Prácticas AIN

pyGOMAS

Práctica 2: Comunicación y Coordinación
Servicios + Contract-Net

Índice

- Registro de Servicios
- Comunicación y Coordinación
  - Contract-Net

Recordatorio

- Hay definidos tres tipos de roles en los **agentes externos**:
  - ***Soldier***: soldado de tipo general
    - ALLIED: va a por la bandera y vuelve a la base
    - AXIS: patrulla alrededor de la bandera
  - ***Medic***: acude a curar
  - ***FieldOps***: acude a dar munición
- Un agente asume un único rol durante toda la partida
- Cada rol tiene unas características y ofrece unos determinados **servicios**

Registro de servicios (I)

- Un rol debe registrar un **servicio** para que el resto de roles puedan solicitarlo:

  .register_service("servicio_a")

  Envia un mensaje al agente de servicio para registrar un servicio denominado "servicio_a" que estará disponible para su equipo.

- Ej:

  .register_service("general");

  registra el servicio "general" para su equipo.

Registro de servicios (II)

Registros que se hacen por defecto en todos los agentes:

- ALLIED

  .register_service("allied");

  Soldado: *.register_service("backup");*

  Médico: *.register_service("medic");*

  Fieldops: *.register_service("fieldops");*

- AXIS

  .register_service("axis");

  Soldado: *.register_service("backup");*

  Médico: *.register_service("medic");*

  Fieldops: *.register_service("fieldops");*

Registro de servicios (III)

- ¿Cómo saber que servicios hay disponibles desde un agente?

.get_medics: Solicita al agente de servicios la lista de los médicos de su equipo.
Con la respuesta se crea una creencia: *myMedics(Medics_list)*

.get_fieldops: Solicita al agente de servicios las lista de los operadores de campo de su equipo. Con la respuesta se crea una creencia: *myFieldops(Fieldops_list)*

Registro de servicios (IV)

- ¿Cómo saber que servicios hay disponibles desde un agente?

.get_backups: Solicita al Agente de Servicios los soldados de su equipo.

.get_service("servicio_a"): Solicita al Agente de Servicios otro servicio (distinto de los tres anteriores) a los agentes tropa de su equipo que lo ofrezcan.

La respuesta llega en forma de nueva creencia *servicio_a(L)*

```
+servicio_a(L)
<-
.print("Los agentes de mi equipo con el servicio_a son: ", L).
```

Registro de servicios (V)

- NOTA: Todos las acciones .get siempre excluyen al propio agente que hace la solicitud de la lista que devuelven.

- Ejemplo de uso

  Desde un plan se ejecuta: *.get_medics;*

  Existe otro plan de la forma:

```
+myMedics(M)
  <-  .println("Mis médicos disponibles son: ", M);
      .length(M, X);
      if (X==0){  .println("No tengo médicos");  }.
```

  Nota: si el agente que ejecuta este código fuese médico no aparecería en la lista M

Registro de servicios (VI)

- Ejemplo de uso de un servicio nuevo

Un agente A ejecuta: *.register_service("coronel");*

Otro agente B ejecuta: *.get_service("coronel");*

B además dispone del siguiente plan:

```
+coronel(A)
    <-
        .print("Mi coronel es:", A);
        -coronel(_).
```

Registro de servicios (VII)

- Ejemplo de uso de un servicio nuevo

  Si el coronel ha muerto la lista estará vacía.

  Una alternativa es que B ejecute lo siguiente:

```
.get_service("coronel");
.wait(2000);  // un tiempo prudencial
if (coronel(A)){  .print("Mi coronel es:", A);  -coronel(_); } .
```

Coordinación (I)

- pyGOMAS dispone de mecanismos que permiten la coordinación entre agentes:
  - Sin comunicación (implícita):
    - Sensorización del entorno:
      - Ej.: Seguir al primer compañero que se vea
  - Con comunicación (explicita):
    - Mediante paso de mensajes

Coordinación (II)

- Con comunicación
  - Envío de mensajes mediante la acción interna

    .send(Rec, Perf, Cont)

Donde:
Rec → receptor del mensaje (puede ser una lista)
Perf → performativa (tell, untell, achieve, …)
Cont → contenido

Coordinación (III)

