Multi-agent systems

Andrej Lúčny

KAI FMFI UK

lucny@fmph.uniba.sk

http://www.agentspace.org/mas

We are going to deal with creatures which have language communication (Gregorian kind of mind)

How work with a language structures in a reasonably way?

Though natural language seems to be not treatable for computer, we still can find a reasonable way how to approach the problem

... and that is use of a rigid language of mathematical logic

That language differs from natural one a lot, but still it can be concerned as its improvement

Minsky about Logic

- Human mind is not based on logic
- Human mind is just chaining information
- Logic is such a chaining that only information as mutually relevant as possible are chained
- Laughing is state od mind the information chaining encounter troubles with inconsistence

First-order logic

Language:

$$\forall x, y, z.(P(x, y) \land P(y, z)) \rightarrow P(x, z)$$

- variables
- functions (functions with zero arity are constans)
- predicates
- conjunctions
- parenthesis
- quantifies
- option: equation

- terms `car(mercedes,silver)`
- literals

`¬Cheap(car(mercedes,silver))`

Proof

- Language of Logic provide us syntactical operations (substitution, modus ponens) which chaining corresponds to a proof of a theorem (finding a truth)
- Assumptions of the proof are the fundamental axioms and axioms of a theory

First order logic is a very strong theoretical concept …

- Correctness:
  Whatever we have proved is true in any model of theory
- Godel's completeness theorem:
  Whatever is true in any model of theory can be proved from axioms of the theory (and fundamental axioms)

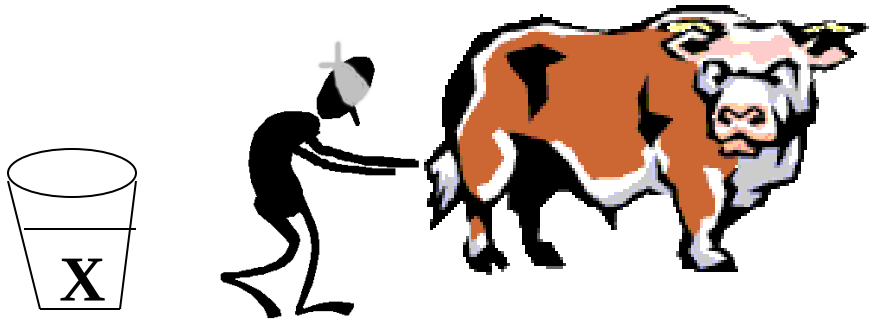
… but has limits

- The first Godel's incompleteness theorem:
  In any consistent theory which contains arithmetics and which axioms can be generated aby a computer program, there are contentions (Godel's formulas), which cannot be proved or refuted
- (We know that such contention are true in the standard model of natural numbers. But there are non-standard models in which they are false and the proof mechanism is not working for them)
- The second Godel's incompleteness theorem:
  consistency of such theorems can be formulated but cannot be proved.

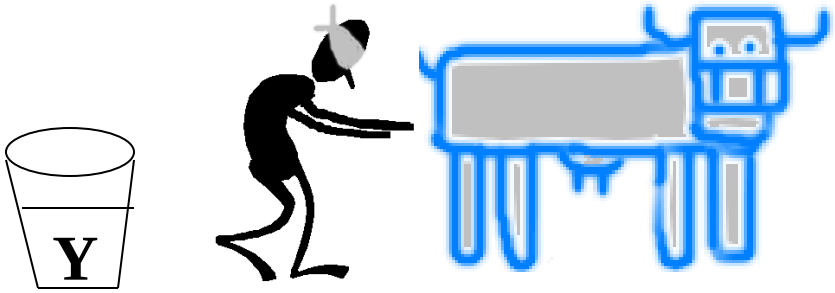
Can machine think?

