Multiagent Systems

3. Practical Reasoning Agents

B. Nebel, C. Becker-Asano, S. Wölfl

Albert-Ludwigs-Universität Freiburg

May 14, 2014

Background

Practical Reasoning I

Practical Reasoning is reasoning directed towards **actions**, i.e. deciding what to do.

Principles of practical reasoning applied to agents largely derive from work of philosopher **Michael 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 agent believes."
> (after Wooldridge, p. 65)

Fundamentally different from **theoretical reasoning**, which is concerned with **belief**, e.g. reasoning about a mathematical problem.

Practical Reasoning II

Most important ⇒ agent has to stop reasoning and **take action** in a timely fashion.

Practical reasoning is foundation for

Belief-Desire-Intention

model of agency.
It consists of two main activities:

1. Deliberation: deciding **what** to do
2. Means-ends reasoning: deciding **how** to do it

Combining them appropriately
⇒ foundation of deliberative agency

Deliberation & Means-ends reasoning

Deliberation:

- is concerned with determining `what one wants to achieve` (considering preferences, choosing goals, etc.)
- generates **intentions** (interface between deliberation and means-ends reasoning)

Means-ends reasoning:

- is used to determine how the goals are to be achieved by thinking about **suitable actions**, resources and how to "organize" activity
- generates **plans** which are turned into actions

Intentions I

Demarcation of the term “intentions”:

- In ordinary speech, intentions refer to actions or to states of mind; here we consider the latter.
- Our focus: **future-directed intentions** also called **pro-attitudes** that tend to lead to actions.
- We make **reasonable attempts** to fulfill intentions once we form them, but they may change if circumstances do.

Intentions II

Main properties of intentions:

- **Intentions drive means-ends reasoning**: If I adopt an intention I will attempt to achieve it, this affects action choice
- **Intentions persist**: Once adopted they will not be dropped until achieved, deemed unachievable, or reconsidered
- **Intentions constrain future deliberation**: Options inconsistent with intentions will not be entertained
- **Intentions influence beliefs concerning future practical reasoning**: Rationality requires that I believe I can achieve intention

Intentions: Bratman's model

Bratman's model suggests the following properties:

1. Intentions pose problems for agents, who need to determine ways of achieving them
2. Intentions provide a 'filter' for adopting other intentions, which must not conflict
3. Agents track the success of their intentions, and are inclined to try again if their attempts fail
4. Agents believe their intentions are possible
5. Agents do not believe they will not bring about their intentions
6. Under certain circumstances, agents believe they will bring about their intentions
7. Agents need not intend all the expected side effects of their intentions

Desires

Desires:

- describe the states of affairs that are considered for achievement, i.e. basic preferences of the agent.
- are much weaker than intentions, they are not directly related to activity:

"My desire to play basketball this afternoon is merely a potential influence 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, after Wooldridge, p. 67)

BDI Architecture

Background

BDI Architecture

Jason reasoning cycle

Perception

Planning

Action

Formal model of BDI

STRIPS

Blocks world

Formal model of Planning

General BDI control loop

Commitment

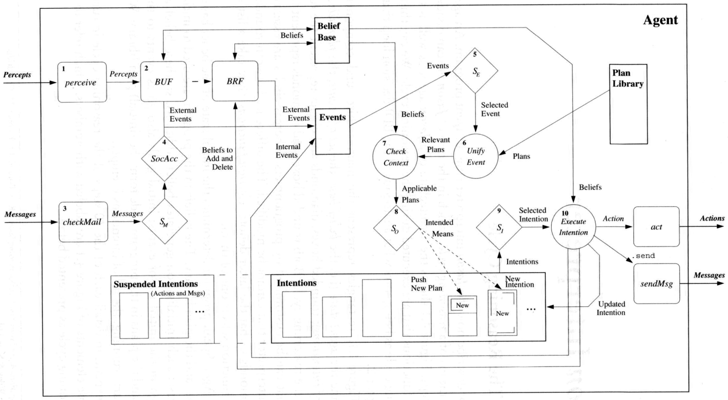
Summary

The BDI Architecture

Sub-components of overall BDI control flow:

- Belief revision function
  - Update beliefs with sensory input and previous belief
- Generate options
  - Use beliefs and existing intentions to generate a set of alternatives/options (=desires)
- Filtering function
  - Choose between competing alternatives and commit to their achievement
- Planning function
  - Given current belief and intentions generate plan for action
- Action generation: iteratively execute actions in plan sequence

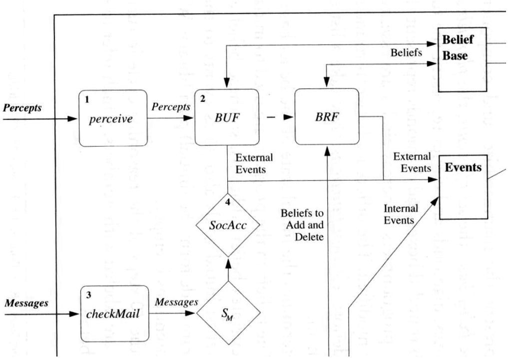
The Jason reasoning cycle



The Jason reasoning cycle; Bordini et al. (2007), p. 68

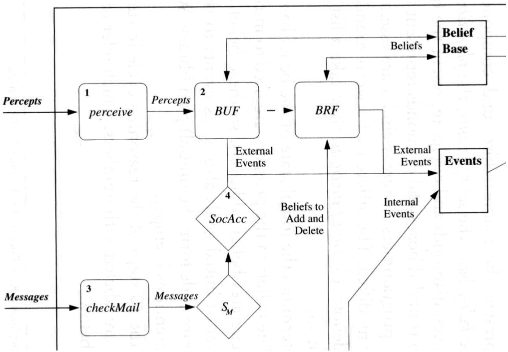
- Rounded boxes and diamonds can be customized (Java)
- Circles are essential parts of Jason ⇒ not modifiable

(1/2) Perception & Belief update



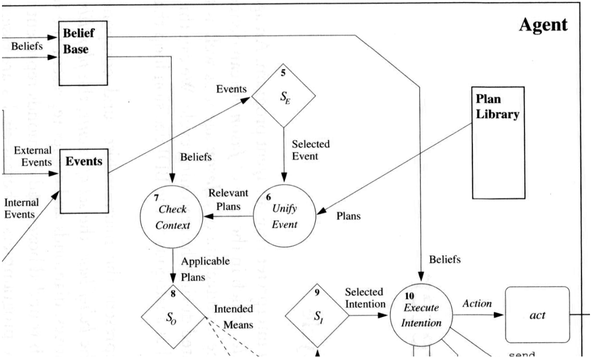
- Sense environment and update beliefs via Belief Update Function BUF
- **perceive** and **BUF** can be reprogrammed ⇒ interface to real world robots

(3/4) Messages & SocAcc



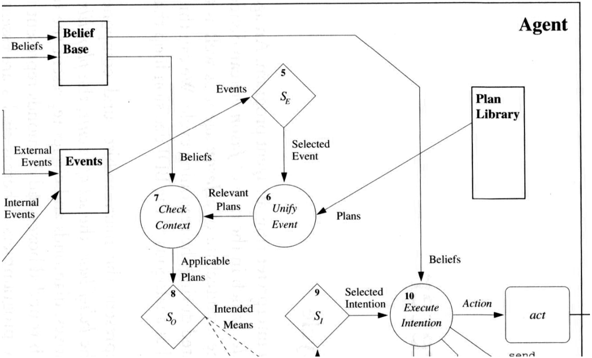
- Messages received via **checkMail** method
- Selecting 'Socially Acceptable' messages in **SocAcc** method ⇒ kind of a low-level "spam filter"

