Multiagentové systémy

Andrej Lúčny

KAI FMFI UK

lucny@fmph.uniba.sk

http://www.agentspace.org/mas

Budeme sa zaoberať tvormi, ktoré dokážu myslieť pomocou jazyka (gregoryovský typ myslenia)

Ako rozumne manipulovať s konštruktami jazyka?

Ak aj nevieme s jazykom manipulovať ako človek, stále ešte môžeme nájsť nejaký rozumný spôsob

... A to je použitie matematickej logiky

Jazyk matematickej logiky sa významne líši od ľudského, ale bol vytvorený sťaby jeho zdokonalenie.

Minsky o logike

- Ľudská myseľ nefunguje na báze logiky
- Ľudská myseľ reťazí informácie
- Logika je také reťazenie, že zreťazí len informácie relevantné k sebe najviac ako je len možné
- Smiech je stav mysle, kedy sa pri reťazení informácii zdetekuje spor

Predikátová logika

Jazyk logiky tvoria

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

- Premenné
- Funktory (z toho nulárne sú konštanty)
- Predikátové symboly
- Logické spojky
- Zátvorky
- Kvatifikátory
- prípadne aj znamienko rovnosti

- termy `car(mercedes,silver)`
- literály

`¬Cheap(car(mercedes,silver))`

Dôkaz

- S jazykom logiky sa dajú robiť syntaktické operácie (substitúcia a modus ponens), ktoré vhodne zreťazené, zodpovedajú dôkazu teorémy (nájdeniu pravdy)
- Predpokladmi dôkazu sú základné axiómy a axiómy teórie

Predikátová logika má úžasnú teoretickú silu

- Veta o korektnosti:
  čokoľvek takto dokážeme je pravdivé v každom modeli teórie
- Godelova veta o úplnosti:
  čokoľvek je splnené v každom modeli teórie, dá sa aj z axiómov tejto teórie aj dokázať

