Agentes BDI e AgentSpeak(L)

October 27, 2006

Definição de agente

Um *agente* é um sistema de hardware ou software situado em um ambiente que foi projetado para atingir seus propósitos e que apresenta as seguintes propriedades:

- Autonomia
- Reatividade
- Proatividade
- Interação social

Abordagens mais fortes da noção de agência

Seguindo uma abordagem mais forte de agência, um agente também pode apresentar as seguintes propriedades:

- Noções *Mentalísticas*:
  - Crenças (Beliefs)
  - Desejos (Desires)
  - Intenções (Intentions)
- Noções *Emocionais*:
  - Confiança (Trust)
  - Amizade (Friendship)
  - Desconfiança (Suspiciousness)

Agentes BDI

- Sistemas situados em ambientes que mudam continuamente
- Entradas também são percebidas continuamente
- Afetam o ambiente através de ações

O agente necessita decidir (selecionar) que ações deve ser executadas, num dado instante de tempo, a partir de um conjunto de alternativas e opções.

Uma ***função de seleção*** permite que o agente alcançe seus objetivos, levando em conta:

- os recursos computacionais do agente
- as características do ambiente onde ele está situado.

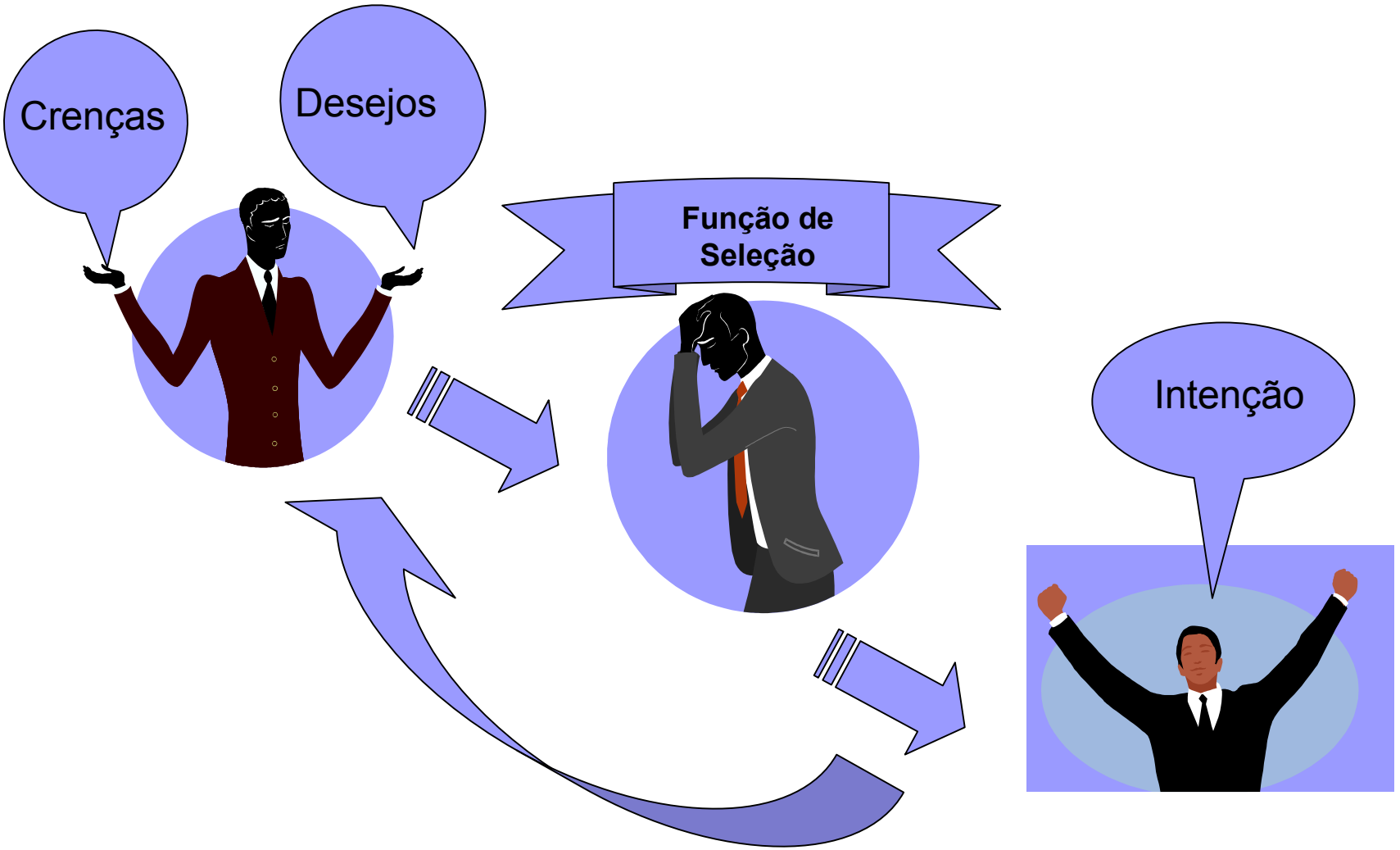
Agentes BDI

- Esta função necessita de dois tipos de dados:
- **Crenças (beliefs):**
  - representas as propriedades dos ambiente
  - são atualizadas apropriadamente após cada percepção (ação sensora).
  - formam o componente informativo do agente.
- **Desejos (desires)**
  - contém as informações sobre que objetivos deveriam ser buscados pelo agente, incluindo prioridades e retornos destes objetivos
  - representam o estado ***motivacional*** do agente.

Agentes BDI

- **Intenções (intentions)**
  - representam o curso de ação atual do agente (a saída da mais recente chamada para a afunção de seleção)
  - formam o componente *deliberativo* do agente.

Agents BDI



AgentSpeak(L)

- Linguagem de programação lógica que se propõe a diminuir a diferença entre teoria e prática na criação de agentes BDI
- Baseada numa correspondência um-para-um entre a Teoria de Modelos, Teoria de Provas e o Interpretador Abstrato (semântica operacional).

  - Extensão natural da programaçàо em lógica para a arquitetura BDI
  - Fornece um framework abstrato elegante para a programação de agentes BDI.