- Alan Turing has asked the question in 1950 (as a result he proposed so called Turing's test). He imagine a machine which consistently responds questions: yes or no.
- If one put a Godel's formula to the machine, it cannot provide a correct answer, it must stay in endless loop.
- Turing allowed inconsistency, e.g. machine has a timeout and it expires the machine answers randomly or "I have no idea".

AI aims to create artificial systems which corresponds to living intelligent systems, namely to human



X = Y ?



We have no idea about their nature, but we can test them by **Turing's test**

Procedural programming languagues

- Derivation from logical formulas can be turned to programming language
- For instance Horn clauses corresponds to language PROLOG. (Each formula can be expressed in from of Horn clauses by skolemization)
- Silimar language is OWL (for semantic web)

PROLOG

```
father(janko,jozko).
father(jozko,ferko).
grandfather(X,Y) :- father(X,Z), father(Z,Y).
?- grandfather(janko,ferko).
#TRUE
?- grandfather(ferko,janko).
#FALSE
```

PROLOG

If A implies B, i.e.

B ← A = B v ¬A

B is unsatisfiable just when A is unsatisfiable

father(janko,jozko) ← F farther(janko,jozko)

father(jozko,ferko) ← F farther(jozko, ferko)

grandfather(X,Y) ← father(X,Z) & father(Z,Y)

F ← grandfather(janko,ferko) ¬grandfather(janko, ferko)

is unsatisfiable, so that grandfather(janko,ferko) is true

F ← grandfather(ferko,janko) ¬grandfather(ferko, janko)

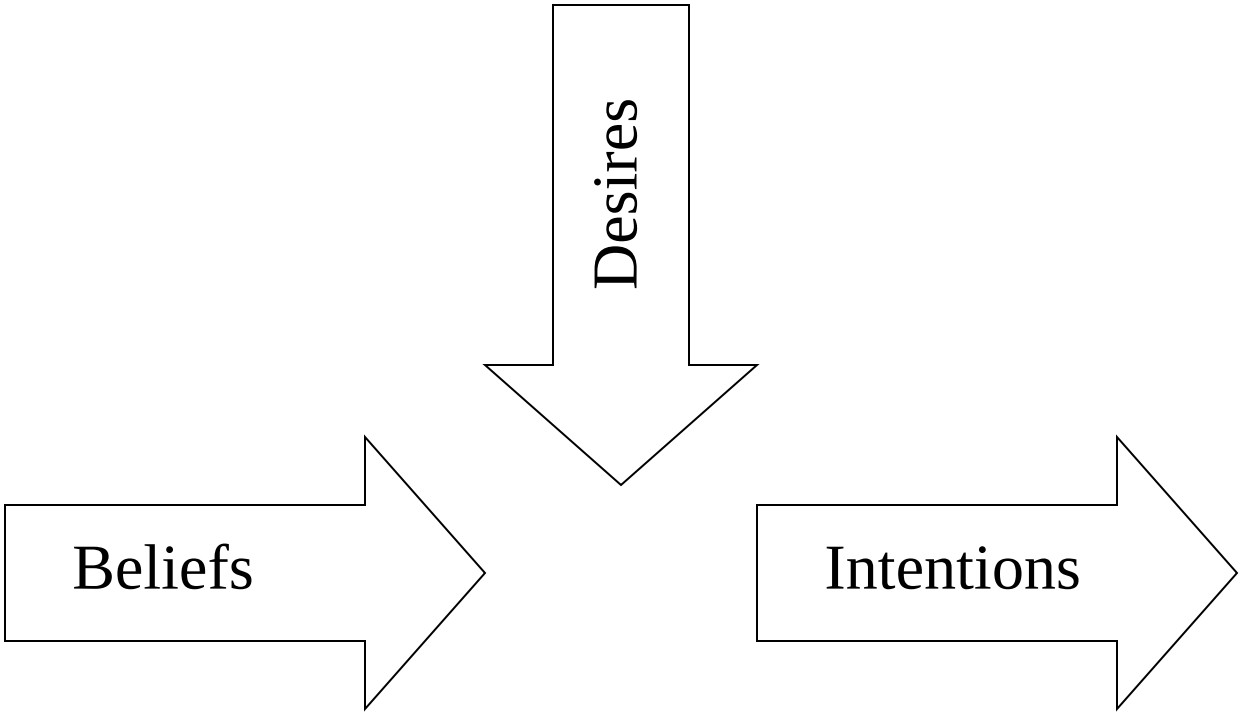
is not unsatisfiable so that grandfather(ferko,janko) can be true (closed world assumption)

