

Human-like artificial creatures

5. Belief-Desire-Intention

Cyril Brom

Faculty of Mathematics and Physics

Charles University in Prague

`brom@ksvi.mff.cuni.cz`

(c) 4/2006

Outline

1. Practical reasoning and Belief Desire Intention
2. Implementation
3. Jam, Jason, GOAL

Practical reasoning

- A model of decision making. Practical reasoning is a reasoning directed towards actions
 - what to do (**deliberative** reasoning)
 - how to do it (**means-ends** reasoning)
- Practical reasoning is not theoretical reasoning!
 - problem-solving vs. how to buy a ticket
- Limited computational resources
- The central concept of practical reasoning is a triad "belief – desire – intention"
 - the state of a BDI creature in any given moment is (Bel, Des, Int).
- Originally, Bratman offered a framework for assessment of an agent rationality
 - however, it is implementable
 - probably the first was the Procedural Reasoning System (Stanford)

Beliefs

- BDI architecture contains explicit representation of Beliefs, Desires, Intentions
- **Beliefs** represent information the agent has about its current environment ("environmental memory")
 - may be false

Intentions and desires

- Intentions present-directed (now) vs. future-directed vs. policy-based vs. ...
- **Intentions** are adopted / committed desires
 - **desires** are future agent's possibilities
 - intentions are states (of mind) that the agent has committed to trying to achieve
 - I've decided to drink a milk shake vs. I desire to drink a shake, but I'm fat.
- Intentions towards goals vs. towards means
- Intentions:
 - persist (but sometimes, intentions must be dropped)
 - drive means-ends reasoning
 - constrain future deliberation
 - influence beliefs upon which future practical reasoning is based
- The problem how often to reconsider intentions and eventually drop some is the problem of balancing between pro-active (goal-directed) and reactive (event driven) behaviour. Different types (static/dynamic) of environments require different types of reasoning

Abstract interpreter

- The state of a BDI agent in any given moment is (B, D, I)
 - current beliefs, desires, intentions

do

// generate new possibilities

options \leftarrow option-generator (events, B, D, I)

// select the best opportunities to perform

selected-options \leftarrow deliberate(options, B, D, I)

// adopt a selected opportunity as a subintention, or execute its actions

I \leftarrow I \cup selected-options[non-atomic]

execute(selected-options[atomic])

get-new-external-events()

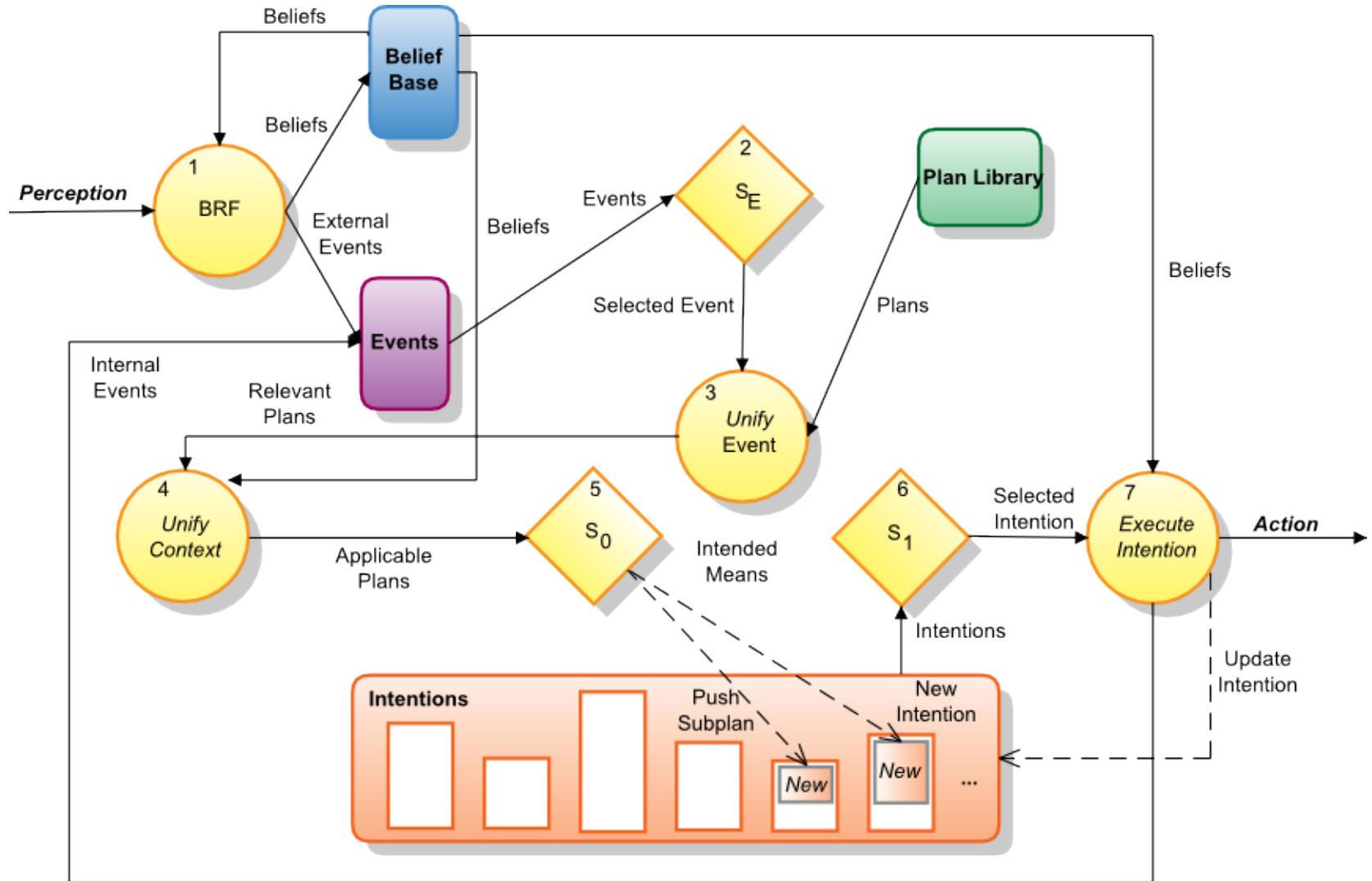
drop-successful-goals(B, D, I)