Ej: A1 quiere enviar un mensaje a los soldados médicos de su equipo diciendo que vayan a su posición (para ayudar, para coordinarse, para reagruparse, …)

```
…
?position(Pos);
  ?myMedics(M);    // se supone que antes he ejecutado .get_medics
.send(M, tell, ir_a(Pos));
```

Coordinación (IV)

Ej: los médicos del equipo deberían disponer de un plan de la forma:

```
+ir_a(Pos)[source(A)]
 <-
   .println("Recibido un mensaje de tipo ir_a de ", A, "para ir a: ", Pos).
```

Coordinación (V)

Ej: Si queremos que los soldados hagan algo más sofisticado

```
+ir_a(Pos)[source(A)]
<-
   .println("Recibido mensaje ir_a de: ", A, " para ir a: ", Pos);
   +ayudando;
      .goto(Pos).
```

Mejoras:

- Comprobar si A tiene autoridad sobre el soldado
- Hacer caso sólo si tengo salud, armamento o las dos cosas
- Revisar antes otras tareas pendientes

Coordinación (VI)

- Estrategias vistas (o por ver) en clase:
  - Organización jerárquica: El jefe manda !!!
  - Contract Net: Delegación de tareas
  - Social Choice: Votamos !!!
  - Subastas: quien me ofrece algo mejor !!!
  - …

Protocolo de red de contratos: Contract Net

Contract Net Protocol CNP (Smith,1988).

Desarrollado para la resolución distribuida de problemas

Sigue usándose extensivamente en los SMAs

Sirve para que un agente contrate tareas a otros agentes

Suposiciones:

1. la negociación es un proceso local que no implica control centralizado,
2. existe un medio bidireccional para intercambiar información,
3. cada parte en la negociación evalúa la información desde su propia perspectiva
4. el acuerdo final se alcanza mediante selección mútua.

Contract Net

Un ejemplo …

Un soldado solicita paquetes de medicina

- Pide ayuda a todos los médicos
- Cada médico se propone o no a ayudarlo
- El soldado elige el médico más adecuado (por distancia, por vida, …)

