What is a LB agent?

A logic-based agent is a software agent defined by:

- a language $L$
- a logic theory $T$ over the language $L$
- a declarative semantics $DS(T)$ of $T$
  
  meaning of $T$: set of logical consequences of $T$
- a procedural semantics $PS(T)$
  how to compute the logical consequences of $T$
- provide properties of $PS(T)$
  ideally, $PS(T)=DS(T)$
Capabilities of our agents

We propose an approach to agents that can:

- **reason** and **react** to the environment (including other agents)
- **update** their own knowledge, reactions and goals
- **interact** by updating the theory of another agent
- **decide** whether to accept an update depending on the requesting agent
- **prefer** among possible choices
- **abduce** hypotheses to explain observations

Updating agents

**Updating agent**: a rational, reactive agent that can dynamically change its own knowledge and goals

- makes observations
- updates its knowledge, reactions and goals
- thinks a bit (rational)
- selects and executes an action (reactive)
### Example 1

| S₁: | sleep ← not watch_tv |
|     | watch_tv ← tv_on |
|     | tv_on |
|     | sleep ⇒ set (alarm_clock) |
|     | ?- sleep |
|     | no |

| S₂: | not tv_on ← power_failure |
|     | power_failure |
|     | ?- sleep |
|     | yes |
|     | action: set (alarm_clock) |

| S₃: | not power_failure |
|     | ?- sleep |
|     | no |

### Preferring agents

**Preferring agent:** an agent that is able to prefer knowledge and reactions when several alternatives are possible

- agents can express preferences about their own rules and hypotheses
- preferences can be updated, possibly on advice from others
Example 2

Let the theory of an agent John contain:

\[ S_1: \]
- \((r_1)\) work \leftrightarrow \text{not money}
- \((r_2)\) beach \leftrightarrow \text{money, not mountain}
- \((r_3)\) mountain \leftrightarrow \text{money, not beach}
- \((r_4)\) summer
- \((r_5)\) not money

\( r_2 < r_3 \leftrightarrow \text{summer} \)

\[ S_2: \]
- money

\[ ?- \text{work} \]
- yes

\[ ?- \text{work} \]
- no

\[ ?- \text{mountain} \]
- no

\[ ?- \text{beach} \]
- yes

Example 3

Let \( A = \{\text{fire, smoking(john)}\} \) be the hypotheses (abducibles) of an agent Mary

\[ S_1: \]
- smoke \( \leftarrow \) fire
- smoke \( \leftarrow \) smoking(john)
- fire \( \Rightarrow \) give_alarm
- smoking(john) \( \Rightarrow \) scream

\[ S_2: \]
- fire \( < \) smoking(john) \( \leftarrow \) at(john, pub)
- at(john, pub)

\[ ?- \text{smoke} \]
- \{fire\}
- \{smoking(john)\}

\[ ?- \text{smoke} \]
- \{fire\}

action: give_alarm
Agent architecture

Java

Control Cycle

InterProlog

Rational P

can abduce

XSB Prolog

Reactive P+R

cannot abduce

XSB Prolog

Agent architecture’s implementation

by Mattias Engberg

Action Handler

External Interface

UpdateH

Projects

int.project

ext.project

Rational P

Reactive P+R
Fw: logic-based controllers

- Use of agents as logic-based controllers
  (joint work with A. Lombardi)
  - simple artificial world: balloon environment

Fw: agent organizational structures

- Formalize organizational structures for epistemic multi-agent systems (eMAS)
  - groups, institutions, societies, etc.
  - norms, regulations, etc.

- Ongoing implementation of a platform supporting the interactions of our logic-based agents as well as some forms of agent structures (by Mattias Blixt)
Use of preference reasoning at query time to facilitate the retrieval of information wrt. users’ interests  (joint work with A. Vitória)

- how to incorporate abduction:
  abductive preferences leading to conditional answers depending on accepting a preference

- group preference:
  how to tackle the problem arising when we have several users query the system together

Applications

Applications in which our agent technology can have a significant potential to contribute are web applications, e.g.

- information integration
  how to integrate data from multiple heterogeneous sources and to provide a uniform interface, e.g. to provide info about movies

- web-site management
  self-reconfigurable and adaptive web sites
  declarative representation of web sites allows:
  - to automatically reconstruct them, e.g. on usage patterns and can adapt themselves wrt. user profiles
  - to enforce integrity constraints on web sites, e.g. no dangling pointers