  - Baseada em uma linguagem de primeira ordem restrita, mas que suporta eventos e ações.
  - O comportamento do agente (sua interação com o ambiente) é ditado pelo programa escrito em AgentSpeak(L).

AgentSpeak(L) e B - D - I

- O ***estado de crenças*** (***belief state***) do agente forma o estado atual do agentes, que incorpora um modelo do próprio agente, do ambiente ao seu redor e dos outros agents com que se relaciona.
- Os ***desejos*** (***desires***) do agente são os estados que o agente quer atingir baseado tanto em estímulos internos quanto externos.
- Quando um agente se compromente com um conjunto particular de planos para atingir um determinado objetivo, então estes planos parcialmente instanciados são referidos como a intenção associada aquele objetivo $\Rightarrow$ Portanto, ***intenções*** (***intentions***) são planos ativos que o agente adota para atingir sua metas.

Noções Básicas de AgentSpeak(L)

- A especificação de um agente em AgentSpeak(L) consiste de:
  - um conjunto base de ***crenças***
    - fatos em termos de programação em lógica
  - um conjunto de ***planos***.
    - “receitas” sensíveis ao contexto e ativadas por eventos que permitem uma decomposição hierárquica dos objetivos e a execução de ações com o propósito de atingir uma meta.

Noções Básicas de AgentSpeak(L)

- ***átomo de crença (belief atom)***
  - predicado de primeira-ordem em sua notação usual
  - um átomo de crença ou sua negação são denominados de ***literais de crença*** (***belief literals***).

Noções Básicas de AgentSpeak(L)

- ***objetivo (goal)***
  - o estado do sistema que o agente quer atingir.
- dois tipos de objetivos (ou metas):
  - ***meta (objetivo) de realização (achievement goal)***
    - representados por predicados prefixados com o operador “!”
    - declara que o agente que o agente pretende atingir um determinado estado do mundo, onde o predicado associado é verdadeiro.
    - na prática, estes objetivos iniciam a execução de subplanos.
  - ***objetivo de teste***
    - representados por predicados prefixados com o operador‘?’
    - retornam a unificação do predicado associada com uma das crenças do agentes, falha caso nenhuma unificação seja encontrada.