drop-impossible-goals(B, D, I)

until quit

[Sing et al., 1999]

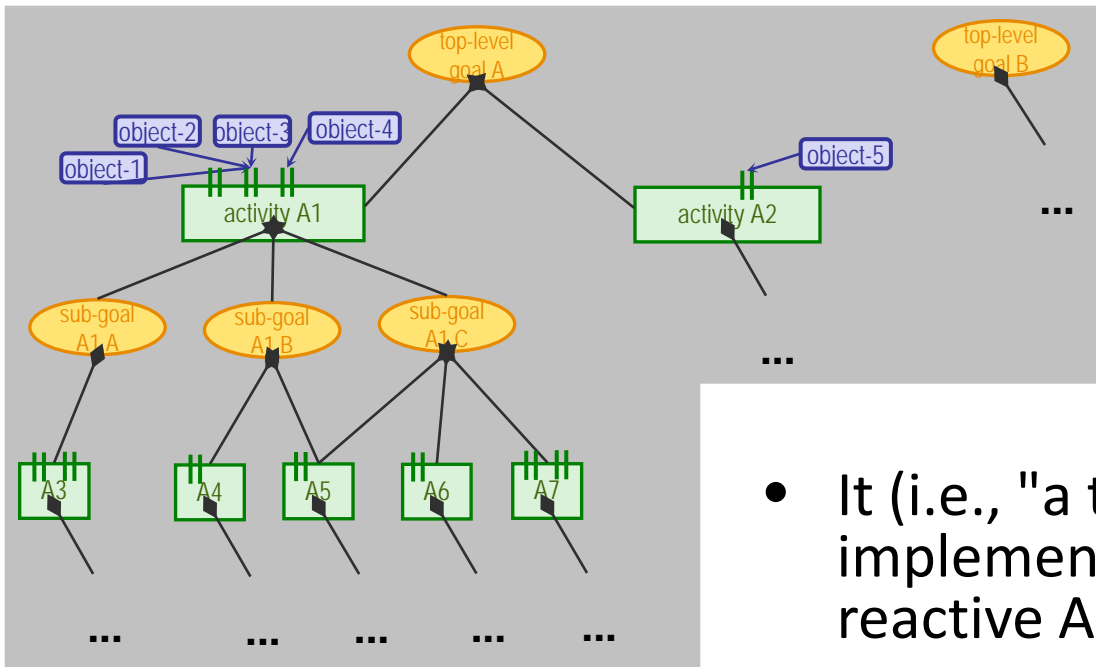
Abstract interpreter



BDI Interpreter – notes

- Typically operates with prescribed plans and an intentional stack
- Plans are stored structures that determines how to achieve an intention
 - preconditions: a body of the plan is believed to be an option whenever its invocation condition / precondition are satisfied
 - atomic actions
 - generation of a new goal that can be adopted as a subintention
 - means-ends are not performed typically
- Intentional stack holds all adopted intentions / subintentions
- Deliberation
 - with respect to the time-constrains
 - random, priorities or meta-level reasoning

