3. Reasoning in Agents
Part 2:
BDI Agents

Javier Vázquez-Salceda
SMA-UPC

Practical Reasoning

- •Introduction to Practical Reasoning
- •Planning

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

Practical Reasoning

Intentions

1. Intentions pose problems for agents, who need to determine ways of achieving them.
   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 such that ϕ and ψ are mutually exclusive.
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 ϕ fails, then all other things being equal, it will try an alternative plan to achieve ϕ.
4. Agents believe their intentions are possible.
   That is, they believe there is at least some way that the intentions could be brought about.

Practical Reasoning

Intentions

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 ϕ was not possible.
6. Under certain circumstances, agents believe they will bring about their intentions.
It would not normally be rational of me to believe that I would *bring my intentions about; intentions can* fail*. Moreover, it does not make sense that if I believe ϕ is inevitable that I would adopt it as an intention.*
7. Agents need not intend all the expected side effects of their intentions.
If I believe $\phi \rightarrow \psi$ and I intend that ϕ, I do not necessarily intend ψ 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!

Practical Reasoning

Intentions vs Desires

- 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)

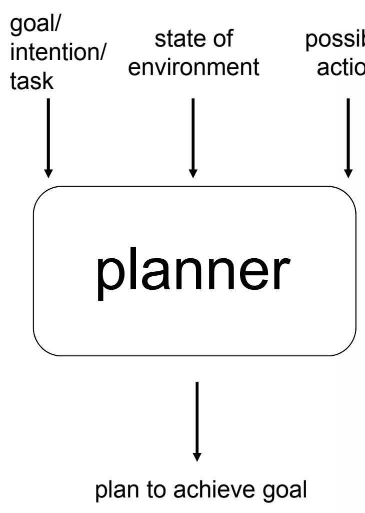
Means-End Reasoning

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
- Basic idea is to give a planning agent:
  - representation of goal/intention to achieve
  - representation actions it can perform
  - representation of the environment
  - and have it generate a *plan* to achieve the goal

Means-End Reasoning: Planning

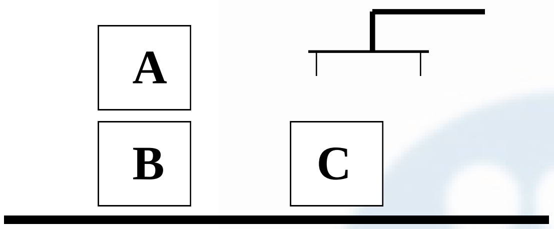
Planners



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

Means-End Reasoning: Planning

The Blocks World (I)



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

Means-End Reasoning: Planning

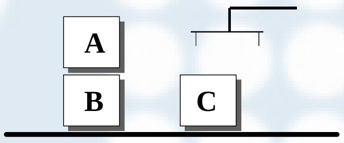
The Blocks World (II)

- 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 configuration shown before:

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

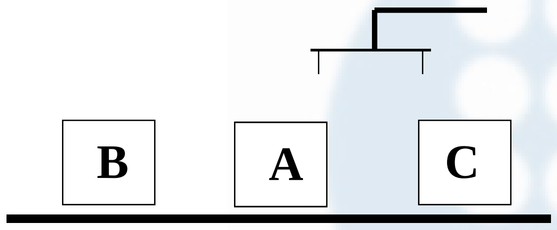


- Use the *closed world assumption*: anything not stated is assumed to be *false*

Means-End Reasoning: Planning
The Blocks World (III)

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

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

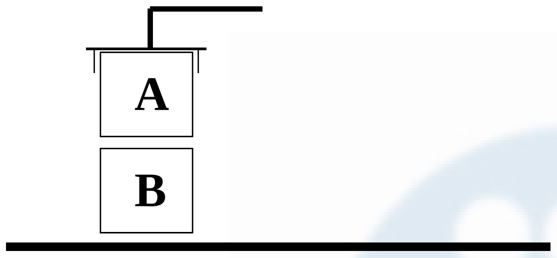


Means-End Reasoning: Planning
The Blocks World (IV)

- *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*

Means-End Reasoning: Planning
The Blocks World (V)



- Example 1:
  The *stack* action occurs when the robot arm places the object x it is holding on top of object y.