Noções Básicas de AgentSpeak(L)

- ***evento de disparo (trigger)***
  - são eventos que podem iniciar a execução de um plano.
  - um *event* pode ser
    - interno, quando um objetivo secundário deve ser alcançado
    - externo, quando gerado pelo processo de atualização de crenças em resposta a percepções do ambiente.
  - dois tipos de eventos de trigger:
    - relacinadis a *inclusão* (‘+’) ou *exclusão* (‘-’) of atitudes (crenças ou objetivos).

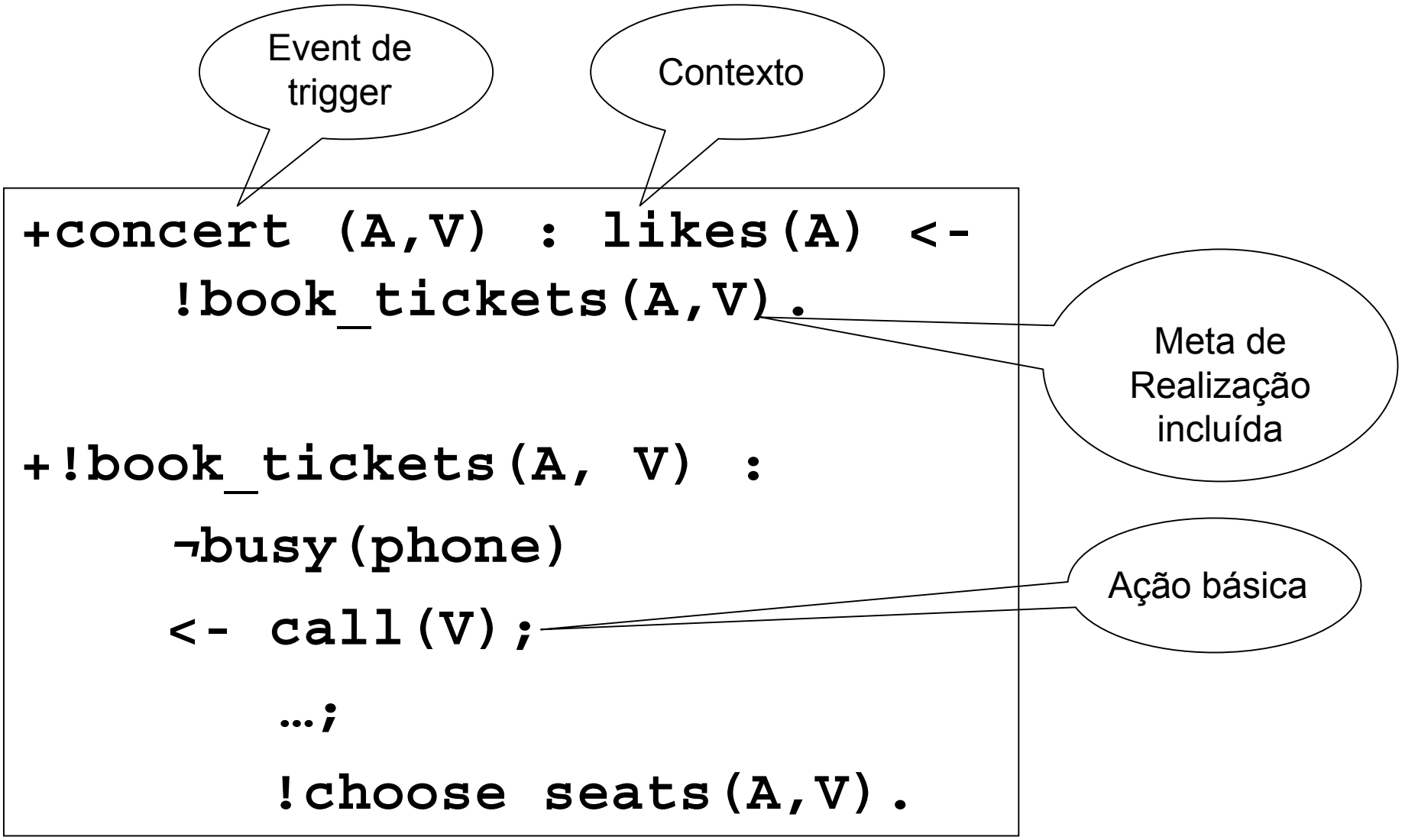
Noções Básicas de AgentSpeak(L)