Ejemplo Pedir ayuda médica

Soldado

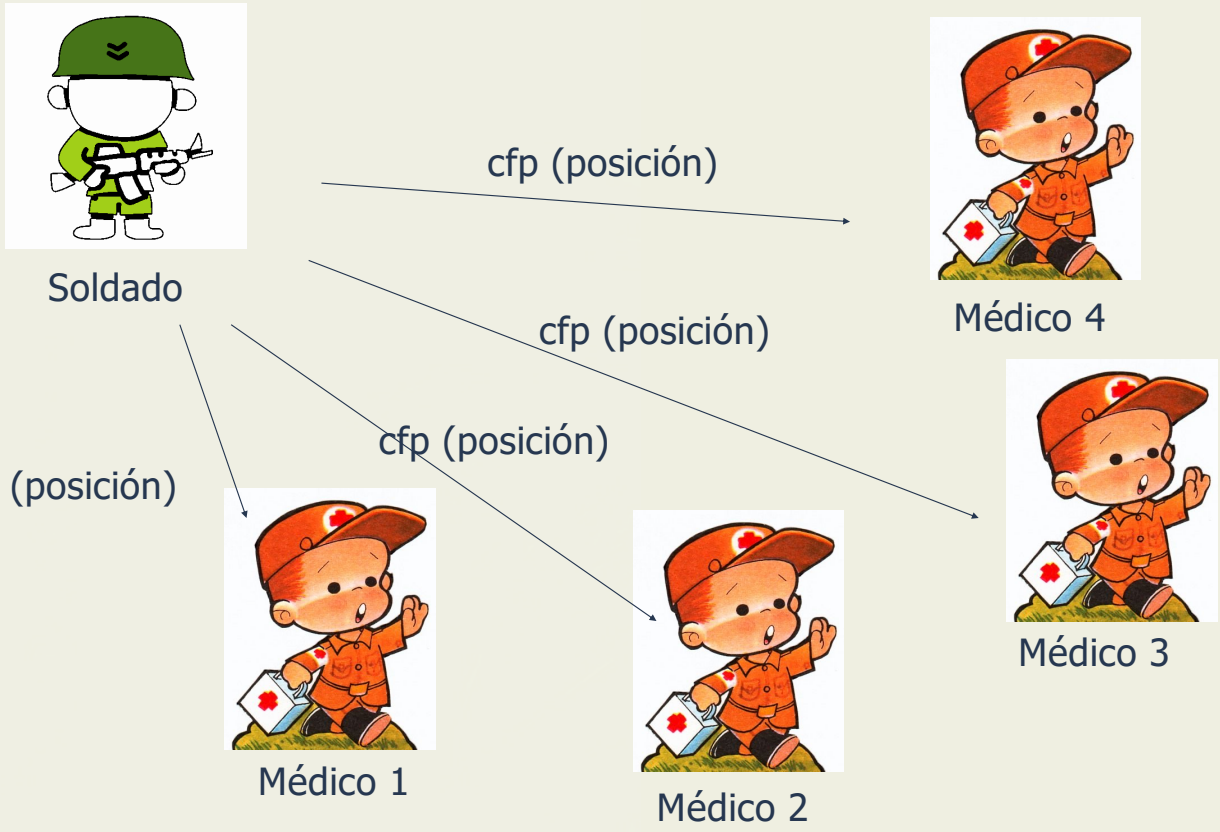
Médico 4

Médico 3

Médico 1

Médico 2

Ejemplo Pedir ayuda médica



Ejemplo Pedir ayuda médica

Soldado

?

Médico 4

?

Médico 3

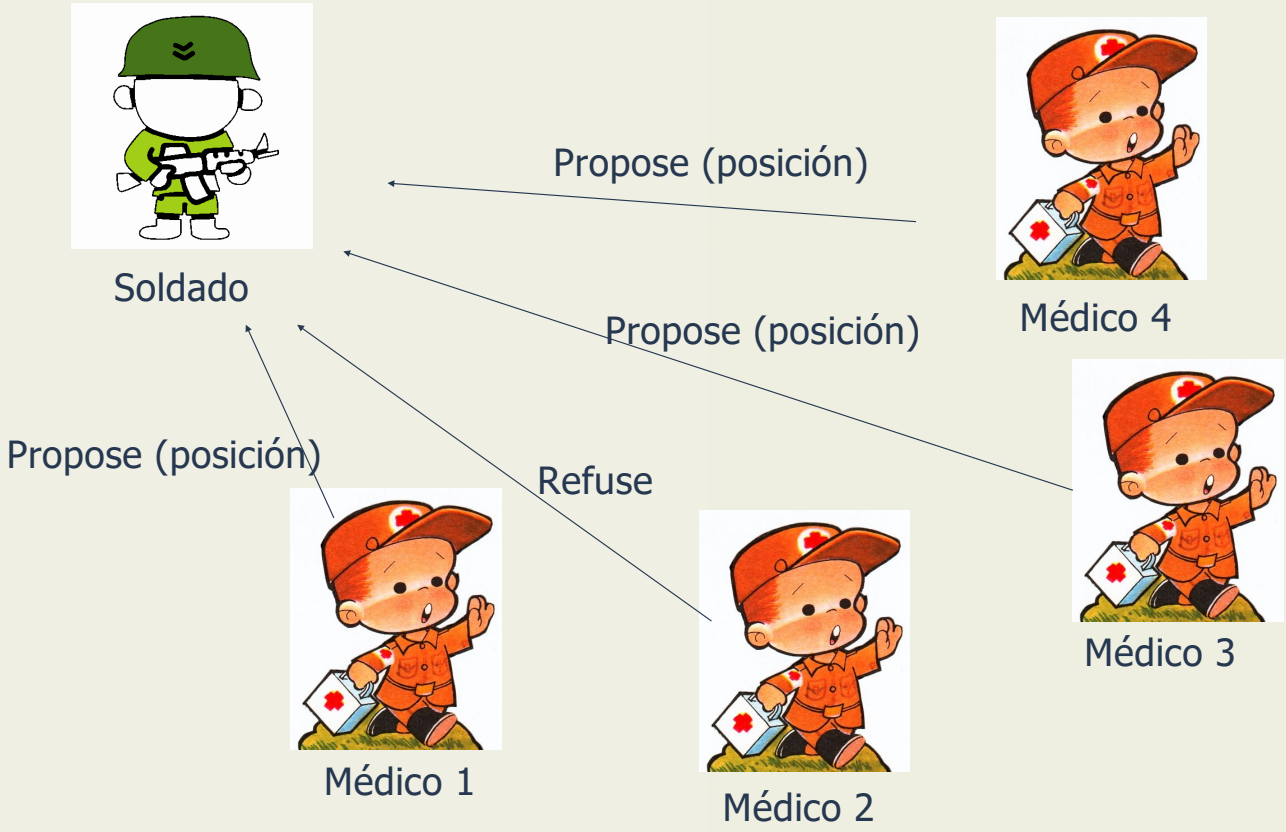
?

