Kooperáció és intelligencia

Mit tanultunk eddig?
BDI ágensek, kísérletek

MI

Ágens: érzékelés – következtetés – cselekvés

Ágensarchitektúrák – beprogramozott / következtetett cselekvés (hibrid …)

Ágensprogram – jó, optimális cselekvés szelektor

erőforrás-adottságok, környezeti adottságok

Ágensprogram – ágens architektúra – környezet : együttes hatása

KTR

Környezet = több ágens – hasznos --- kooperáció

- ellenség ---- konfliktus (versengés)

Szervezet – nyílt/zárt, típusok: …, Hatása: komplexitás csökkenés

Kommunikáció

Protokollok - kooperatív

- konfliktust feloldó

„szabványosított” nyelvek: ACL

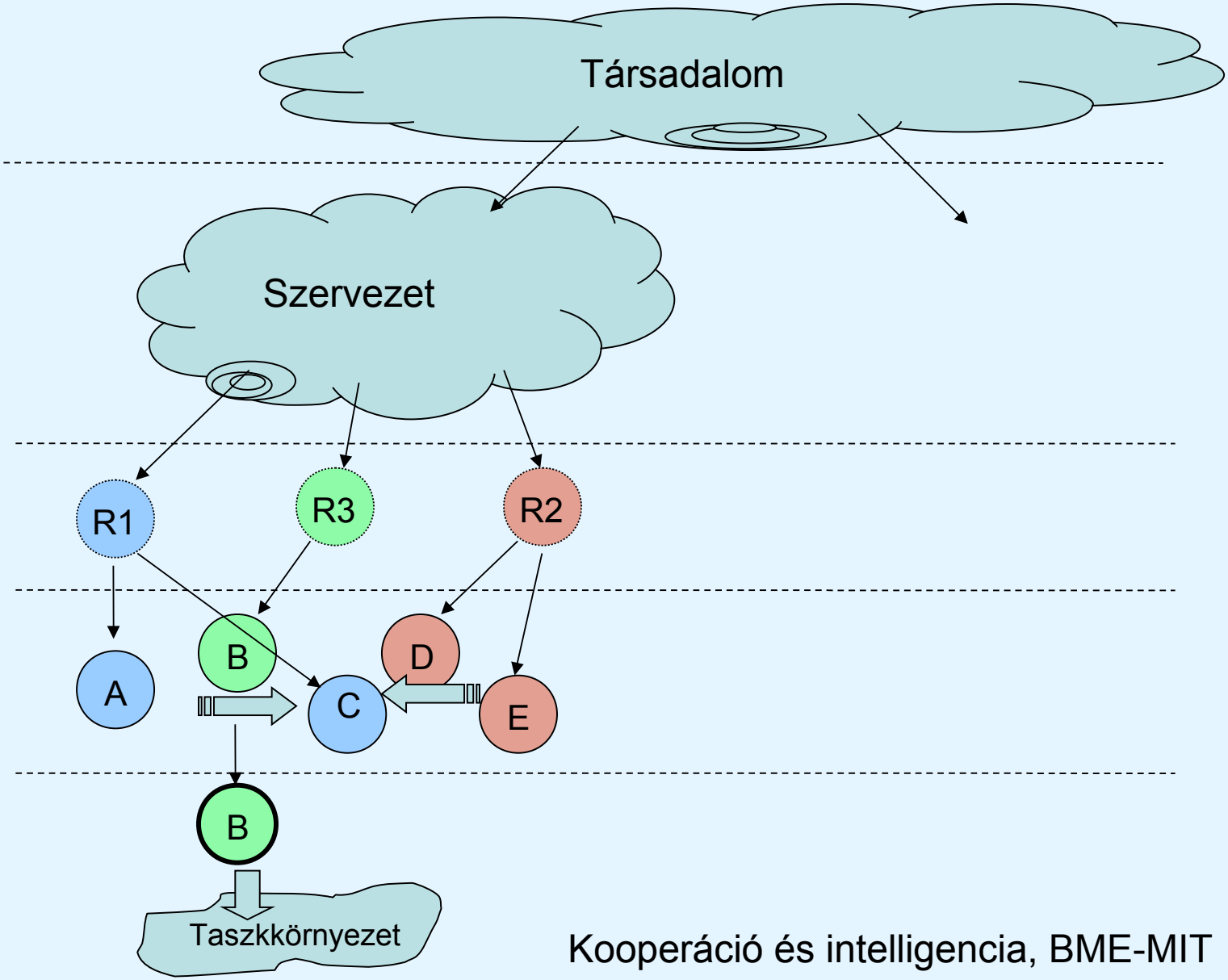
Háttér: Tny, beszéd/ szólás aktus

KQML: „hagyma-szerkezet”, szemantikai definiáltság

FIPA ACL – közösség platform

- nyílt/zárt mechanizmus, ACL, protokollok, ...

Én + mások = szervezet



Ágens architektúrák

Reaktív, Deliberatív (célorientált), Hibrid, …

BDI – **Belief, Desire, Intention** (gyakorlati humán következtetés, 1987)
ágens egy célorientált entitás, racionálisan cselekszik.

Belief – Hiedelem – **információs attitűd**,
az ágens által rendelkezett információk reprezentálása:
környezet reprezentációja, cselekvésenként frissített

Desire (Goal) - Kívánalom (cél) – **motivációs attitűd**,
célállapot explicit reprezentációja – a cselekvés
végrehajtásának az oka, a célokkal kapcsolatos prioritások,
kompromisszumok.

Intention (Plan) – Szándék (terv) – **deliberatív attitűd**,
a cél elérésének jelenlegi eszközei → **a jövőbeli cselekvések**

Kooperáció és Intelligencia – intelligens ágensekről tovább

BDI ágensek programozása

- Sok kis HF, egy nagy HF

Tanulás többágenses szervezetekben

Többágenses szervezetek modellezése – **játékelméleti** modellek