- ***Planos***
  - se referem as ações básicas que um agente é capaz de executar no seu ambiente.

```
p ::= te : ct <- h
```

Onte:

- `te` *- evento de trigger (denota o propósito do plano)*
- `ct` *- uma conjunção de consultas de crenças representando um contexto.*
  - *O contexto deve ser uma consequência lógica das crenças atuais do agente, para o plano ser aplicável.*
- `h` *- uma sequência de ações básicas ou (sub)objetivos que o agente tem que atingir quando o plano, se aplicável, é selecionado para execução.*


Event de trigger
Contexto
+concert (A,V) : likes(A) <-
!book_tickets(A,V).
+!book_tickets(A, V) :
¬busy(phone)
<- call(V);
…;
!choose seats(A,V).
Meta de Realização incluída
Ação básica

Noções Básicas de AgentSpeak(L)

- ***Intenções***
  - Planos que o agente selecionou para execução.
  - Intenções são executadas um passo por vez.
  - Um determinado passo pode:
    - consultar ou alterar as crenças
    - executar ações no mundo externo
    - suspender a execução (do agente) até uma certa condição é satisfeita
    - submeter novas metas (objetivos).
  - As operações executadas num passo pode gerar novos eventos, que podem iniciar novas intenções.
  - Uma intenção tem sucesso, quando todos os seus passos são completados. Ela falha quando certas condições ocorrem ou quando ações reportam erros.

Sintaxe de AgentSpeak(L)

ag	::=	bs ps	
bs	::=	at_1. … at_n.	(n≥0)
at	::=	P(t_1, … t_n)	(n≥0)
ps	::=	p_1 … p_n	(n≥1)
p	::=	te : ct <- h.	
te	::=	+at | -at | +g | -g	
ct	::=	true | l_1 & … & l_n	(n≥1)
h	::=	true | f_1 ; … ; f_n	(n≥1)
l	::=	at | not (at) | ~at	
f	::=	A(t_1, … t_n) | g | u	(n≥0)
g	::=	!at | ?at	
u	::=	+at | -at	

Semântica Informal de AgentSpeak(L)

- O interpretador de AgentSpeak(L) deve gerenciar:
  - Um conjunto de eventos
  - Um conjunto de intenções
  - Três funções de seleção.

Semântica Informal de AgentSpeak(L): Eventos

- ***Eventos*** que podem iniciar a execução de planos (trigger) podem ser:

  externos**,** quando originários da percepção do agente $\Rightarrow$ originários da inclusão ou exclusão de crenças ocasionada pelas percepções do agentes. Eventos externos criam novas intenções**.**

  - ***internos*****,** quando são gerados pelo próprio agente através da execução de um plano $\Rightarrow$ um (sub) objetivo em um plano gera um evento to tipo “inclusão de uma nova meta de realização”