Médico 1

?

Médico 2

Ejemplo Pedir ayuda médica



Ejemplo Pedir ayuda médica

Soldado

?

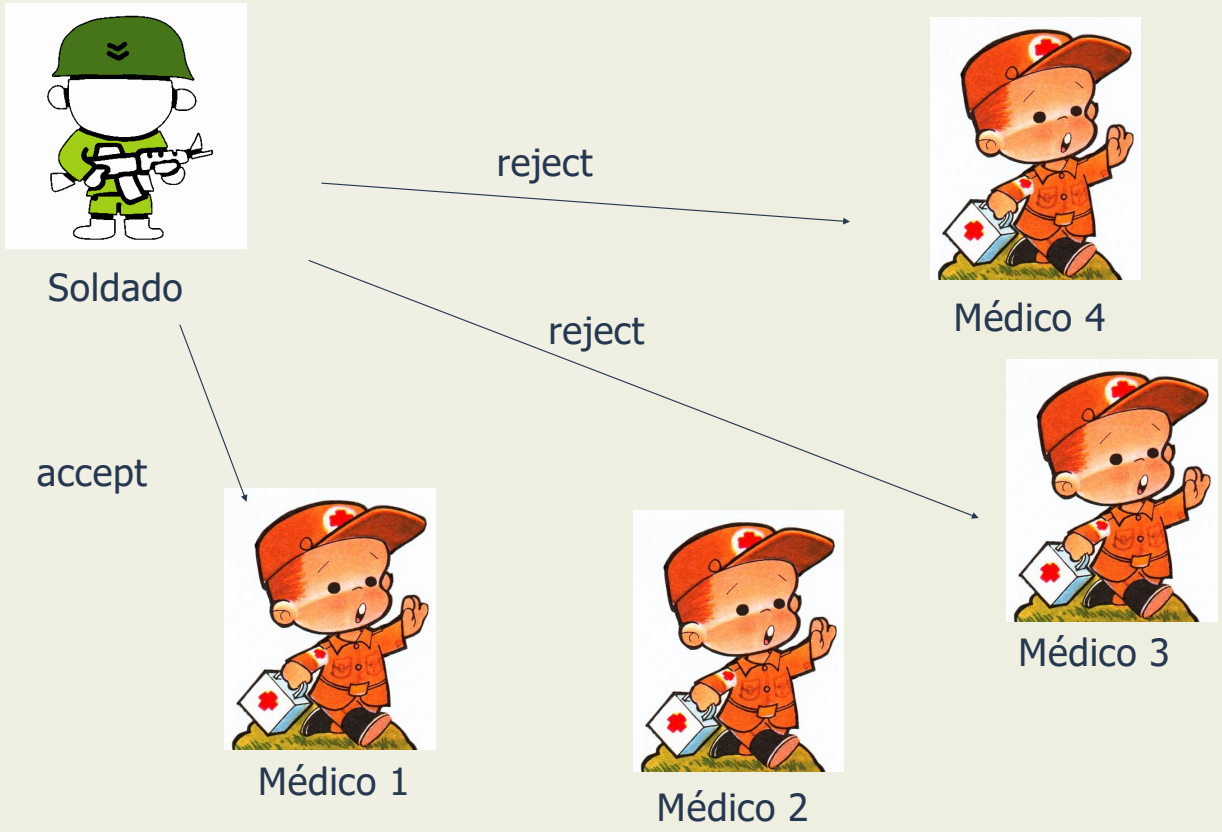
Médico 4

Médico 3

Médico 1

Médico 2

Ejemplo Pedir ayuda médica



Ejemplo **implementado**

Médico 1

Médico 2

Médico 3

Médico 4

1º El soldado **pedirá** ayuda cuando llegue a una determinada posición

2º Todos los médicos le **harán propuestas** de ayuda

3º El soldado **elegirá** a uno de ellos (en este ejemplo al primero)