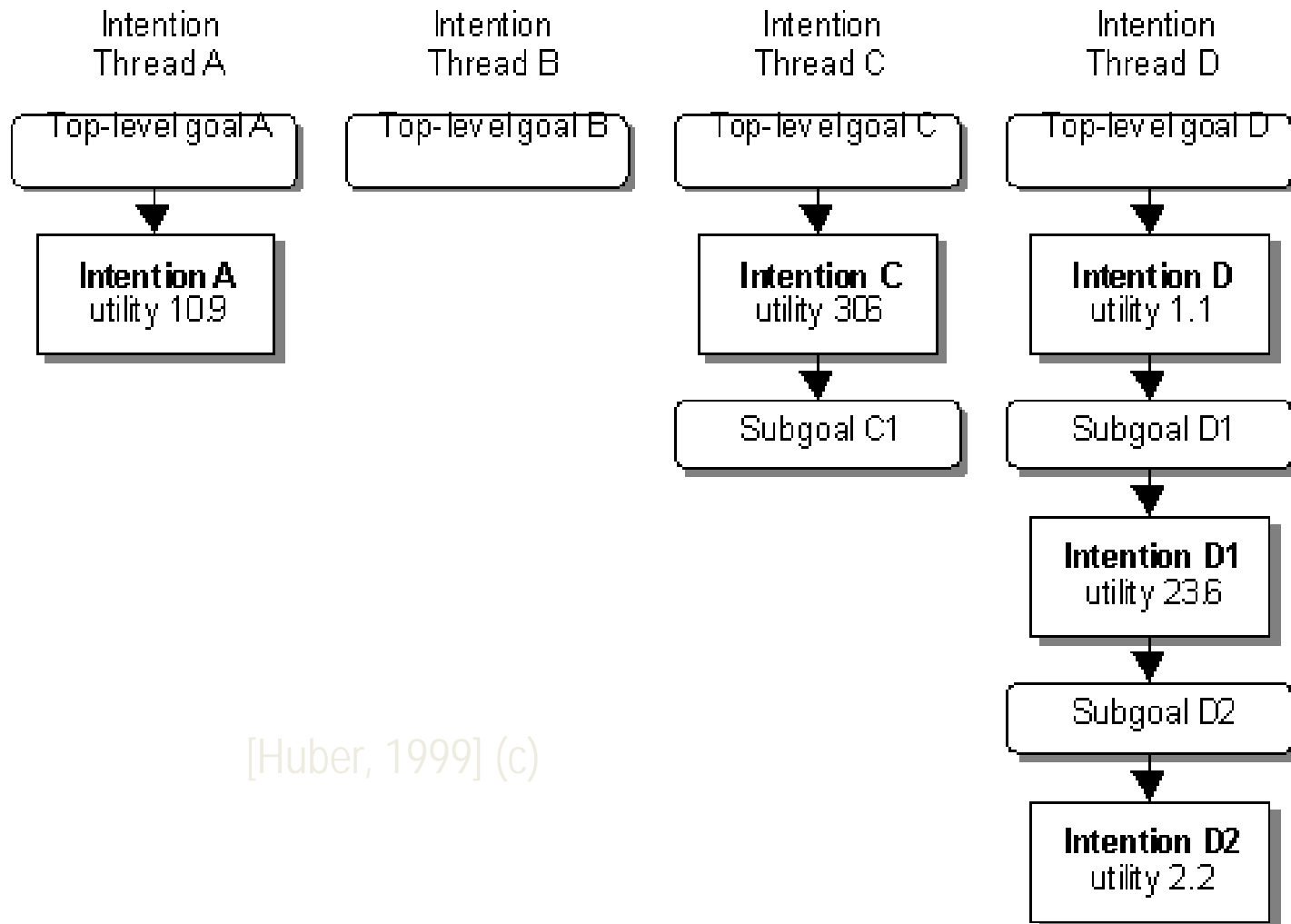
BDI Interpreter - notes



Is the creature able to answer the question what it is going to do this afternoon?