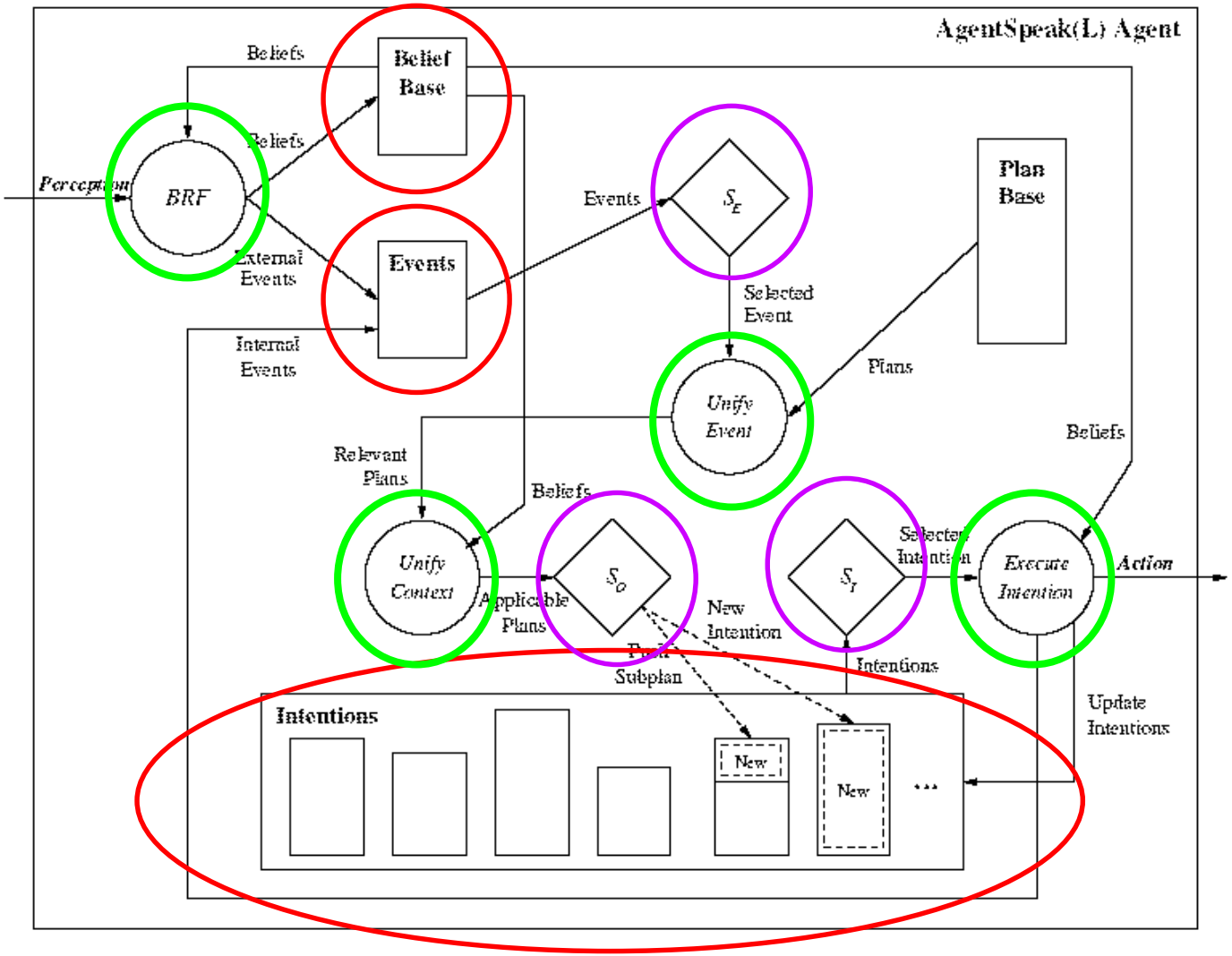
Semântica Informal de AgentSpeak(L): Intenções

- ***Intenções*** são cursos particulares de ação que um agente se comprometeu a fim de tratar certos eventos. Cada intenção é formada por uma *pilha* de planos instanciados (com execução habilitada).