4º Todos los médicos **recibirán** una respuesta (positiva o negativa)

5º El médico elegido **acudirá** a dar ayuda

Soldado

Implementación del ejemplo

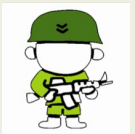
Veamos el inicio del protocolo desde el iniciador (soldado)

1º El soldado **pedirá** ayuda cuando llegue a una determinada posición

```
+flag (F): team(100)
 <-
   .goto([100, 0, 100]);
   +initpos.
```

```
+target_reached(T): team(100) & not (pedidaayuda)
 <-
 .print("He llegado al punto donde solicito ayuda");
 -initpos;
 +pedidaayuda;
  .get_medics.
```

```
+myMedics(M): pedidaayuda
<-
 .print("Pido ayuda");
  ?position(Pos);
   +bids([]);
   +agents([]);
   .send(M, tell, savemeproposal(Pos));
   .wait(1000);
   !!elegirmejor;
  -myMedics(_) .
```



Soldado

Implementación del ejemplo

Veamos el inicio del protocolo desde el médico

2º Todos los **médicos** le **harán propuestas** de ayuda

```
+savemeproposal(Pos)[source(A)]: not (ayudando(_,_))
<-
 ?position(MiPos);
 .send(A, tell, mybid(MiPos));
 +ayudando(A, Pos);
 -savemeproposal(_);
 .print("enviada propuesta de ayuda") .
```

Médico

Implementación del ejemplo

Veamos el inicio del protocolo desde el médico

2º Todos los médicos le **harán propuestas** de ayuda

Y el **soldado** las va recibiendo y almacenando

```
+mybid(Pos)[source(A)]:  pedidaayuda
<-
    .print("Recibo propuesta");
    ?bids(B);
    .concat(B, [Pos], B1);  -+bids(B1);
    ?agents(Ag);
    .concat(Ag, [A], Ag1);  -+agents(Ag1);
    -mybid(Pos).
```

Soldado

Implementación del ejemplo

Veamos el inicio del protocolo desde el médico

3º El soldado **elegirá** a uno de ellos (en este ejemplo al primero)

```
+!elegirmejor: bids(Bi) & agents(Ag)
<-
   .print("Selecciono el mejor: ", Bi, Ag);
  .nth(0, Bi, Pos);  // no elijo el mejor, me quedo con el primero
  .nth(0, Ag, A);
   .send(A, tell, acceptproposal);
  .delete(0, Ag, Ag1);
  .send(Ag1, tell, cancelproposal);
  -+bids([]);
  -+agents([]).
```

Soldado

```
+!elegirmejor: not (bids(Bi))
<-
  .print("Nadie me puede ayudar");
  -pedidaayuda.
```

Implementación del ejemplo

Veamos el inicio del protocolo desde el médico

4º Todos los médicos **recibirán** una respuesta (positiva o negativa)

```
+acceptproposal[source(A)]: ayudando(A, Pos)
<-
  .print("Me voy a ayudar al agente: ", A, "a la posicion: ", Pos);
  .goto(Pos).
```

```
+cancelproposal[source(A)]: ayudando(A, Pos)
<-
  .print("Me cancelan mi proposicion");
  -ayudando(A, Pos).
```

Médico

Implementación del ejemplo

Veamos el inicio del protocolo desde el médico

5º El médico elegido **acudirá** a dar ayuda

```
+target_reached(T): ayudando(A, T)
 <-
   .print("MEDPACK! para el agente:", A);
   .cure;
   ?miposicion(P);
   .goto(P);
   -ayudando(A, Pos).
```

Médico

Implementación del ejemplo

Veamos el ejemplo en ejecución

Dudas

Enviad vuestras dudas por:

- Teams (en horario de clase)
- Correo: vjulian@upv.es
  carrasco@dsic.upv.es
- o por Poliformat