Tecnologías Pregunta-Respuesta

Ejercicio E1

Solución de los ejercicios sobre Sintaxis de AgentSpeak

Departamento de Ingeniería de Sistemas Telemáticos http://moodle.dit.upm.es

- E1: Los planes h1, h2 y h3 se ejecutan cuando en el agente robot aparece la meta !has(owner, beer). (fácil)
 - Modifícalos para que sólo se ejecuten cuando el origen de la meta sea el agente owner.
 - ¿Crees que esto puede hacer que deje de funcionar el sistema? ¿Por qué?















UPM

Solución de ejercicios

Ejercicio E1

- En el archivo robot asl debemos modificar los planes @h1, @h2 y @h3 de manera que el triggering event sea específico para el agente owner
 - La fuente de que procede el evento Jason la marca automáticamente con la anotación source(owner)

Introducción a AgentSpeak

Ejercicio E1

```
@h1 +!has(owner,beer)[source(owner)] : ... <- ...</pre>
@h2 +!has(owner,beer)[source(owner)] : ... <-</pre>
@h3 +!has(owner,beer)[source(owner)] : ... <- ...</pre>
```





















- Efectivamente esto puede hacer que el sistema deje funcionar si el robot recibe la orden de llevar a cabo la meta ! has (owner, beer) de algún otro agente que no sea el agente owner, incluido de sí mismo.
 - Esto ocurre en la última linea del cuerpo del plan @a1

Ejercicio E1

- Para solucionar este error, debemos incluir automáticamente que la meta procede del agente owner
 - Debemos añadir la anotación source(owner)

```
@a1
+delivered(beer,Qtd,OrderId[source(supermarket)]
     <- +available (beer, fridge);
         !has (owner, beer) [source (owner)].
```









Introducción a AgentSpeak









Introducción a AgentSpeak

Ejercicio E2

- E2: Mejora el código del supermarket agent para tener en cuenta el stock de cerveza. Supón un stock inicial de 20 cervezas y disminuye dicho stock con cada envío al robot. Si no guedan suficientes cervezas
 - Informa al robot de que no cerveza (fácil)
 - Envía todas las que queden (medio)

Ejercicio E2

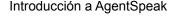
- Hay que modificar el archivo supermarket.acl
- Debemos tener en cuenta el stock de cerveza. para ello añadimos una nueva creencia al inicio
 - stock(beer, 20)
- Hay que considerar dos casos.
 - Si tenemos stock suficiente: @order1
 - Si no tenemos stock suficiente: @order2













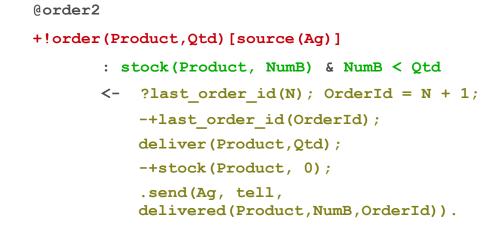






Ejercicio E2

```
@order1
+!order(Product,Qtd)[source(Aq)]
       : stock(Product, NumB) & NumB >= Qtd
       <- ?last order id(N); OrderId = N + 1;</pre>
           -+last order id(OrderId);
           deliver(Product,Otd);
           -+stock(Product, NumB-Qtd);
           .send(Aq, tell,
           delivered(Product,Qtd,OrderId)).
```











Introducción a AgentSpeak

Introducción a AgentSpeak







Introducción a AgentSpeak

10

Ejercicio E3

 E3: ¿Qué ocurre si se intercambian de orden los planes m2 y m1? (fácil)

En caso de que el código fuera

```
@m1 + !at(robot, P) : at(robot, P)
   <- true.
@m2 +!at(robot,P) : true
   <- move towards(P);
!at(robot, P).
```

¿Qué ocurre si se cambian ahora de orden?

Ejercicio E3

- En el primer caso no ocurre nada porque el contexto de los planes @m1 y @m2 son mutuamente excluyentes.
 - Es decir, no se pueden cumplir las circunstancias definidas en el context de ambos planes a la vez















12

- En el segundo caso, al incluir un plan con el context true antes que otro que se activa con el mismo triggering event, hace que el segundo nunca se active
 - Pues Jason, ante dos planes que se activan con el mismo evento y que ambos cumplen las condiciones definidas en el contexto escoge activar el primero de ellos

Ejercicio E4

• E4: Crea un nuevo agente supermarket con el mismo código del supemarket agent.

Modíficalos para que nada más iniciar la ejecución le envíen al robot el precio de la cerveza

```
.send(robot, tell, price(beer, 3))
```

Modifica el código del robot para que compre la cerveza del supermercado más barato (medio)









Introducción a AgentSpeak









Introducción a AgentSpeak

14

Ejercicio E4

- Debemos incluir dos agentes supermarket. Para ello:
 - O bien creamos otro archivo .asl en el que definimos el nuevo agente supermarket2.asl
 - O bien usamos las funcionalidades que proporciona Jason para crear varios agentes a partir del mismo fichero asl

• #2

Ejercicio E4

Modificamos el archivo de extensión mas2j

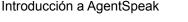
```
MAS domestic robot {
    environment: HouseEnv(qui)
    agents: robot;
            owner;
             supermarket
                  agentArchClass SupermarketArch #2;
```



















- Cada uno de los agentes supermercado se crean a partir del mismo fichero .asl.
 - De manera que para que ofrezcan un precio diferente debemos usar math.random(10)
 - Da un número aleatorio entre 0 y 10 con decimales
 - Para eliminar los decimales usamos math.round(P)

Ejercicio E4

En supermarket.asl

```
!send_price.

// tell the price
+!send_price : true
<- P = 5 + math.random(10);
    .send(robot, tell, price(beer, math.round(P))).</pre>
```









Introducción a AgentSpeak

Introducción a AgentSpeak







Introducción a AgentSpeak

18

Ejercicio E4

- El agente robot cada vez que recibe una oferta debe comprobar si es la más barata
 - La almacenamos en la creencia cheapest(Price, Agent)
- A la hora de hacer un pedido, usamos el agente almacenado como el más barato

Ejercicio E4

Para almacenar el precio en robot.asl

```
+price(beer, Price)[source(Ag)]
    : cheapest(Cheap, Seller) & Cheap > Price
    <- -+cheapest(Price, Ag);

+price(beer, Price)[source(Ag)]
    : not cheapest(_, _)
    <- +cheapest(Price, Ag);</pre>
```



















- Para hacer el pedido en robot.asl
 - Modificamos el plan @h2

@h2

```
+!has(owner,beer)
: not available(beer,fridge)
<- ?cheapest(beer,_,Seller);
    .send(Seller, achieve, order(beer,5));
    !at(robot,fridge).</pre>
```



- Además se filtran las entregas que son por parte del agente supermarket. Como ahora ya no se llama así, habrá que eliminar ese filtro
 - Modificamos el plan @a1

```
@a1
+delivered(beer,Qtd,OrderId) [source(supermarket)]
   : true
   <- +available(beer,fridge);
   !has(owner,beer).</pre>
```









Introducción a AgentSpeak









Introducción a AgentSpeak

22

Ejercicio E4

- Otra solución consiste en calcular el precio barato a través de una regla
 - Es más complicada pues trabaja con arrays en Jason