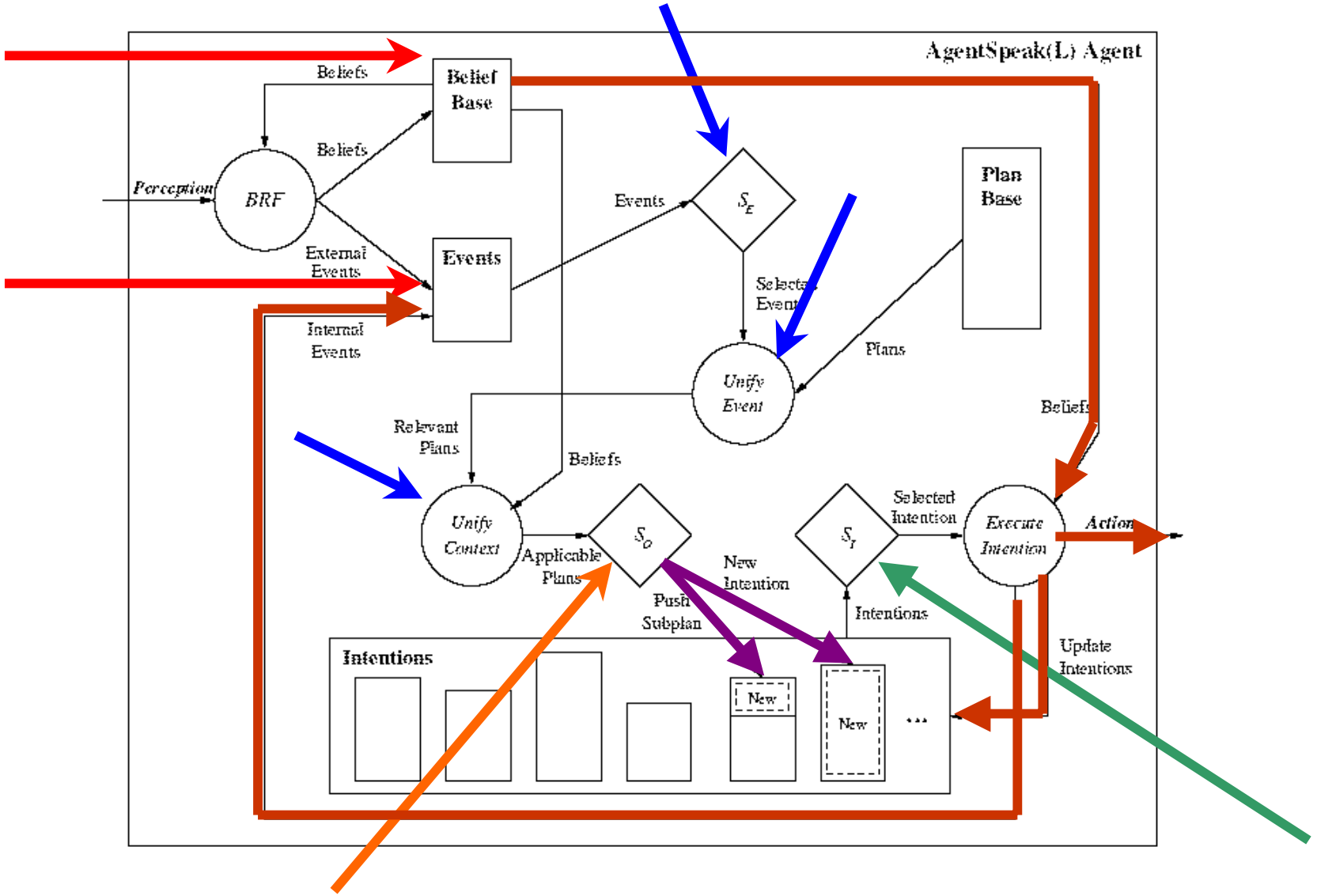
Semântica Informal de AgentSpeak(L): Funções de Seleção

- **SE** (a função de seleção de eventos)
  - seleciona um único evento de um conjunto de eventos
- **SO**
  - seleciona uma "opção" (isto é, um particular plano aplicável) de um conjunto de planos aplicáveis
- **SI**
  - seleciona uma intenção particular de um conjunto de intenções.
- *As funções de seleção são específicas do agente, no sentido que elas poderiam selecionar elementos baseadas nas características do agente.*

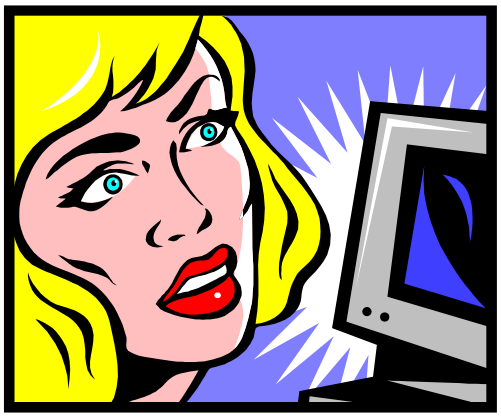
Semântica Informal de AgentSpeak(L)



Semântica Informal de AgentSpeak(L)



Exemplo de Programação em AgentSpeak(L)



ALICE

- Durante o almoço, redirecione as chamadas para a Carla.
- Quando estou ocupado, chamadas recebidas dos colegas devem ser redirecionadas para a Denise.