Többágenses szervezetek szociális egyetértése – **szavazáselmélet**

Többágenses szervezetek munkamegosztása – **aukciók** és közös **tervek**

Fogalmi rendszerek kezelése – **ontológiák** és leíró logikák

Kooperáció és Intelligencia – pontok … pontok

- 8 x kis HF (1-2 órás munka): 0 … 4 pont/HF, min. 5 pont
- egy nagy HF (2 hetes munka): 0 … 40 pont, min. 40%
- (ZH: 0 … 50 pont, min. 40%, tovább megy a pont-min.)
- V: 0 … 50 pont, min. 40%

Aláírás: minden legalább min.

Jegy:
Min. teljesítés: 5 + 16 + (0) + 20 = 41 (41)
Max. teljesítés: 32 + 40 + (30) + 50 = 152 (122)
„Jó” teljesítés: 16 + 30 + (20) + 40 = 106 (86)

- 40	elégtelen	(ZH nélkül)
41–69	elégséges	41-59
70–99	közepes	60-79
100–125	jó	80-99
126-	jeles	100-

BDI ágensek programozási nyelvei – AgentSpeak(L)

Ágens viselkedése:

hiedelmi állapot – saját modellje, környezet modellje,
más ágensek modellje
kívánalmak – a kívánatos hiedelmi állapotok
szándékok – végrehajtásra kiválasztott, vagy már részben végre is hajtott tervek, melyek további cselekvései meghatározzák az ágens közeljövőbeli viselkedését

Ágens AgentSpeak specifikációja:

hiedelmek halmaza (logikai jellegű tények)
tervek halmaza (kontextus-érzékeny, esemény-triggerelt receptek a hierarchikusan dekomponálható célok elérésére, elemei cél-orientált cselekvések).

AgentSpeak(L) – A programozás elemei

hiedelem atom – elsőrendű predikátum.

cél – a rendszer állapota, amit ágens szeretne megvalósítani. Kétféle cél:

teljesítési célok – predikátumok “!” prefixxel.

ágens egy olyan világállapotot el szeretne érni, amelyben a jelzett predikátum igaz lesz.

Programozásban a résztervek végrehajtását indítják.

teszt célok – predikátumok “?” prefixxel.

Illeszkednek (egyesítés) az ágens megfelelő hiedelmeire.

