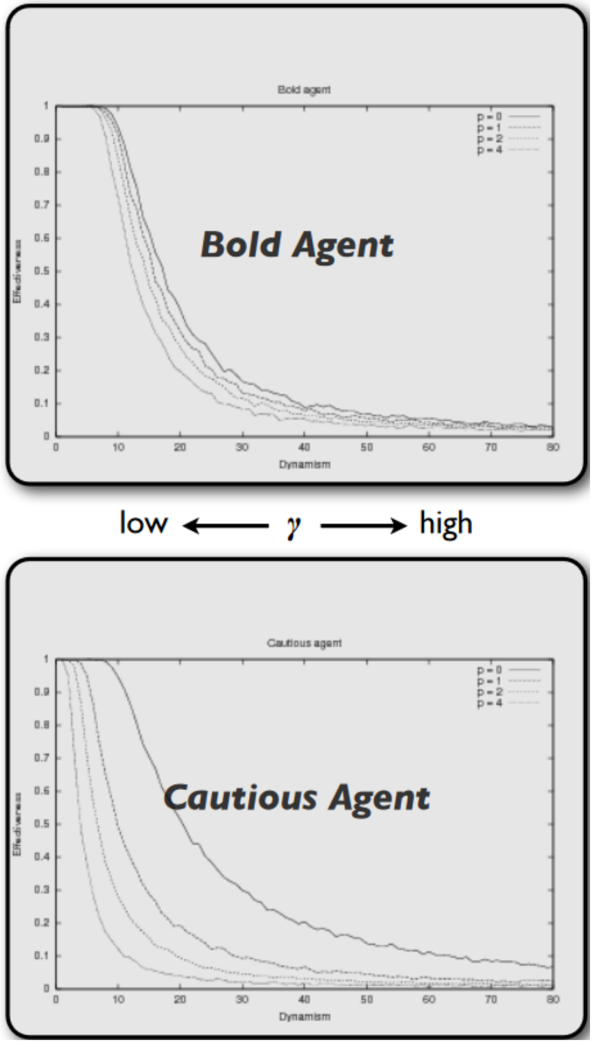
- Dynamism in the environment is represented by the rate of world change, γ.

Optimal Intention Reconsideration

- if γ 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.

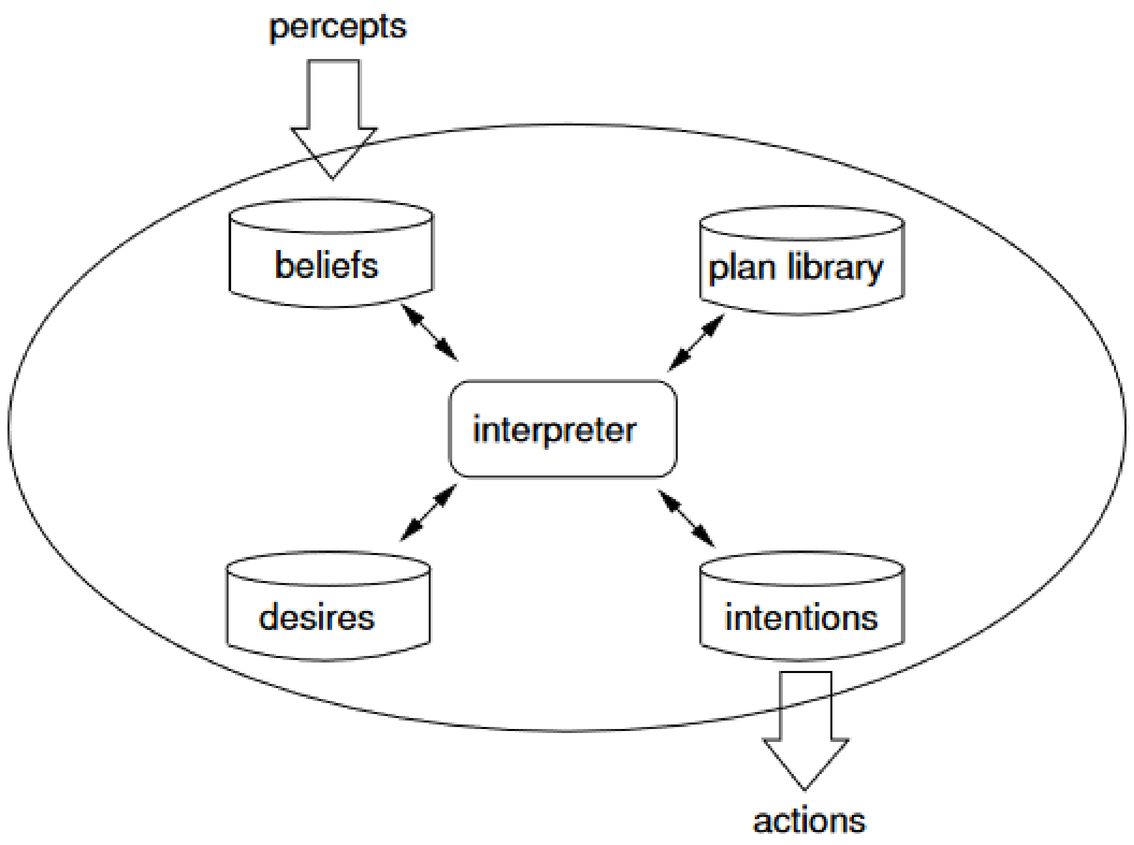
- If γ is high (i.e., the environment changes frequently), then cautious agents can outperform bold agents.
    - This is because they are able to recognise when intentions are doomed, and also to take advantage of serendipitous situations and new opportunities when they arise.

- When planning costs are high, this advantage can be eroded.



Implemented BDI Agents: Procedural Reasoning System (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 orderto 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.



Readings for this week

- M.Wooldridge: An introduction to MultiAgent Systems – Ch. 4 Practical Reasoning Agents