- It (i.e., "a typical BDI implementation") resembles reactive AND-OR trees
- It is actually a robust reactive architecture
 - except of deliberation
- It operates only with present-directed intentions
- Jack, JAM, Jason

Jason / JAM intentional stack



[Huber, 1999] (c)

JAM memory example

FACTS:

```
FACT robot_status "Ok";
FACT partner_status "Ok";
FACT robot_initialized "False";
FACT robot_localized "False";
FACT robot_registered "False";
FACT robot_position 10000 10000 0;
FACT robot_location "Unknown";
FACT self "CARMEL";
FACT partner "BORIS";
FACT object_found "False";
FACT object_delivered "False";
FACT communication_status "Ok";
FACT plan_empty "False";
FACT destination "Room4";
FACT next_room "Room3";
FACT next_node "Node12";
```

[Huber, 1999]

```

Plan: {
NAME:
    "Example plan"
DOCUMENTATION:
    "This is a nonsensical plan that shows all of the possible actions"
GOAL:
    ACHIEVE plan_example $distance;
PRECONDITION: (< $distance 50);
CONTEXT:
    RETRIEVE task_complete $STATUS;
    (== $STATUS "False");
BODY:
    QUERY determine_task $task;
    FACT problem_solved $task $solved;
    OR
    {
        TEST (== $solved "YES");
        WAIT user_notified;
        RETRACT working_on_problem "True";
    }
    {
        TEST (== $solved "NO");
        ACHIEVE problem_decomposed;
    }
}
    ASSIGN $result (* 3 5);
    };
    UPDATE (task_complete) (task_complete "True");
FAILURE:
    UPDATE (plan_example_failed) (plan_example_failed "True");
    EXECUTE print "Example failed.  Bailing out"
ATTRIBUTES: "test 1 cpu-use 3.0";
EFFECTS:
    UPDATE (task_complete) (task_complete "True");
• }

```

JAM plan example

[Huber, 1999]

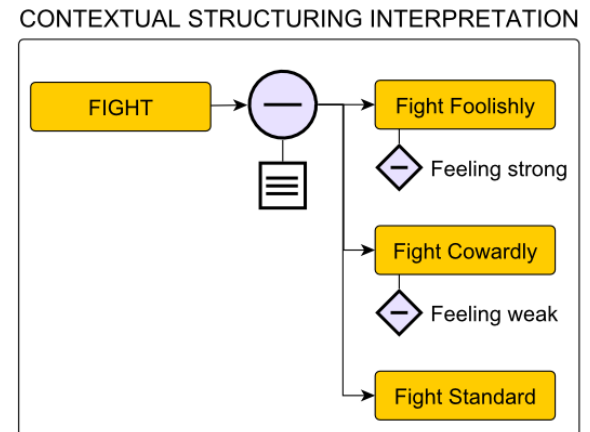
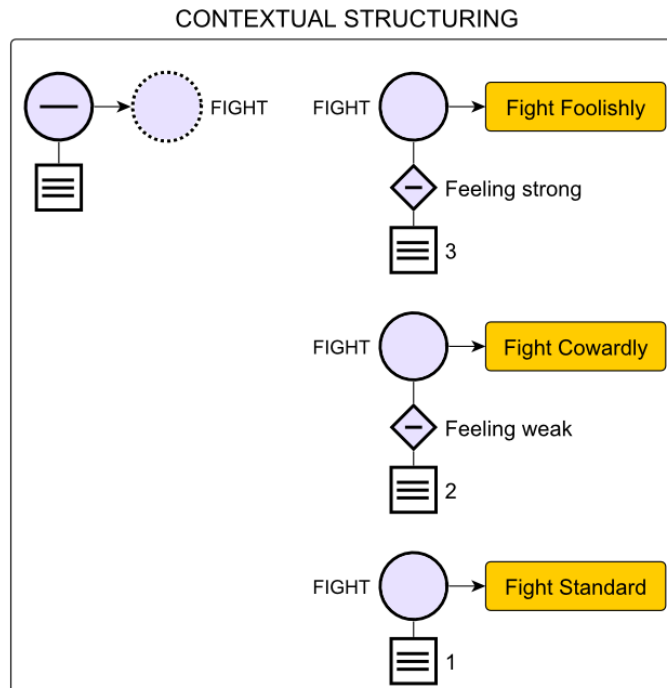
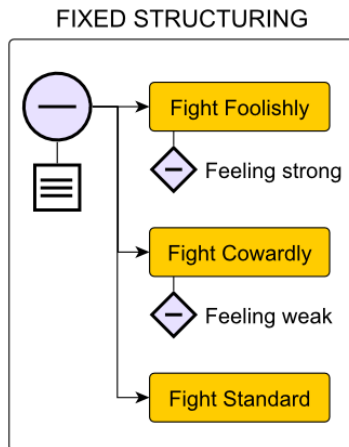
```
// in case I am in a dirty location  
+dirty: true <- suck.
```

```
// in case I am in a clean location  
+clean: pos(l) <- right.  
+clean: pos(r) <- left.
```