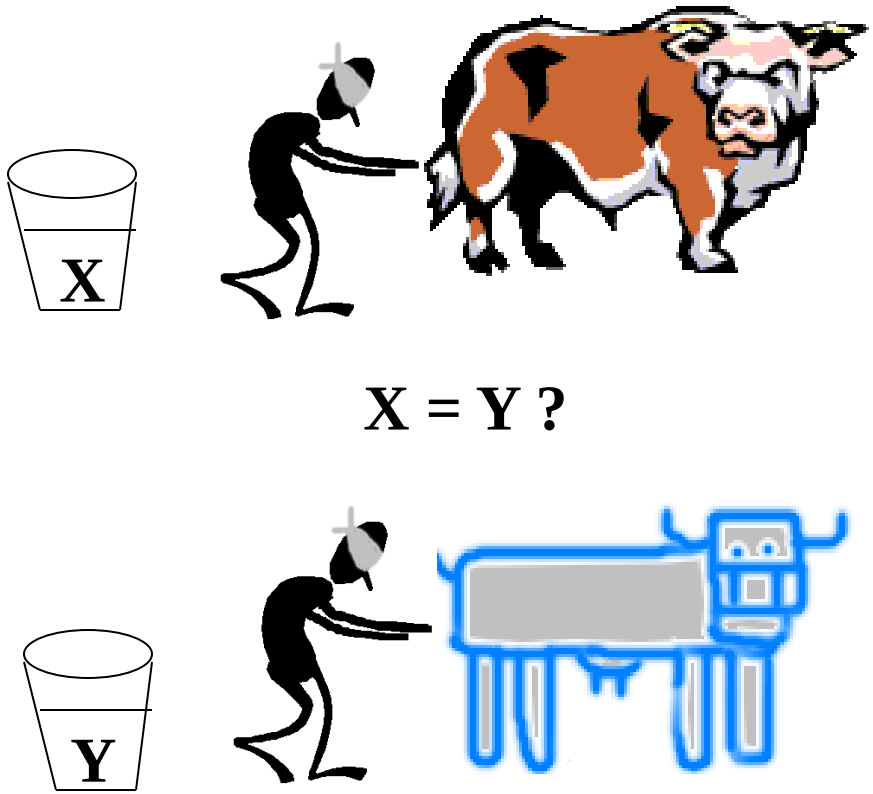
Ale nie je všemocná

- Prvá Godelova veta o neúplnosti:
  pre bezospornú teóriu obsahujúcu aspoň aritmetiku a ktorej axiómy sa dajú vygenerovať počítačom, existujú tvrdenia (Godelove formuly), ktoré sa nedajú dokázať ani vyvrátiť
  (V štandardnom modeli prirodzených čísel sú inak splnené, čoho sa my vieme dobrať. Existujú však neštandardné modely, v ktorých splnené nie sú)
- Druhá Godelova veta o neúplnosti:
  bezospornosť týchto teórii možno v rámci nich samotných sformulovať, ale nemožno ju dokázať.

Môžu stroje myslieť?

- Alan Turing riešil v 1950 otázky či stroje môžu myslieť (výsledkom bol Turingov test)
- Čo urobí stroj, ktorému podhodia na vstup rozhodnúť (z neho odvodenú) Godelovu formulu? Pokiaľ pracuje bezosporne, tak sa „zacyklí“.
- Turing to rieši tým, že spornosť dovolí, napr. si stroj odmeria čas a potom hodí mincou, čo je pravda, alebo povie, že nevie, ...

Cieľom UI je vytvárať umelé systémy, ktoré zodpovedajú dostupným živým, resp. človeku