(5) Selecting an event



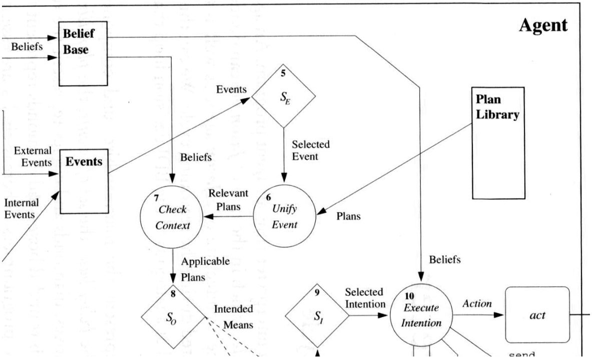
- **Events** represent either environment changes or internal changes (related to goals)
- Per reasoning cycle **only one** pending event is processed (FIFO principle in default implementation)
- Customize this to handle priorities

(6) Retrieving all relevant plans



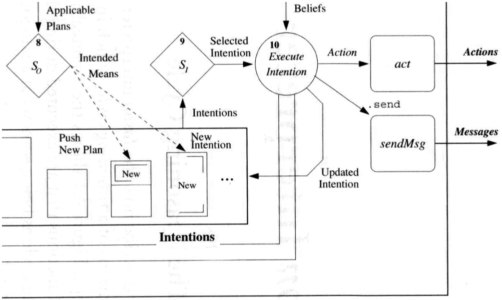
- Check **Plan Library** component for all relevant plans
- *Triggering event* of plan needs to **unify** with selected event
- Returns **set of relevant plans**

(7) Check plan contexts



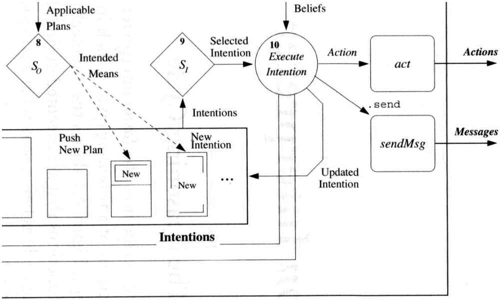
- Select from *relevant plans* those that are **applicable**
- Only true, when a plan's **context** is a *logical consequence* of the agent's **Belief Base**
- Returns **set of applicable plans**

(8) Selecting one applicable plan



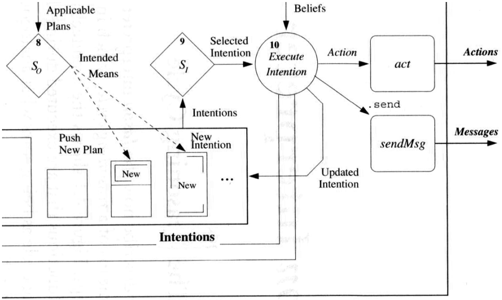
- Committing to a plan ⇒ forming an **intention**
- *Applicable plan selection function* $\mathcal{S}_{\mathcal{O}}$ can be customized
- Default function $\mathcal{S}_{\mathcal{O}}$ uses **first-come-first-selected** heuristics ⇒ depends on **order of plan definitions!!!**

(9) Selecting an intention



- Default *intention selection function* $\mathcal{S}_{\mathcal{I}}$ ⇒ **round-robin**
- Only **one action** of each intention is executed
- Select top-most intention, execute its first step, push it back to end of list (can be customized, of course)
- ⇒ **dividing attention equally** over *all* intentions

(10) Executing one step of an intention



- Intention is a stack of partially instantiated plans, e.g.:
  `[ +!g : true <- a2. | +b : true <- !g; a1. ]`
- Body of first plan is considered, here only a2
- First formula is *dealt with*, here action a2, and *deleted*
- Updated intention is pushed back to intention stack

The BDI architecture – formal model

- Let $B \subseteq Bel$, $D \subseteq Des$, $I \subseteq Int$ be sets describing **beliefs**, **desires**, and **intentions** of an agent
- Percepts Per and actions Ac as before
- $Plan$ set of all plans (for now: sequences of actions)