To confirm thruth of a contention we try to prove unsatisfiability of its negation by the back tracking algorithm (backward chaining)

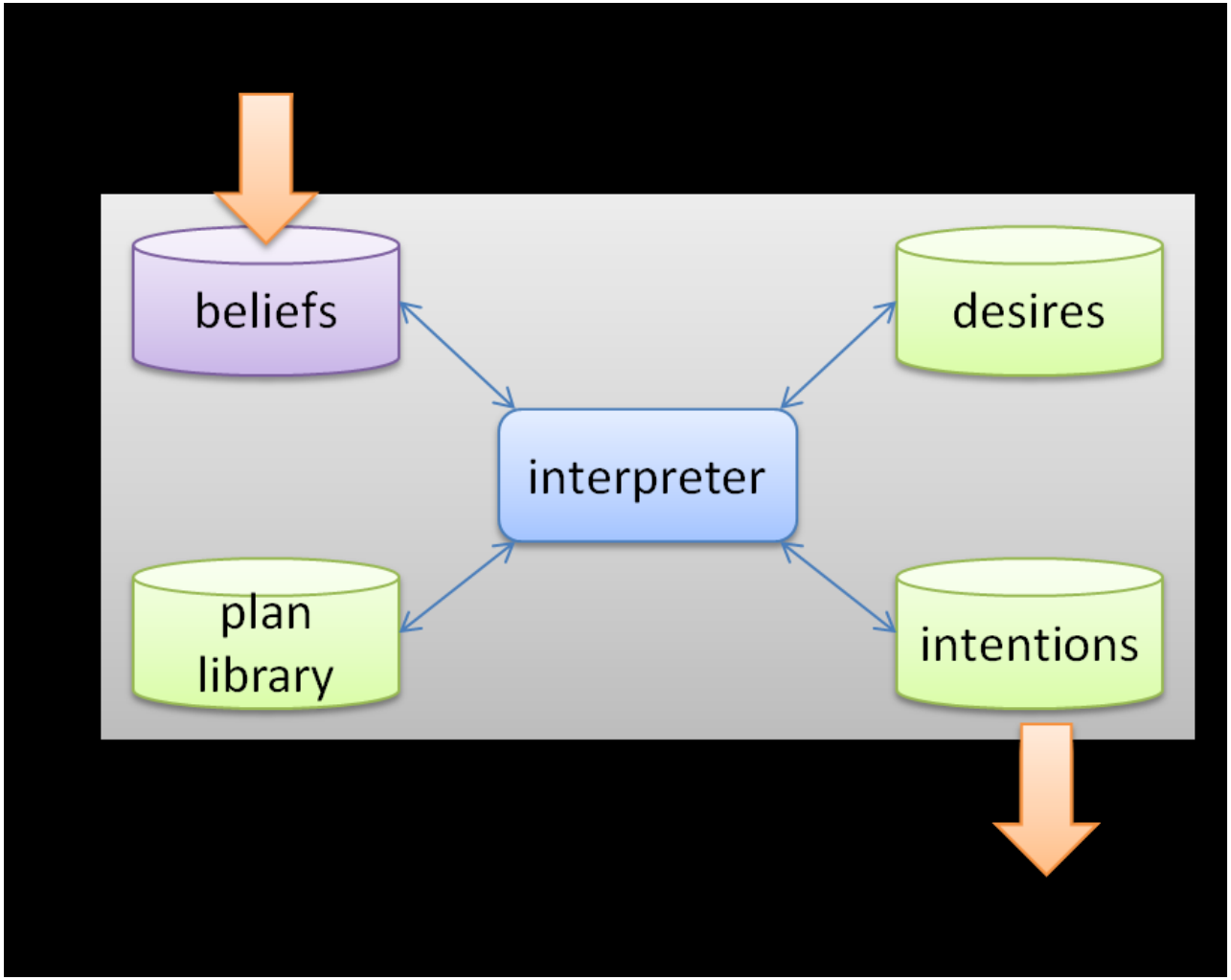
Non-monotonic logic

- First-order logic is not suitable for expression of changes, so it is not best for description what is happening in mind of sensing, acting and communicating agents
- However it is possible to extend it by non-monotonic mechanisms which are adding and deleting formulas to keep their mutual consistency (or alternatively formulas can be inconsistent but their model is changing and is always consistent - Kripke)