Kudarcot adnak vissza, ha megfelelő egyesítés nem lehetséges.

triggerelő esemény – *megadja*, mely esemény indíthat egy terv végrehajtását. Egy esemény lehet:

belső – amikor szükség van egy részcél teljesítésére

külső – hiedelem-frissítésből generált a környezet érzékeléséből adódóan

triggerelő esemény két típusa:

a hiedelmek és célok (attitűdök) ***hozzáadása*** (‘+’) **vagy *törlése*** (‘-’)

AgentSpeak(L) – A programozás elemei

Ágens viselkedése: receptkönyvtár (“tervek”) =
reaktív tervvégrehajtó rendszer

Terv:

Elérendő cél	:	Engedélyező kontextus	←	Tényleges terv lépései	.

Cél: +!goal, +?goal, −!goal, −?goal, +belief, −belief **teljesítési célok**, **teszt célok**

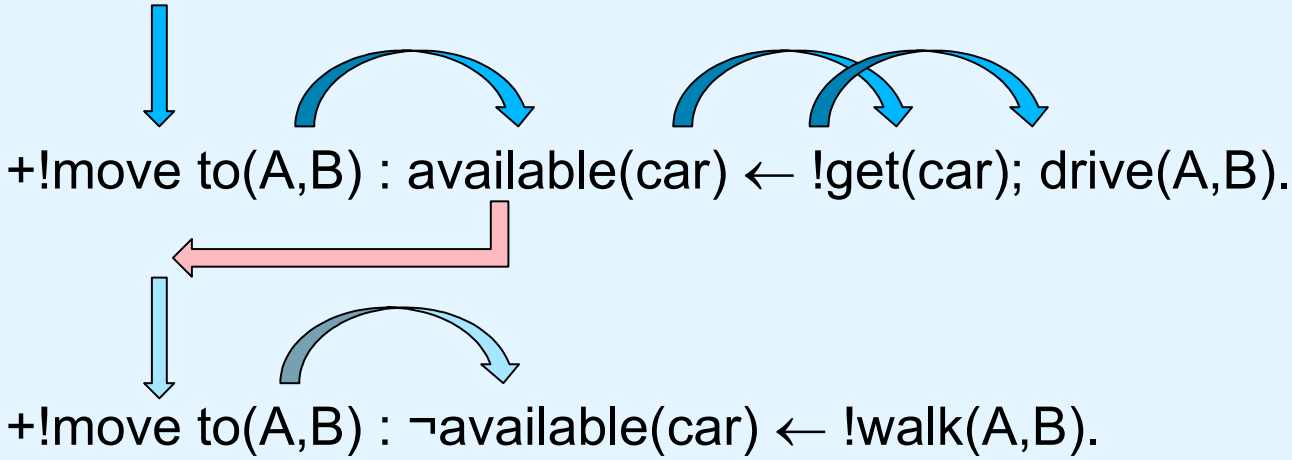
Kontextus: belief | Context ∧ Context | Context ∨ Context | ¬Context | ∃x.Context

Terv lépései: action | +belief | −belief | ?Context | !event | Plan; Plan

AgentSpeak(L) – A programozás elemei

Végrehajtási (interpreter) ciklus:

1. esemény aktivizálódik (új elfogadott cél, új érzékelés, lényeges változás a hiedelmekben, stb.)
2. eseményt kezelő tervek kikeresése (***releváns tervek***)
3. igaz kontextussal rendelkező terv **alkalmazható**, alkalmazható terv kiválasztása, törzsének lefuttatása
4. ha a terv kiválasztása kudarcba fullad, alternatív alkalmazható terv futtatása, amíg van választás, különben a kudarc tova terjesztése.