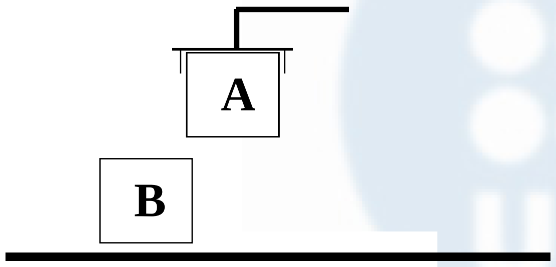
	$Stack(x, y)$
pre	$Clear(y) \wedge Holding(x)$
del	$Clear(y) \wedge Holding(x)$
add	$ArmEmpty \wedge On(x, y)$

Means-End Reasoning: Planning
The Blocks World (VI)

- Example 2:
  The *unstack* action occurs when the robot arm picks an object x up from on top of another object y.

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

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



Means-End Reasoning: Planning
The Blocks World (VII)

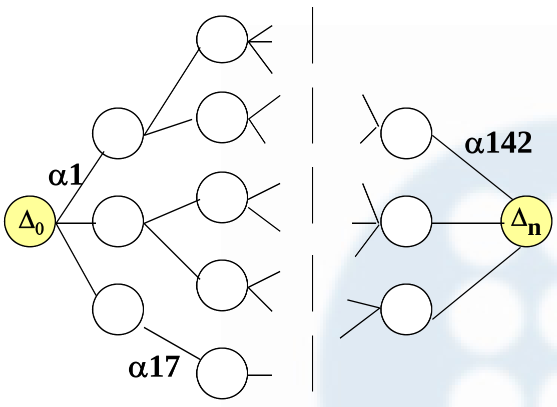
- Example 3:
  The *pickup* action occurs when the arm picks up an object x from the table.

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

- Example 4:
  The *putdown* action occurs when the arm places the object x onto the table.

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

Means-End Reasoning: Planning
Planning Theory (I)



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

Means-End Reasoning: Planning

Planning Theory (II)

- $Ac = \{\alpha_1, \ldots, \alpha_n\}$: a fixed set of actions.
- $< P_\alpha, D_\alpha, A_\alpha >$ a descriptor for an action $\alpha \in Ac$
  - P_α is a set of formulae of first-order logic that characterise the *precondition* of action α
  - D_α is a set of formulae of first-order logic that characterise those *facts* made false by the performance of α (the delete list)
  - A_α is a set of formulae of first-order that characterise those facts made *true* by the performance of α (the add list)
- A **planning problem** is a triple $<\Delta, O, \gamma>$

Means-End Reasoning: Planning

Planning Theory (III)

- $\pi=(\alpha_1, \ldots, \alpha_n)$: a plan with respect to a planning problem $<\Delta, O, \gamma>$ determines a sequence of $n+1$ models:

$$\Delta_0, \Delta_1, \ldots, \Delta_n$$

- where $\Delta_0 = \Delta$ and

$$\Delta_i = (\Delta_{i-1} \setminus D_{\alpha_i}) \cup A_{\alpha_i} \quad \text{for} 1 \leq i \leq n$$

- A plan π is acceptable iff $\Delta_{i-1} \vdash P_{\alpha_i}$,for all $1 \leq i \leq n$
- A plan π is correct iff
  - π is acceptable, and
  - $\Delta_n \vdash \gamma$

Means-End Reasoning: Planning
Limitations

- 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**
- **However, planning technology has evolved a lot in the last decade, and there are practical planners that are being used in time-constrained systems! (especially in the game industry)**
  - **New heuristics to reduce the search space**
  - **Minor simplifications or restrictions to expresiveness to keep it within computable bounds**

Implementing Practical Reasoning Agents

•Agent Control Loop
•BDI Theory
•Implemented BDI agents

Implementing Practical Reasoning Agents

Agent Control Loop Version 1

- A first step 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

Agent Control Loop Version 1

- 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. (Maximizes expected utility.)
- So in step 4 the agent has selected an intention to achieve that would have been optimal *if it had been achieved at the time it observed the world (step 2).*
  - But unless *deliberation time* (time between steps 2 and 4) is really small, then the agent runs the risk that the intention selected is no longer optimal by the time the agent has fixed upon it.
  - This is *calculative rationality*.
- Deliberation is only half of the problem: the agent still has to determine *how* to achieve the intention.