BDI Agents



BDI Agents

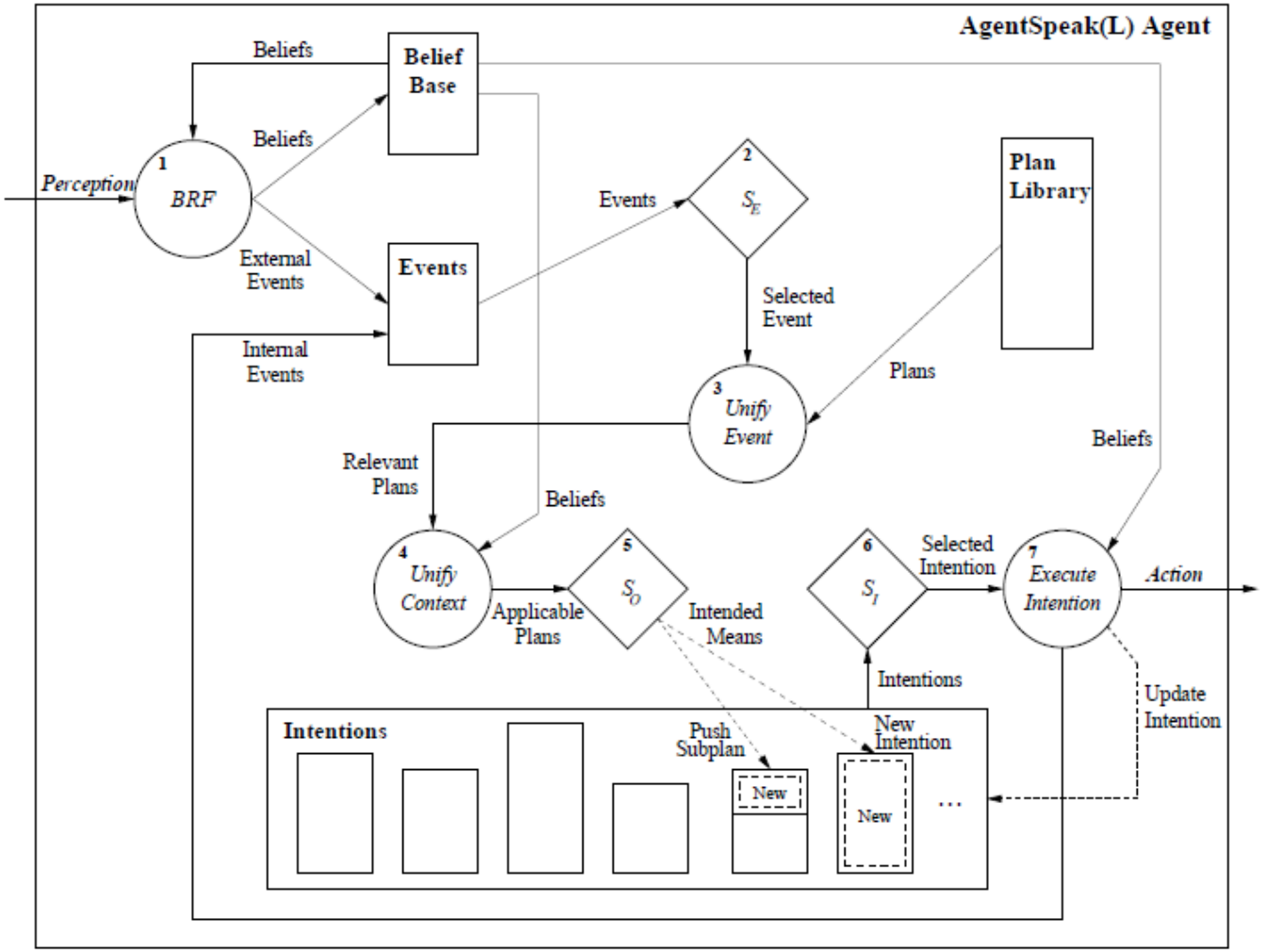


BDI - agents

```
Initialize state();
do
    options := option-generator(event-queue, B,G,I);
    selected-option := deliberate(options,B,G,I);
    I :=  update-intentions(selected-option,I);
    execute(I);
    event-queue := event-queue + get-new-external-events();
    drop-successful-attitudes(B,G,I);
    drop-impossible-attitudes(B,G,I);
until quit.
```

AgentSpeak

- Implementation of BDI Agents



AgentSpeak

Language:

- variables X
- functions girl(X), ... (constants 1, 2, ...)
- predicates observed(t)
- conjuctions & | not <–
- no parenthesis, just ; and .
- quantifier for () { … }
- equation =

- Literal (grounded)

observed(girl(anicka))

- Term

Extensions

- numbers and arithmetic operators A+1=B
- inequations >= <=
- embeded predicates: .atom .literal .string
- desires, goals: !find(anicka) ?beauty(anicka)
- plans + –
- counts #
- constraints : height(X) & X >= 150
- nameless variable _ e.g. height(_)

Non-monotonic mechanisms

- Embeded procedures manipulating formulas:

.add_plan	add to plans
.abolish	remove from beliefs
.asserta	add to beliefs
.broadcast	send a message to all agents
.create_agent	launch a new agent
.date	get current date
.time	get current time
.desire	add to desires