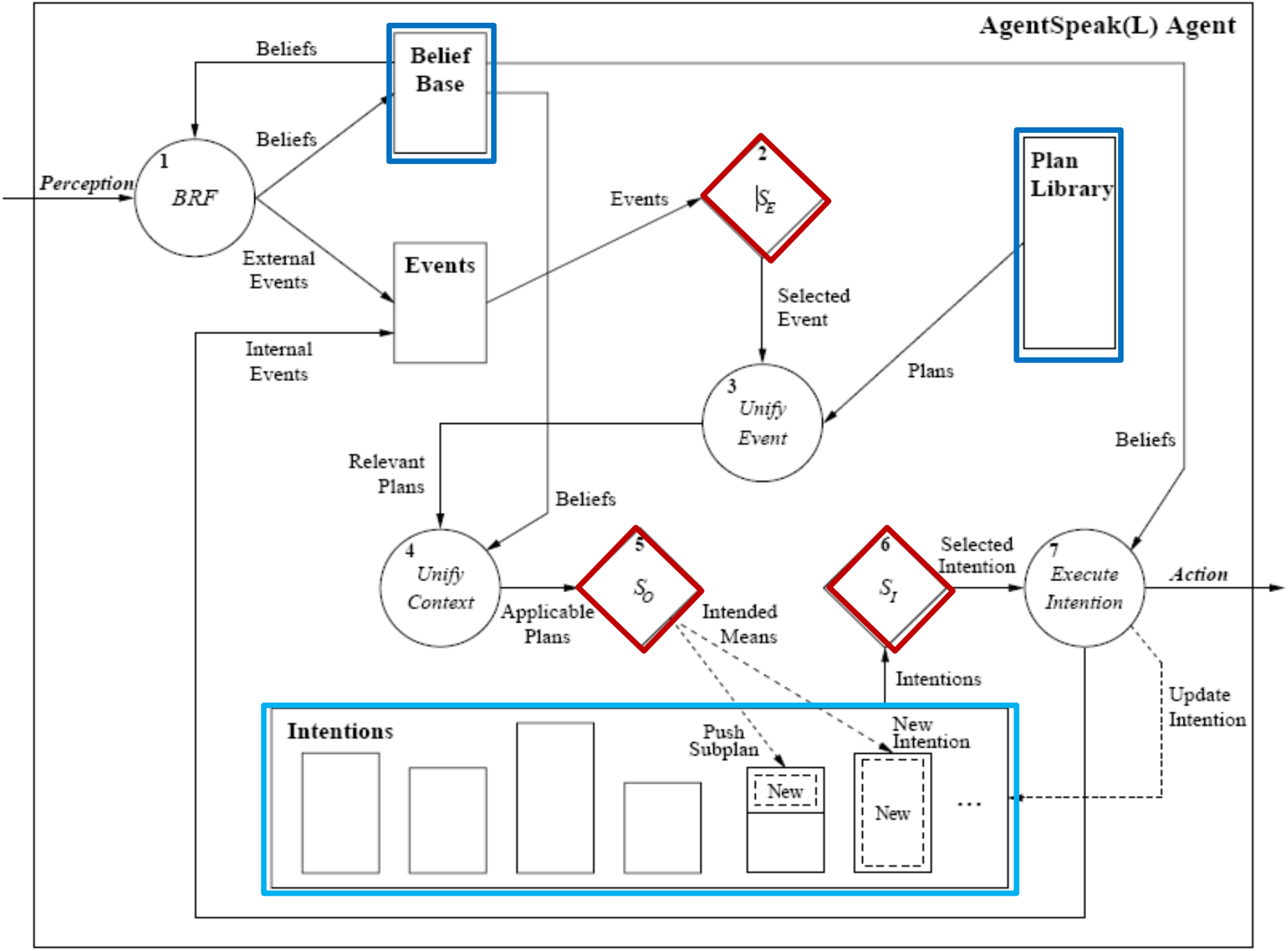
AgentSpeak(L) – A programozás elemei

Szándékok

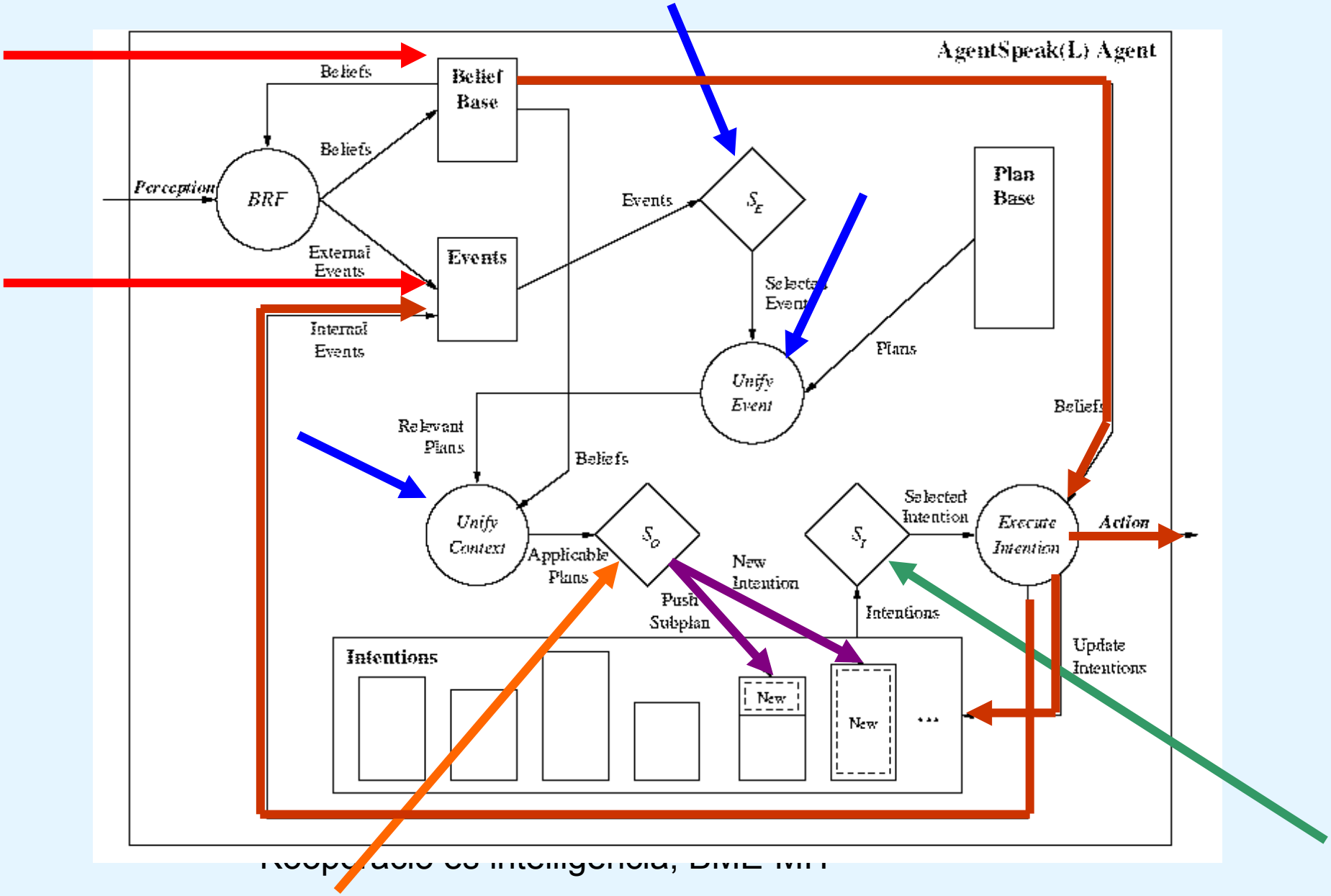
- a végrehajtásra megválasztott tervek
- szándékok végrehajtása lépésenként
- egy-egy lépés a hiedelmeket vagy lekérdezi, vagy megváltoztatja
- cselekvések eszközelése a külső környezeten(-ben)
- végrehajtás felfüggesztése, amíg a szükséges feltételek be nem állnak
- új célok feladása
- egy lépés generálhat új eseményeket és azok indíthatnak új szándékokat
- szándék teljesül, ha minden lépése sikeresen bekövetkezik
- kudarc, ha a feltételek nem teljesülnek, cselekvések hibával térnek vissza

AgentSpeak(L) **interpretere** = menedzsment

- események halmaza
- szándékok halmaza
- szelektor függvények (ágens-specifikus):
  - **SE** esemény szelektor
  - **SO** opció (alkalmazható terv) szelektor
  - **SI** szándék szelektor


AgentSpeak(L) Agent
Beliefs
Belief Base
Beliefs
Perception
1
BRF
External Events
Events
Events
2
S_E
Plan Library
Selected Event
Internal Events
Plans
3
Unify Event
Beliefs
Relevant Plans
Beliefs
4
Unify Context
5
S_O
Applicable Plans
Intended Means
6
S_I
Selected Intention
7
Execute Intention
Action
Intentions
Update Intention
Intentions
Push Subplan
New Intention
New
New
...