Implementing Practical Reasoning Agents
Agent Control Loop Version 1

- So, this agent will have overall optimal behaviour in the following circumstances:
  1. When deliberation and means-ends reasoning take a vanishingly small amount of time; or
  2. When the world is guaranteed to remain static while the agent is deliberating and performing means-ends reasoning,
     - the assumptions upon which the choice of intention to achieve and plan to achieve the intention remain valid until the agent has completed deliberation and means-ends reasoning; or
  3. When an intention that is optimal remains optimal until the agent has found a way to achieving it.

Implementing Practical Reasoning Agents
Agent Control Loop Version 2

- 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.      $B := brf(B, \rho)$;
5.      $I := deliberate(B)$;
6.      $\pi := plan(B, I)$;
7.      $execute(\pi)$
8.  end while
```

Implementing Practical Reasoning Agents

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;
    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.

Implementing Practical Reasoning Agents

Agent Control Loop Version 3

```
Agent Control Loop Version 3
1.
2.   $B := B_0$;
3.   $I := I_0$;
4.   while true do
5.       get next percept $\rho$;
6.       $B := brf(B, \rho)$;
7.       $D := options(B, I)$;
8.       $I := filter(B, D, I)$;
9.       $\pi := plan(B, I)$;
10.      $execute(\pi)$
11.  end while
```

Implementing Practical Reasoning Agents

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.

This time, you angrily return Willie for **overcommitment.** 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. Willie again accedes, saying *"Yes, Sir."* (Its attitude problem seems to have been fixed.) 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."

Implementing Practical Reasoning Agents

Commitment Strategies

- The following *commitment strategies* are commonly discussed in the literature of rational agents:
  - *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.
- An agent has commitment both to *ends* (i.e., the 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 $\rho$;
      $B := brf(B, \rho)$;
      $D := options(B, I)$;
      $I := filter(B, D, I)$;
      $\pi := plan(B, I)$;
      while not $empty(\pi)$ 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
```

Implementing Practical Reasoning Agents

Agent Control Loop Version 4

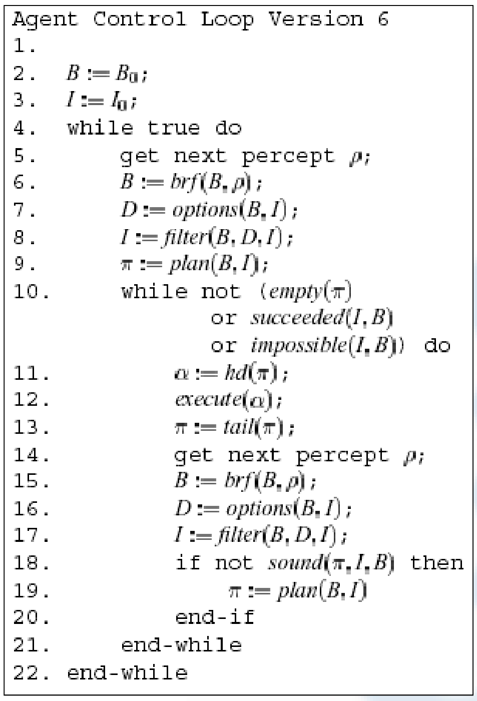
- Still overcommitted to intentions: Never stops to consider whether or not its intentions are appropriate
- Modification: stop to determine whether intentions have succeeded or whether they are impossible: (*Single-minded commitment*)

```
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
```