Aké to sú, to nechávame otvorené, ale keď nejaký dostaneme, vieme overiť či je taký - **Turingov test**

Procedurálne programovacie jazyky

- Odvodzovanie z logických formúl sa dá premeniť na programovací jazyk
- Napríklad Hornove klauzuly, na ktoré sa dá skolemizáciou premeniť každá logická formula, zodpovedajú programovaciemu jazyku PROLOG
- Podobne OWL (sémantický web)
- vždy to má určité problémy, napr. klasický PROLOG nemá negáciu

PROLOG

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

PROLOG

Ak A implikuje B, tj.

B ← A = B v ¬A

B je nesplniteľné práve vtedy, keď A je nesplniteľné

otec(janko,jozko) ← F otec(janko,jozko)

otec(jozko,ferko) ← F otec(jozko, ferko)

dedko(X,Y) ← otec(X,Z) & otec(Z,Y)

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

je nesplniteľné, a teda dedko(janko,ferko) je pravda

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

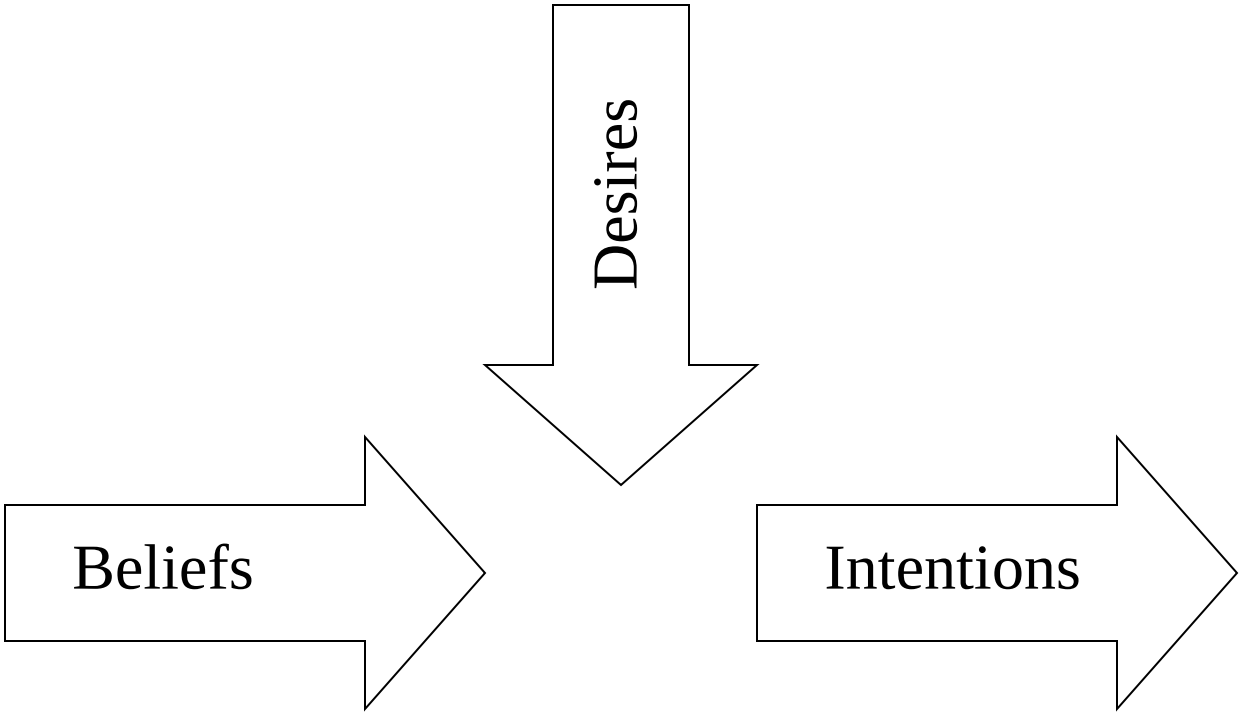
nie je nesplniteľné a teda dedko(ferko,janko) je prípadne nepravda (predpoklad uzavretého sveta)