AgentSpeak(L) – A programozás elemei



Példa: 3 ágens aukciózik

ag1.asl

Mindig 6-tal fogad

```
+auction(N)[source(S)]: true <- .send(S, tell, place_bid(N,6)).
```

ag2.asl

Általában 4-gyel fogad,
de ha ag3-mal áll össze, akkor 0-át

```
default_bid_value(4).
ally(ag3).
```

```
+auction(N)[source(S)]: not alliance <-  ?default_bid_value(B);
                                         .send(S, tell, place_bid(N,B)).
+auction(N)[source(S)]: alliance <- .send(S, tell, place_bid(N,0)).
+alliance[source(A)]: .my_name(I) & ally(A) <- .print("Alliance proposed by ", A);
                                               ?default_bid_value(B);
                                                .send(A,tell,bid(I,B));
                                                .send(A,tell,alliance(A,I)).
```

Példa: 3 ágens aukciozik

ag3.asl

Magabiztosan 3-mal indul, de ha 3 aukciót veszít, akkor szövetséget ajánl ag2-nek és átveszi az ő licitjét is.

```
default_bid_value(3).
ally(ag2).
threshold(3).
```

```
+auction(N)[source(S)]: (threshold(T) & N<T) | (.my_name(I) & winner(I)
                        & ally(A) & not alliance(I,A)) <- !bid_normally(S,N).
+auction(N)[source(S)]: .my_name(I) & not winner(I) & ally(A) & not alliance(I,A)
          <- !alliance(A); !bid_normally(S,N).
+auction(N)[source(S)]: alliance(_,A)
          <- ?default_bid_value(B); ?bid(A,C); .send(S, tell, place_bid(N,B+C)).
+!bid_normally(S,N): true  <- ?default_bid_value(B); .send(S, tell, place_bid(N,B)).

@prop_alliance[breakpoint]
+!alliance(A): true  <- .send(A,tell,alliance).
```

Példa: 3 ágens aukciozik

Auctioner.asl Indít aukciókat, meghirdeti a győztest

```
all_bids_received(N) :- .count(place_bid(N,_),3).
```

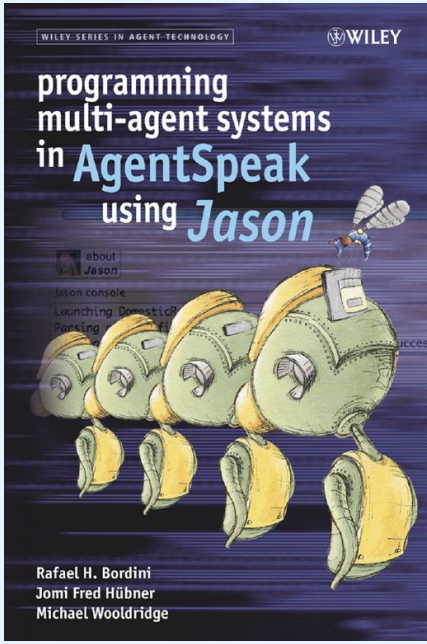
```
+!start_auction(N): true  <-  -+auction(N); -+winner(N, none, 0);
                              .broadcast(tell, auction(N)).
                                                  // Ezt a célt a GUI állítja elő
@pb1[atomic]
+place_bid(N,V)[source(S)]: auction(N) & winner(N,CurWin,CurVl) & V>CurVl
          <- -winner(N,CurWin,CurVl); +winner(N,S,V); !check_end(N).
@pb2[atomic]
+place_bid(N,_): true  <- !check_end(N).

+!check_end(N): all_bids_received(N) & winner(N,W,Vl)
     <- .print("Winner is ",W," with ", Vl); show_winner(N,W);
        .broadcast(tell, winner(W)); .abolish(place_bid(N,_)).
+!check_end(_).
```