Jason plan example

What is actually the most interesting is "contextual plan invocation".

=> Interesting “node type”



GOAL – Blocks world

The screenshot shows the Eclipse IDE interface with the following components:

- Script Explorer:** Shows the project structure for 'BlocksWorld', including 'blocksworld', 'blocksworld.mas2g', 'bw2agents.mas2g', 'stackBuilder.goal', 'tableAgent.goal', and 'BW4T2'.
- Main Editor:** Displays the content of 'stackBuilder.goal'. The code is as follows:

```
% This agent moves blocks in order to achieve a target configuration of blocks (the goal in its goal base).

% Initialize the agent and it's mental state.
init module {
  knowledge {
    % only blocks can be on top of another object.
    block(X) :- on(X, _).
    % a block is clear if nothing is on top of it.
    clear(X) :- block(X), not( on(_, X) ).
    % the table is always clear.
    clear(table).
    % a tower is any non-empty stack of blocks that sits on the table.
    tower([X]) :- on(X, table).
    tower([X, Y | T]) :- on(X, Y), tower([Y | T]).
  }
  goals {
    % a single goal to achieve a particular configuration of blocks.
    % assumes that these blocks are available in the Blocks World.
    on(b1,b5), on(b2,table), on(b3,table), on(b4,b3), on(b5,b2), on(b6,b4), on(b7,table).
  }
  actionspec {
    % moves a block on top of another block or to the table.
    move(X, Y) {
      % a block can only be moved elsewhere if that block and the place to move it to are both clear.
      % on(X, Z) retrieves the current (and soon to be old) position where block X sits on.
      % not( on(X,Y) ) prevents an agent from moving a block on the table to another place on the table.
      % because a block cannot be put on top of itself not( X=Y ) is included in the precondition.
      pre { clear(X), clear(Y), on(X, Z), not( on(X, Y) ), not( X=Y ) }
      % effect of moving block X on top of Y is that we have on(X, Y).
      % after moving, block X is no longer on top of Z and we remove on(X, Z).
      post { not( on(X, Z) ), on(X, Y) }
    }
  }
}
```
- Outline:** Shows the structure of the 'initmodule()' module, with 'knowledge()' expanded to show 'block', 'X', 'on', and 'X'.
- Problems/Console:** Displays the message 'No consoles to display at this time.'

GOAL – Blocks world

```
event module{
  program{
    forall bel(percept(on(X,Y)), on(X,Z), not(Y=Z)) do insert(on(X,Y), not(on(X,Z))).
  }
}
```

```
main module{
  program{
    if a-goal(tower([X,Y|T]), bel(tower([Y|T])) then move(X,Y).
    if a-goal(tower([X|T])) then move(X,table).
  }
}
```


End.

References

1. M. Wooldridge: *An Introduction to MultiAgent Systems*. John Wiley & Sons (1995)
2. Gerhard Weiss (ed.): *Multiagent Systems*. Ch. 8 (by Singh et al.), MIT Press (1999)
3. Huber, M. J.: JAM: A BDI-theoretic mobile agent architecture. In: *Proceedings of the 3rd International Conference on Autonomous Agents (Agents'99)*. Seattle (1999) 236-243
4. Bratman, M. E.: *Intention, Plans and Practical Reason*. Harvard University Press, Cambridge, Ma (1987)
5. Jack homepage: <http://www.agent-software.com/shared/home/index.html>
6. IVE homepage: <http://urtax.ms.mff.cuni.cz/ive/public/about.php>
7. Sing M. P., Rao A. S., Georgeff M. P: BDI Implementations, chapter 8.4. In: *Multiagent systems* (Wies, G. – eds.) 1999