Exemplo em AgentSpeak(L): Crenças

```
user(alice).
user(bob).
user(carla).
user(denise).
~status(alice, idle).
status(bob, idle).
colleague(bob).
lunch_time(“11:30”).
```

Exemplo em AgentSpeak(L): Planos

```
user(alice).
user(bob).
user(carla).
user(denise).
~status(alice, idle).
status(bob, idle).
colleague(bob).
lunch_time("11:30").
```

"Durante o almoço, redirecione as chamadas para a Carla".

+invite(X, alice) : lunch_time(T) ← !call_forward(alice, X, carla). (p1)

"Quando estou ocupado, chamadas recebidas dos colegas devem ser redirecionadas para a Denise".

+invite(X, alice) : colleague(X) ← call_forward_busy(alice,X,denise). (p2)

+invite(X, Y): true ← connect(X,Y). (p3)

Exemplo em AgentSpeak(L): Planos

```
user(alice).
user(bob).
user(carla).
user(denise).
~status(alice, idle).
status(bob, idle).
colleague(bob).
lunch_time("11:30").
+invite(X, alice) : lunch_time(T) ← !call_forward(alice, X, carla).      (p1)
+invite(X, alice) : colleague(X)  ← !call_forward_busy(alice,X,denise).  (p2)
+invite(X, Y): true               ← connect(X,Y).                        (p3)
```

```
+!call_forward(X, From, To) : invite(From, X)
  ← +invite(From, To), - invite(From,X)                    (p4)
```

```
+!call_forvard_busy(Y, From, To) : invite(From, Y)&
  not(status(Y, idle)))
      ← +invite(From, To), - invite(From,Y).      (p5)
```

Exemplo em AgentSpeak(L)

```
user(alice).
user(bob).
user(carla).
user(denise).
~status(alice, idle).
status(bob, idle).
colleague(bob).
lunch_time("11:30").

+invite(X, alice) : lunch_time(T)
            ← !call_forward(alice, X, carla).          (p1)
+invite(X, alice) : colleague(X)
            ← call_forward_busy(alice,X,denise).       (p2)
+invite(X, Y): true   ←  connect(X,Y).                 (p3)
+!call_forward(X, From, To) : invite(From, X)
  ← +invite(From, To), - invite(From,X)                (p4)
+!call_forvard_busy(Y, From, To) :
         invite(From, Y)& not(status(Y, idle)))
      ← +invite(From, To), - invite(From,Y).           (p5)
```

Execução - 0

- um novo evento é percebido no ambiente:
  `-lunch_time("11:30")` (encerrou o horário do almoço).
- Não há plano para este evento, ele apenas é retirado da base de crenças do agente:

```
user(alice).
user(bob).
user(carla).
user(denise).
~status(alice, idle).
status(bob, idle).
colleague(bob).
```

Execução - 1

- um novo evento é percebido no ambiente: `+invite(bob, alice)` (há uma chamada para Alice de Bob).
- Há três planos *relevantes* para este evento (p1, p2 e p3)
  - este evento se encaixa como evento de trigger destes três planos.