Jason – AgentSpeak platform Javaban, kiterjesztésekkel

AgentSpeak(L) teljes erejű interpretere
sok kiterjesztés, függvény … hatékony ágensprogramozás
többágenses rendszer több hoszton konfigurálható
(+ Jade, ...)
Java implementáció
Open Source GNU LGPL alatt:
http://jason.sourceforge.net/

Jason – AgentSpeak platform Javaban, kiterjesztésekkel



Environment: percepts, environment actions
ilternal actions, beliefs, goals

- KQML alapú inter-ágens kommunikáció
  tell, untell, achieve, unachieve
  tellHow, untellHow, askIf, askAll, askHow

1’ rendű nyelv, nincs modalitás

Terv címkék, hiedelmek annotálása
hiedelem [source(Ag)], hiedelem [source(percept)]
hiedelem [source(self)]

Szabadon kiterjeszthető, felhasználói szinten módosítható (Java)

Tervkudarc kezelése: +!g: ... kudarca = -!g: ... bővítés
permanens cél: -!g: true ⫴ !g.

CWA – **OWA**: negálás, mint kudarc, erős negálás: (John McCarthy)
Iskolabusz haladhat át egy vasúti átjárón,
ha a vonat nem közeledik.

Át ← not Vonatjön (ha nem tudjuk, hogy a vonat közeledik)
Át ← ¬Vonatjön (ha tudjuk, hogy a vonat nem közeledik)

Belső (beépített) cselekvések (több)

BDI desire, drop_desire, drop_all_desires, intend, drop_intention, drop_all_intentions, current_intention, drop_event, drop_all_events, succeed_goal, fail_goal, suspend, resume

BB abolish, findall, count

Plan add_plan, remove_plan, plan_label, relevant_plans

Comm. send, broadcast, my_name

Lists and Sets
member, length, concat, delete, reverse, nth, max, min
sort, list, difference, intersection, union

String length, concat, delete: delete, reverse, substring, string, term2string

Execution if, while, for

Meta progr. atom, structure, literal, list, ground, number, string
add_annot, add_nested_source

Misc. at, wait, create_agent, kill_agent, stopMAS, date, time
fail, perceive, range…

jEdit - HeathrowRobots.mas2j

File Edit Search Markers Folding View Utilities Macros Plugins Help

HeathrowRobots.mas2j (C:\jason\Jason-1.2\examples\airport\)

HeathrowRobots.mas2j

```
13 MAS heathrow {
14
15     infrastructure: Centralised
16
17     environment:
18         HeathrowEnv
19
20     agents:
21         mds agentClass mds.MDSAgent
22             #5;
23
24         cph agentArchClass cph.CPHAgArch
25             agentClass cph.CPHAgent
```

about *Jason*

Jason console

Further Jason functionalities are available in the menu Plugin->Jason.

Project agents

mds agentClass
cph agentArchCl
hd #3:

Error List | Jason IDE

11,3 41% (mas2j,none,Cp1252) - - - - U 7/9Mb

:: Jason Mind Inspector :: cycle 22 ::

Agents

r2
r1

Agent Inspection

Inspection of agent **r1** (cycle #12)

- Beliefs

pos(back,3,0)[source(self)]
pos(r1,3,0)[source(percept)]
pos(r2,3,3)[source(percept)]
garbage(r1)[source(percept)]

- Events

Sel	Trigger	Intention
X	+!ensure_pick(garb)	4

+ Options

- Intentions

Sel	Id	Pen	Intended Means Stack (show details)	
X	4		+!ensure_pick(S)	{ S = garb }
			+!take(S,L)	{ S = garb, L = r2 }
			+!carry_to(R)	{ R = r2, Y = 0, X = 3 }
			+garbage(r1)[source(percept)]	

Actions

Pend	Feed	Sel	Term	Result	Intention
X		X	pick(garb)	false	4

Agent History

Cycle 0 10 20 Cycle 22

Run 5 cycle(s) for all agents view as: html