Model described through a set of abstract functions:

- Belief revision $brf : \mathcal{P}(Bel) \times Per \rightarrow \mathcal{P}(Bel)$
- Option generation $options : \mathcal{P}(Bel) \times \mathcal{P}(Int) \rightarrow \mathcal{P}(Des)$
- Filter to select options
  $filter : \mathcal{P}(Bel) \times \mathcal{P}(Des) \times \mathcal{P}(Int) \rightarrow \mathcal{P}(Int)$
- Means-ends reasoning
  $plan : \mathcal{P}(Bel) \times \mathcal{P}(Int) \times \mathcal{P}(Ac) \rightarrow Plan$

Means-ends reasoning

What does the $plan$ function actually do?
$\Rightarrow$ *how* to **achieve goals** (ends) using available **means**

Classical **AI planning** uses the following representations as inputs:

- A **goal** (intention, task) to be achieved (or maintained)
- Current **state** of the environment (beliefs)
- **Actions** available to the agent

Output is a **plan**, i.e. a "recipe for action" to achieve a goal from current state.

STRIPS: classical planning system

STRIPS most famous classical planning system:

- State and goal are described as logical formulæ
- Action schemata describe preconditions & effects of actions

Most famous application scenario ⇒ Blocks world:

1. Given: A set of cube-shaped blocks sitting on a table
2. Robot arm can move around/stack blocks (one at a time)
3. Goal: configuration of stacks of blocks

Formalization in STRIPS:

- State description through set of literals, e.g.
  `{Clear(A), On(A, B), OnTable(B), OnTable(C), Clear(C)}`
- Same for goal description, e.g.
  `{OnTable(A), OnTable(B), OnTable(C)}`
- Action schemata: precondition/add/delete list notation

Blocks world example

Some action schemata examples:

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)}

PutDown(x) ???

(Linear) plan = sequence of action schema instances

Formal model of planning

Define a **descriptor** for an action $\alpha \in Ac$ as

$$\langle P_\alpha, D_\alpha, A_\alpha \rangle$$

$\Rightarrow$ sets of **first-order logic formulæ** of precondition, delete-, and add-list
(Although these may contain variables and logical connectives we ignore these for now and assume only ground atoms)

A **planning problem** $\langle \Delta, O, \gamma \rangle$ over Ac specifies:

- Δ as the (belief about) initial state (a list of atoms)
- a set of operator descriptors $O = \{\langle P_\alpha, D_\alpha, A_\alpha \rangle | \alpha \in Ac\}$
- an intention γ (set of literals) to be achieved

A **plan** is a sequence of actions $\pi = (\alpha_1, \ldots, \alpha_n)$ with $\alpha_i \in Ac$

Acceptable and correct

In a planning problem $\langle \Delta, O, \gamma \rangle$ a plan π determines a sequence of environment models $\Delta_0, \ldots, \Delta_n$.
For these we have:

- $\Delta_0 = \Delta$
- $\Delta_i = (\Delta_{i-1} \setminus D_{\alpha_i}) \cup A_{\alpha_i}$ for $1 \leq i \leq n$

Then:

- π is **acceptable** wrt $\langle \Delta, O, \gamma \rangle$ iff $\Delta_{i-1} \models P_{\alpha_i}$ for all $1 \leq i \leq n$
- π is **correct** wrt $\langle \Delta, O, \gamma \rangle$ iff π is acceptable and $\Delta_n \models \gamma$

The problem of AI planning:

Find a correct plan π for planning problem $\langle \Delta, O, \gamma \rangle$ if one exists, else announce that none exists.