Výpočet pravdivosti tvrdenia = snažíme sa dokázať nesplniteľnosť jeho negácie prehľadávaním so spätným návratom (spätné reťazenie)

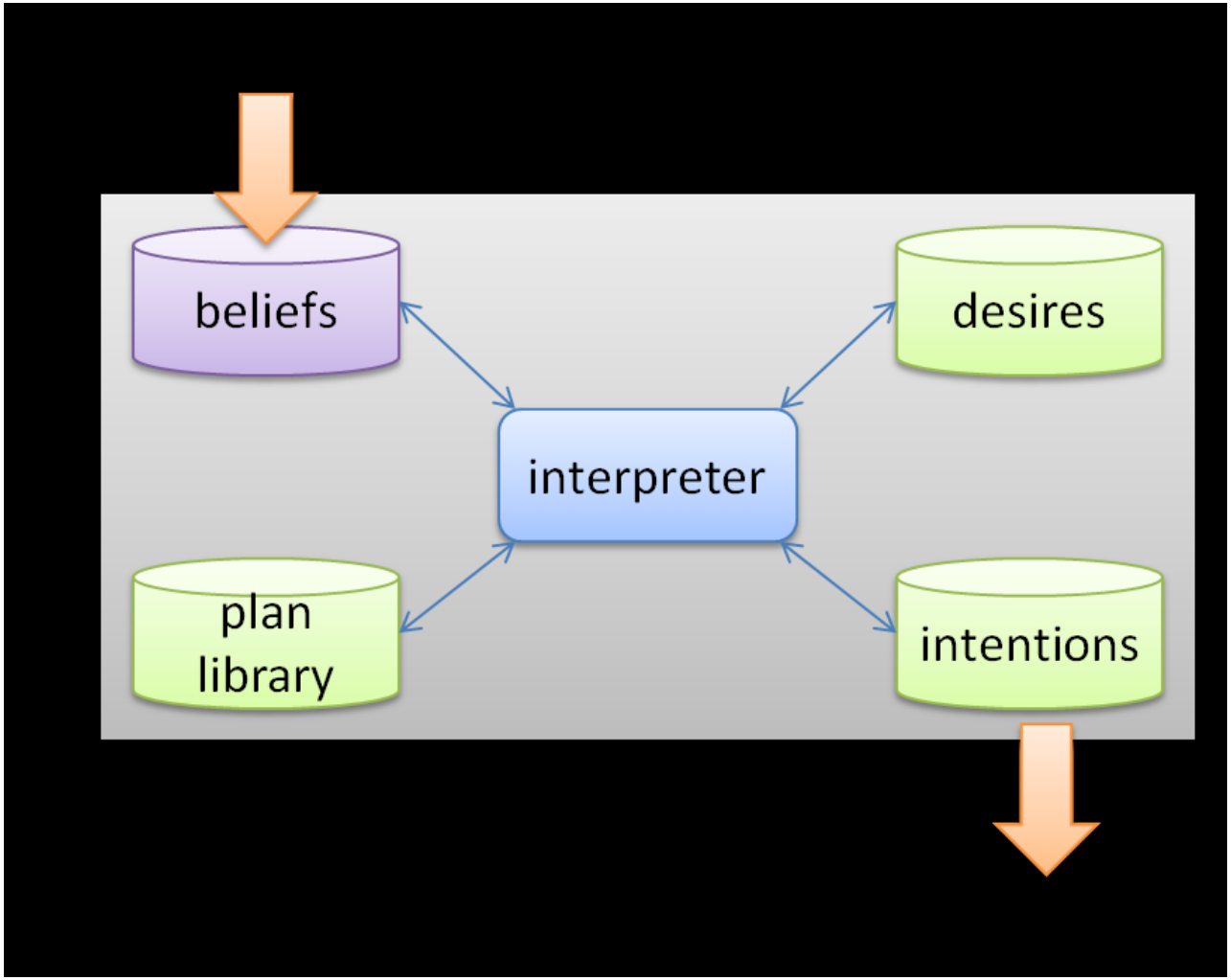
Nemonotónne logiky

- Štandardná predikátová logika sa zaoberá nemennými vecami a nie je vhodná na zachytenie toho, čo sa deje v hlave konajúcich a komunikujúcich agentov
- Dá sa ale rozšíriť o prostriedky nemonotónnosti a to buď z hľadiska mechanizmu, ktorý ruší a pridáva spracúvané formuly (a drží ich v bezospornom stave), alebo z hľadiska mechanizmu na budovanie modelu zo vzájomne sporných formúl (Kripkeho model)