.kill_agent	kill a running agent
.send	send a message to one agent
.wait	wait for a given period

Communication among agents

.send(receiver,tell,"Hallo",noid).

.broadcast(tell, "Hallo").

```
+!kqml_received(Sender, tell, Content, MsgId) <-
.print("received ",Content," from ",Sender," id: ",MsgId).
```

AgentSpeak employs KQML

KQML - ACL

AgentSpeak is able to translate ACL to KQML and vice versa

For example `inform` in ACL is `tell` in KQML

Names of performative are important since AgentSpeak has inbuilt reactions to various perfromatives. E.g. when `tell` is received, the message content is inserted into Beliefs

```
/* Initial beliefs and rules */
b(20).
b(21).
b(22).

/* Initial goals */
!tick.

/* Plans */

+!tick <-
   for (b(X)) { .print(X); }
   .abolish(b(_));
   .sense(b);
   for (b(Y)) { .act(c(Y,4)); }
   .wait(1000);
   !tick.
```

Program in AgentSpeak

- We program mainly by specification of plans

Jason

interpreter of AgentSpeak written in Java

It can cooperate with JADE

It support more infrastructures (from simulation of homogeneous agent in grid to JADE agents in netowork)

Jomi F. Hübner and Rafael H. Bordini, 2004

What is it good for?

We can express logic of agent behavior and it communication in AgentSpeak what can be more readable as in Java.