Practical planning

Below, we will use:

- $head(\pi)$, $tail(\pi)$, $pre(\pi)$, $body(\pi)$ for parts of a plan
- $execute(\pi)$ to denote execution of a whole plan
- $sound(\pi, I, B)$ to denote that π is correct given intentions I and beliefs B

Note:

- Planning does note need to involve plan generation
- Plan libraries can be used (as in Jason)

$\Rightarrow$ Let's integrate **means-ends reasoning** into BDI implementation

BDI control loop (version 1)

Practical Reasoning Agent Control Loop v1:

$B \leftarrow B_0$; $I \leftarrow I_0$;
while true **do**
    $\rho \leftarrow \mathit{see}()$;
    $B \leftarrow \mathit{brf}(B, \rho)$; $D \leftarrow \mathit{options}(B, I)$; $I \leftarrow \mathit{filter}(B, D, I)$;
    $\pi \leftarrow \mathit{plan}(B, I, Ac)$;
    while $\neg(\mathit{empty}(\pi) \vee \mathit{succeeded}(I, B) \vee \mathit{impossible}(I, B))$ **do**
        $\alpha \leftarrow \mathit{head}(\pi)$; $\mathit{execute}(\alpha)$;
        $\pi \leftarrow \mathit{tail}(\pi)$;
    end
end

What could be the problem with this control loop?

Commitment

Are deliberation and planning sufficient to achieve desired behaviour? ⇒ Unfortunately not.

After **filter** function, agent makes a **commitment** to chosen option (this implies temporal persistence)
⇒ How long should an intention persist? (remember dung beetle?)

Three different commitment strategies:

- **Blind/fanatical** commitment: maintain intention until it has been achieved
- **Single-minded** commitment: maintain intention until achieved or impossible
- **Open-minded** commitment: maintain intention as long as it is believed possible

Important: agents commit themselves both to ends (intention) and means (plan)

Commitment to ends and means

With regard to **commitment to means**, the previous control loop implemented single-minded commitment (using predicates $succeeded(I, B)$ and $impossible(I, B)$).

Commitment to ends $\Rightarrow$ **intention reconsideration** (IR):

- When would we stop to think whether intentions are already fulfilled/impossible to achieve?
- Trade-off: intention reconsideration is costly but necessary $\Rightarrow$ meta-level control ($reconsider(I, B)$ predicate)
- IR strategy is optimal if it would have changed intentions had he deliberated again (assuming IR itself is cheap)

Rule of thumb: being "bold" is fine as long as world doesn't change at a high rate

Practical Reasoning Agent Control Loop v2:

```
B ← B₀; I ← I₀;
while true do
    ρ ← see();
    B ← brf(B, ρ); D ← options(B, I); I ← filter(B, D, I);
    π ← plan(B, I, Ac);
    while ¬(empty(π) ∨ succeeded(I, B) ∨ impossible(I, B)) do
        α ← head(π); execute(α);
        π ← tail(π);
        ρ ← see(); B ← brf(B, ρ);
        if reconsider(I, B) then
            D ← options(B, I); I ← filter(B, D, I);
        end
        if ¬(sound(π, I, B)) then
            π ← plan(B, I, Ac);
        end
    end
end
```


Background
BDI Architecture
Jason reasoning cycle
Perception
Planning
Action
Formal model of BDI
STRIPS
Blocks world
Formal model of Planning
General BDI control loop
Commitment
Summary

Summary

Summary

- Discussed practical reasoning systems
- Prevailing paradigm in deliberative agent design
- Deliberation defined as interaction between beliefs, desires, and intentions
- Jason reasoning cycle explained
- Means-ends reasoning and planning
- Commitment strategies and intention reconsideration

⇒ Next time: **Reactive and Hybrid Agent Architectures**

Acknowledgments

These lecture slides are based on the following resources:

- Dr. Michael Rovatsos, The University of Edinburgh `http://www.inf.ed.ac.uk/teaching/courses/abs/abs-timetable.html`
- Michael Wooldridge: **An Introduction to MultiAgent Systems**, John Wiley & Sons, 2nd edition 2009.
- Rafael H. Bordini, Jomi Fred Hübner, Michael Wooldridge: **Programming Multi-Agent Systems in AgentSpeak using Jason**, Wiley, 2007.