BDI Agenty



BDI Agenty

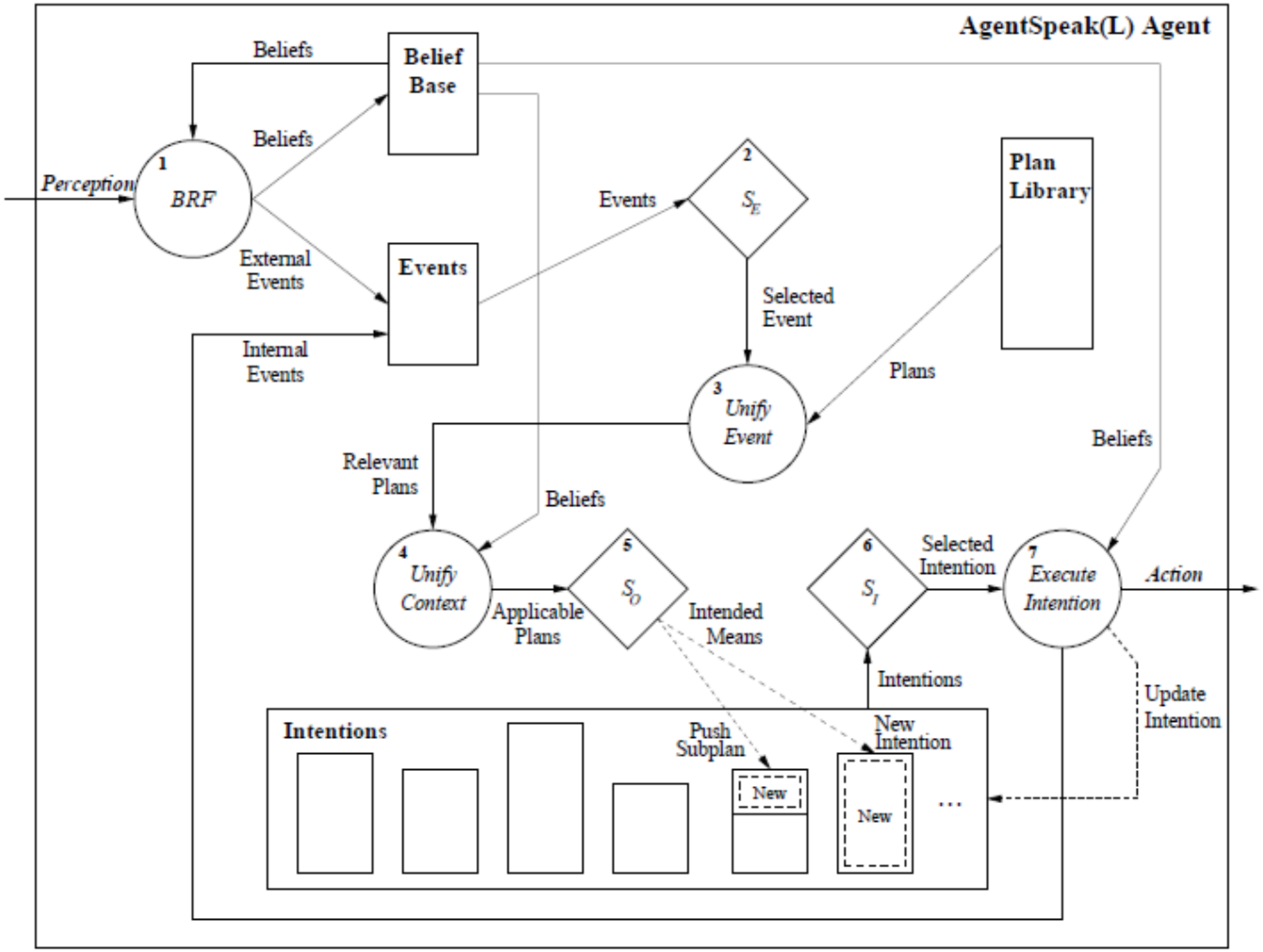


BDI - agents

```
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

- Implementácia BDI Agentov



AgentSpeak

Jazyk tvoria

- Premenné X
- Funktory girl(X), ... (konštanty 1, 2, ...)
- Predikátové symboly observed(t)
- Logické spojky & | not <–
- Zátvorky nemáme, máme ; a .
- Kvantifikátory for () { … }
- prípadne aj znamienko rovnosti =

- Literál (grounded)

observed(girl(anicka))

- Term

Rozšírenia jazyka logiky

- čísla a aritmetické operátory A+1=B
- logické operátory & |
- nerovnosti >= <=
- zabudované predikáty: .atom .literal .string
- desires alebo goals: !find(anicka) ?beauty(anicka)
- plany + –
- počty #
- constraint-y : height(X) & X >= 150
- nepomenovaná premenná _ napr. height(_)

Prvky nemonotónnosti

- Zabudované procedúry manipulujúce formuly:

.add_plan	pridaj do plans
.abolish	odstráň z beliefs
.asserta	pridaj do beliefs
.broadcast	pošli správu všetkým
.create_agent	spusti nového agenta
.date	získaj aktuálny dátum
.time	získaj aktuálny čas
.desire	pridaj do desires
.kill_agent	zabi bežiaceho agenta
.send	pošli správu jednému
.wait	počkaj daný čas

Komunikácia medzi agentmi

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

.broadcast(tell, "Hallo").

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

AgentSpeak používa KQML

KQML - ACL

AgentSpeak dokáže taktiež prekladať ACL na KQML a opačne

Napríklad inform v ACL je tell v KQML

Názvy performatívov sú tu žiaľ dôležité, lebo AgentSpeak má na ne zabudované reakcie, napr keď prijme tell, tak obsah, ak sa dá, vloží do 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 v AgentSpeak-u

- Programujeme hlavne tým, že do agenta vložíme určité plány

Jason

interpreter AgentSpeaku napísaný v Jave

Dokáže kooperovať s JADE

Podporuje viacero tzv. infraštruktúr (simulátor v štvorčekovom svete, JADE agenti, ...)

Jomi F. Hübner and Rafael H. Bordini, 2004

Na čo je to dobré?

Aby sme mohli celú logiku správania sa agenta a spôsob jeho komunikácie s ostanými napísať v AgentSpeaku, čo je predsa len o niečo čitateľnejšie, než keď to napíšeme v Jave.