Planos Relevantes	**Unificador**
`p1: +invite(X, alice) : lunch_time(T)` `  ←!call_forward(alice, X, carla)`	{X=bob}
`p2: +invite(X, alice) : colleague(bob)` `  ← !call_forward_busy(alice, X, denise).`	{X=bob}
`p3 : +invite(X, Y): true ← connect(X,Y).`	{Y=alice, X=bob}

Execução - 2

- somente o contexto do plano p2 é satisfeito - **`colleague(bob)`** => p2 é *aplicável*.
- uma nova intenção baseada neste plano é creada no conjunto de intenções, porque o evento foi externo, gerado pela percepção de uma ocorrência no ambiente.
- O plano inicia a ser executado. Ele adiciona um novo evento, desta vez interno: **`!call_forward_busy(alice,bob,denise).`**

ID Intenção	Pilha de Intenções	Unificador
1	`+invite(X,alice):colleague(X)` `<- !call_forward_busy(alice,X,denise)`	{X=bob}

Execução - 3

- um plano relevante para este evento é encontrado (p5):

Planos Relevantes	**Unificador**
`p5: +!call_forvard_busy(Y, From, To) : invite(From, Y) & not(status(Y, idle)) ← +invite(From, To), - invite(From,Y).`	{From=bob, Y=alice, To=denise}

- p5 tem a condição de contexto verdadeira, assim ele se torna um plano aplicável, que é empilhado sobre a pilha de intenções da Intenção 1

ID Intenções	**Pilha de Intenções**	**Unificador**
1	`+!call_forward_busy(Y,From,To) : invite(From,Y) & not status(Y,idle) <- +invite(From,To); -invite(From,Y)`	{From=bob, Y=alice, To=denise}
	`+invite(X,alice) : colleague(X) <- !call_forward_busy(alice,X,denise)`	{X=bob}

Execução - 4

- Um novo evento interno é criado, **`+invite(bob, denise).`**
- três planos relevantes são encontrados para este evento, p1, p2 and p3.
- Entretanto, somente p3 é aplicável, porque os outros não tem suas condições de contexto satisfeitas.
- O plano é empilhado sobre o topo da intenção atual.

ID Intenção	**Pilha de Intenções**	**Unificador**
1	`+invite(X,Y) : <- connect(X,Y)`	{Y=denise, X=bob}
	`+!call_forward_busy(Y,From,To) : invite(From,Y) & not status(Y,idle) <- +invite(From,To); -invite(From,Y)`	{From=bob, Y=alice, To=denise}
	`+invite(X,alice) : colleague(X) <- !call_forward_busy(alice,X,denise)`	{X=bob}

Execução - 5

- no topo da intenção há um plano cujo corpo contém uma ação.
- a ação é executada, `connect(bob, denise)` e removida da intenção.
- Quandos todas as fórmulas no corpo do plano são removidas (foram executadas), o plano inteiro é removido da intenção, e também a meta de realização que gerou o plano.

ID Intenção	Pilha de Intenções	Unificador
1	`+!call_forward_busy(Y,From,To) : invite(From,Y) & not status(Y,idle) <- -invite(From,Y)`	{From=bob, Y=alice, To=denise}
	`+invite(X,alice) : colleague(X) <- !call_forward_busy(alice,X,denise)`	{X=bob}

- Agora somente resta executar `-invite(bob, alice)` (remover este evento da base de crenças)
- Isto encerra este ciclo de execução, e o processo de inferência pode recomeçar, verificando o estado do ambiente e reagindo aos eventos.

Jason

- um interpretador completo e funcional para AgentSpeak(L)
- suporta várias extensões, provendo uma linguagem de programação para agentes bastante expressiva.
- permite a configuração de um sistema multi-agente rodar em vários *hosts*.
- implementado em Java (portanto é multi-plataforma)
- Fontes disponíveis (*Open Source*) e distribuídos sob licença GNU LGPL.
- http://jason.sourceforge.net/

Características de Jason

- fornece um IDE
- permite a execução de um sistema multiagente distribuído em uma rede
- permite a customização (em Java) das funções de seleção, funções de confiança (trust) e arquitetura geral do agente (incluindo percepções, ações, revisão de crenças e comunicação entre agentes)
- provê uma biblioteca de “ações internas” essenciais
- pode ser extendido de forma fácil e direta pela criação de novas ações internas, programadas em Java.

Conclusão

- AgentSpeak(L) tem muitas similaridades com programação em lógica tradicional (Prolog)
- é bastante intuituva para quem tem experiência em programação em lógica.
- tem uma notação compacta e agradável, que permite a criação de especificações de agentes BDI bastante elegantes.