We can employ an existing scenario in AgentSpeak for further extensions. (Incremental development can be easier)

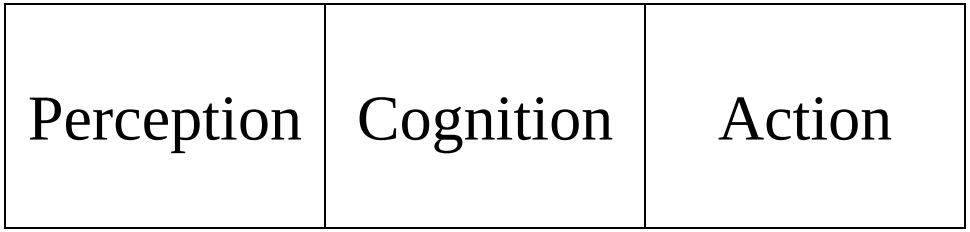
How we can interconnect the AgentSpeak agent with a world?

It is not possible in the language.

Therefore we have more choices how to code the interconnection in Java. Namely we can code custom embedded procedures which :

- turn agent perception into logic formulas
- interpret logic formulas as orders for actions

Thus AgentSpeak is Cognitivism



- Perception and action are separated by a cognitive module which interprets AgentSpeak language

Troubles

- We already know that cognitivism causes some troubles

Frame problem

- Lack of facts about the world got by the modelling process

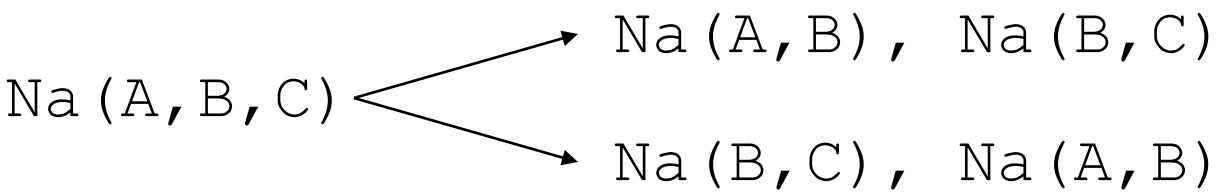
vs.

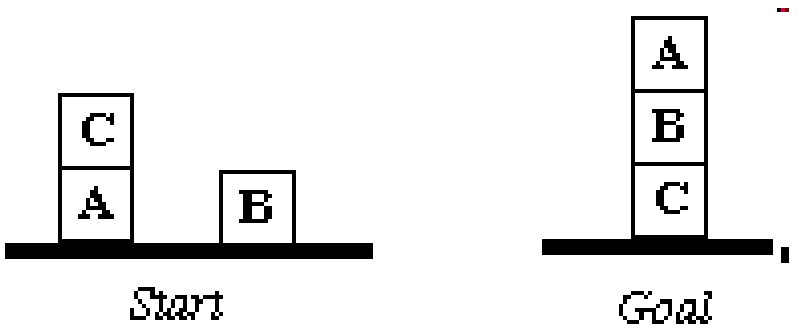
- Lack of computational power for solver to derive plan

- Caused by: modelling is loosing semantics and solver is spending a lot of time by syntactical operations over model parts which have no semantic relation

Sussman anomaly

Solving problems by decomposition to subgoals does not generate optimal solutions.





Solution: hybrid system

- One so called deliberative agent (cognitivism)
  +
  more so called reactive agents (post-cognitivism)
  =
  we divide job, what is easy for deliberative agent and what is easy for reactive agents

But: communication is fine

- No serialization/deserialization of data to message and back if we communicate formulas
- Agents in AgentSpeak communicate and think in the same language
- We can communicate:
  .send(receiver,tell,is(anicka,pretty),noid).
- And then recipient believes:

  is(anicka,pretty)

Communication acts

- Here we see why we need more communication acts:
- content of tell is inserted int Beliefs.
- while content of ask-one is matched to Beliefs and result is answered to the sender.