Môžeme využiť existujúci scenár zapísaný v AgentSpeaku a rozvíjať ho.

Ako je možné implementovať v AgentSpeak-u prepojenie agenta na nejaký svet v ktorom operuje?

Priamo to nie je možné.

Preto máme veľa možností (hlavná je napísať vlastnú zabudovanú akciu) ako v Jave nakódovať aby sa:

- vnímanie agenta menilo na logické formuly
- vykonávanie akcii agenta realizovalo ako interpretácia logických formúl

Ide tu teda o: Kognitivizmus

- Percepcia a akcia sú oddelené a to práve kognitívnym podsystémom, v ktorom vládne AgentSpeak

Problémy

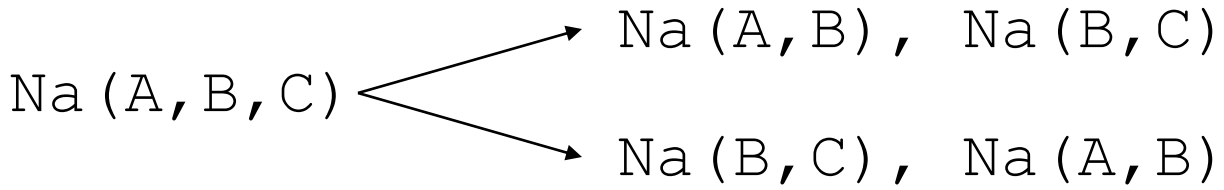
- A sú s tým teda aj spojené všetky problémy kognitivizmu

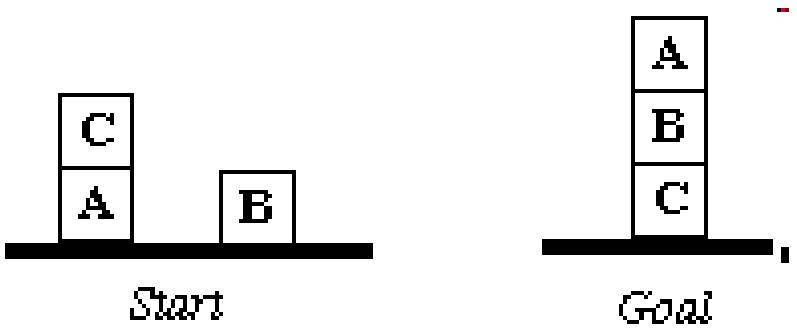
Problém rámca

- Keď kŕmime ľubovoľný inferenčný mechanizmus, zabúdame sémantiku a ďalej pracujeme s informáciami načisto syntakticky.
- Keď ich dáme na jednu hromadu, tak väčšinu času odvodzovania strávime tým, že sa pokúšame do súvisu uviesť veci, ktoré žiadny súvis nemajú („má zelená farba steny vplyv na výbuch bomby?“)
- Problém rámca: dajú sa vybrať kŕmené dáta tak, aby to už fungovalo a ešte sa to stíhalo ?

Sussmanova anomália

Ukazuje, že riešenie problémov rozkladom na podproblémy (neprekladané plánovanie), nevedie k optimálnemu riešeniu





Riešenie: hybridný systém

- Jeden tzv. deliberatívny agent (kognitivizmus)
  +
  viacero tzv. reaktívnych agentov (postkognitivizmus)
  =
  použijeme vždy ten spôsob, ktorý sa nám ľahšie urobí

Ale: komunikácia môže byť OK

- Pri komunikácii medzi agentmi v AgentSpeaku prevod formúl na komunikovanú výpoveď a späť netreba, pokiaľ komunikujú formulami
- Agenty v AgentSpeak-u sa vedia rozprávať rovnakým jazykom v akom myslia.
- Pošleme:
  .send(receiver,tell,byt(anicka,pekny),noid).
- z čoho u príjemcu vznikne Belief:

  byt(anicka,pekny)

Komunikačné akty

- Až tu vidíme, prečo vlastne potrebujeme viac komunikačných aktov:
  obsah, čo prinesie tell, bude zaradený do Beliefs
  zatiaľ čo obsah, ktorý prinesie ask-one bude namatchovaný na Beliefs a výsledok bude odpovedaný senderovi.