Implementing Practical Reasoning Agents

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

Implementing Practical Reasoning Agents

Intention Reconsideration

- 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

    $B := B_0$;
    $I := I_0$;
    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 $reconsider(I, B)$ then
                $D := options(B, I)$;
                $I := filter(B, D, I)$;
            end-if
            if not $sound(\pi, I, B)$ then
                $\pi := plan(B, I)$
            end-if
        end-while
    end-while
```

Implementing Practical Reasoning Agents

Intention Reconsideration: Meta-level control - deliberation

- 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

Implementing Practical Reasoning Agents

Intention Reconsideration: Meta-level control - deliberation

- In situation (1), the agent did not choose to deliberate, and as 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(…)* 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(…)* function is not behaving optimally.
- In situation (3), the agent chose to deliberate, but did not change intentions. In this situation, the *reconsider(…)* function is not behaving optimally.
- In situation (4), the agent chose to deliberate, and did change intentions. In this situation, the *reconsider(…)* function is behaving optimally.
- An important assumption: cost of *reconsider(…)* is *much* less than the cost of the deliberation process itself.

Implementing Practical Reasoning Agents

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*, γ
- Results (not surprising):
  - 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 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.

BDI Theory and 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 *labelled branching time* framework

BDI Logic

- From classical logic: $\wedge,\vee,\neg$, …
- The $CTL*$ *path quantifiers*:
  - $A\phi$ 'on all paths, ϕ'
  - $E\phi$ 'on some paths, ϕ'
- The BDI connectives:
  - $(Bel\ i\ \phi)$ i believes ϕ
  - $(Des\ i\ \phi)$ i desires ϕ
  - $(Int\ i\ \phi)$ i intends ϕ

BDI Logic

- Semantics of BDI 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. . .)

BDI Logic

Axioms of KD45

- (1) $\mathbf{Bel}(p \rightarrow q) \rightarrow (\mathbf{Bel}\ p \rightarrow \mathbf{Bel}\ q)$ (K)

If you believe that p implies q then if you believe p then you believe q

- (2) $\mathbf{Bel}\ p \rightarrow \neg\mathbf{Bel}\ \neg p$ (D)

This is the consistency axiom, stating that if you believe p then you do not believe that p is false

- (3) $\mathbf{Bel}\ p \rightarrow \mathbf{Bel}\ \mathbf{Bel}\ p$ (4)

If you believe p then you believe that you believe p

- (4) $\neg\mathbf{Bel}\ p \rightarrow \mathbf{Bel}\ \neg\mathbf{Bel}\ p$ (5)

If you do not believe p then you believe that you do not believe that p is true

It also entails the two inference rules of *modus ponens* and necessitation:

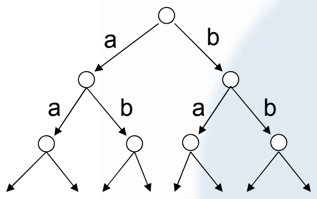
- (5) if p, and $p \rightarrow q$, then q (MP)
- (6) if p is a theorem of KD45 then so is $\mathbf{Bel}\ p$ (Nec)

This last rule states that you believe all theorems implied by the logic

BDI Logic

Temporal Logic: CTL*

- Branching time logic views a computation as a (possibly infinite) tree or DAG of states connected by atomic events
- At each state the outgoing arcs represent the actions leading to the possible next states in some execution



- Variant of branching time logic that we look at is called CTL*, for Computational Tree Logic (star)

BDI Logic

Temporal Logic: CTL* Notation

- In this logic
  - A = "for every path“
  - E = "there exists a path“
  - G = “globally” (similar to □)
  - F = “future” (similar to ◊)
- A and E refer to paths
  - A requires that all paths have some property
  - E requires that at least some path has the property
- G and F refer to states on a path
  - G requires that all states on the given path have some property
  - F requires that at least one state on the path has the property

BDI Logic

Temporal Logic: CTL* Examples

- *AG p*
  - For every computation (i.e., path from the root), in every state, *p* is true
  - Hence, means the same as □p
- *EG p*
  - There exists a computation (path) for which *p* is always true
- *AF p*
  - For every path, eventually state *p* is true
  - Hence, means the same as ◊ *p*
  - Therefore, *p* is *inevitable*
- *EF p*
  - There is some path for which p is eventually true
  - I.e., *p* is "reachable"
  - Therefore, *p* will hold *potentially*

BDI Logic

Axioms

- *Belief goal compatibility*:

  $$(Des\ \alpha) \rightarrow (Bel\ \alpha)$$

  States that if the agent has a goal to optionally achieve something, this thing must be an option.
  This axiom is operationalised in the function *options*: an option should not be produced if it is not believed possible.
- *Goal-intention compatibility*:

  $$(Int\ \alpha) \rightarrow (Des\ \alpha)$$

  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).
  Operationalised in the *deliberate* function.
- *Volitional commitment*:

  $$(Int\ does(a)) \rightarrow does(a)$$

  If you intend to perform some action *a* next, then you do *a* next.
  Operationalised in the *execute* function.

BDI Logic

Axioms

- *Awareness of goals & intentions*:

  $$(Des\ \phi) \rightarrow (Bel\ (Des\ \phi))$$
  $$(Int\ \phi) \rightarrow (Bel\ (Int\ \phi))$$

  Requires that new intentions and goals be posted as events.

- *No unconscious actions*:

  $$done(a) \rightarrow Bel\ (done(a))$$

  If an agent does some action, then it is aware that it has done the action.
  Operationalised in the *execute* function.
  A stronger requirement would be for the success or failure of the action to be posted.

- *No infinite deferral*:

  $$(Int\ \phi) \rightarrow \mathrm{A}\Diamond(\neg(Int\ \phi))$$

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

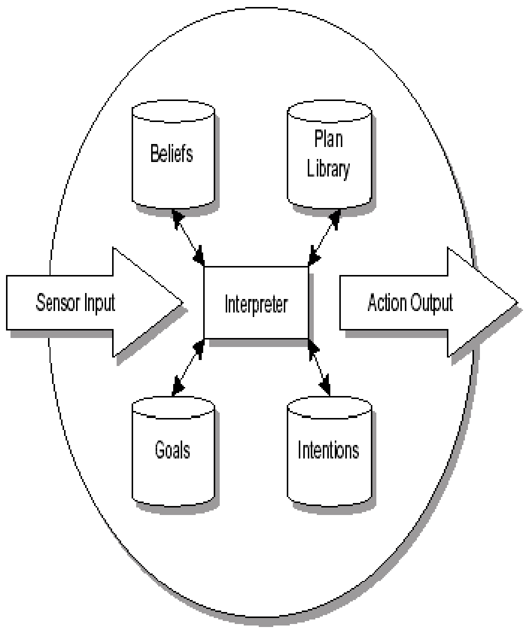
Implemented BDI Agents

- **PRS**
- **IRMA**

Implemented BDI Agents
PRS

- A BDI-based agent architecture: the PRS – Procedural Reasoning System (Georgeff, Lansky)
- 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 realize 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

PRS



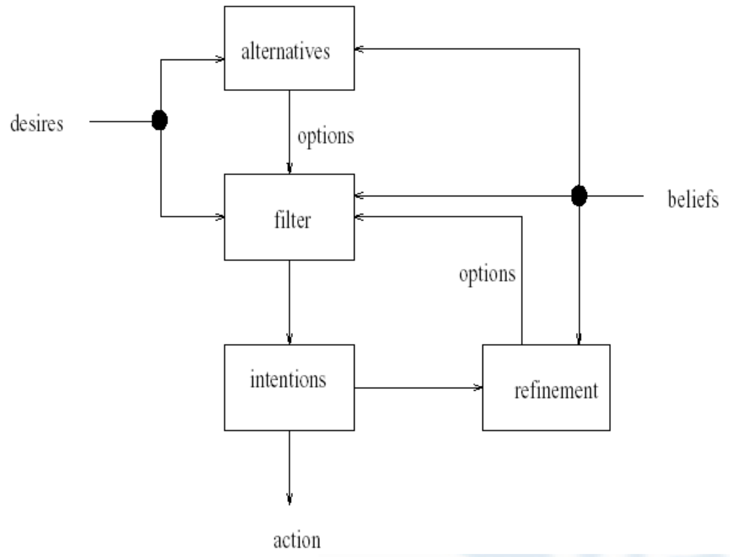
Implemented BDI Agents
IRMA

- IRMA – Intelligent Resource-bounded Machine Architecture – Bratman, Israel, Pollack
- IRMA has four key symbolic data structures:
  - a *plan library*
  - explicit representations of
    - *beliefs*: information available to the agent — may be represented symbolically, but may be simple 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*

Implemented BDI Agents
IRMA

- 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
  - a *deliberation process* responsible for deciding upon the ‘best’ intentions to adopt

Implemented BDI Agents
IRMA



Implemented BDI agents
Other implementations

- AGENTSPEAK
- ARTS
- dMARS
- **JADEx**
- **JASON**
- JACK Intelligent Agents
- SPARK
- **2APL**
- 3APL

References

[1] Wooldridge, M. “Introduction to Multiagent Systems”. John Wiley and Sons, 2002.

[2] Weiss, G. “Multiagent Systems: A modern Approach to Distributed Artificial Intelligence”. MIT Press. 1999. ISBN 0262-23203

[3] Y. Shoham, “An Overview of Agent-Oriented Programming”, in J. M. Bradshaw, editor, Software Agents, pages 271–290. AAAI Press / The MIT Press, 1997.