AIN

Ejercicios

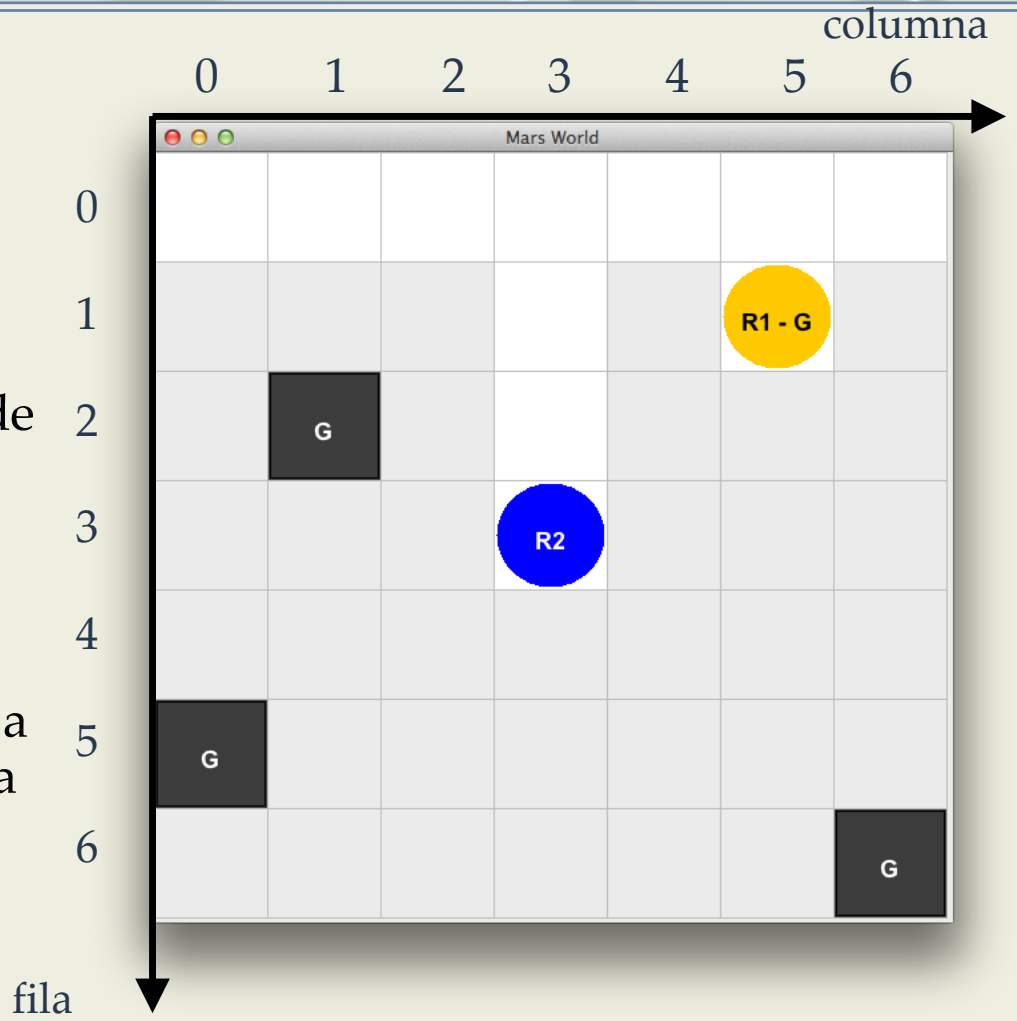
Jason
by Gustave Moreau (1865)
Oil on canvas, 204 x 115.5 cm.
Musée d'Orsay, Paris.
© Photo RMN. Photograph by Hervé Lewandowski.

Jason

a Java-based interpreter for an extended version of AgentSpeak

Ejemplo: Cleaning Robots

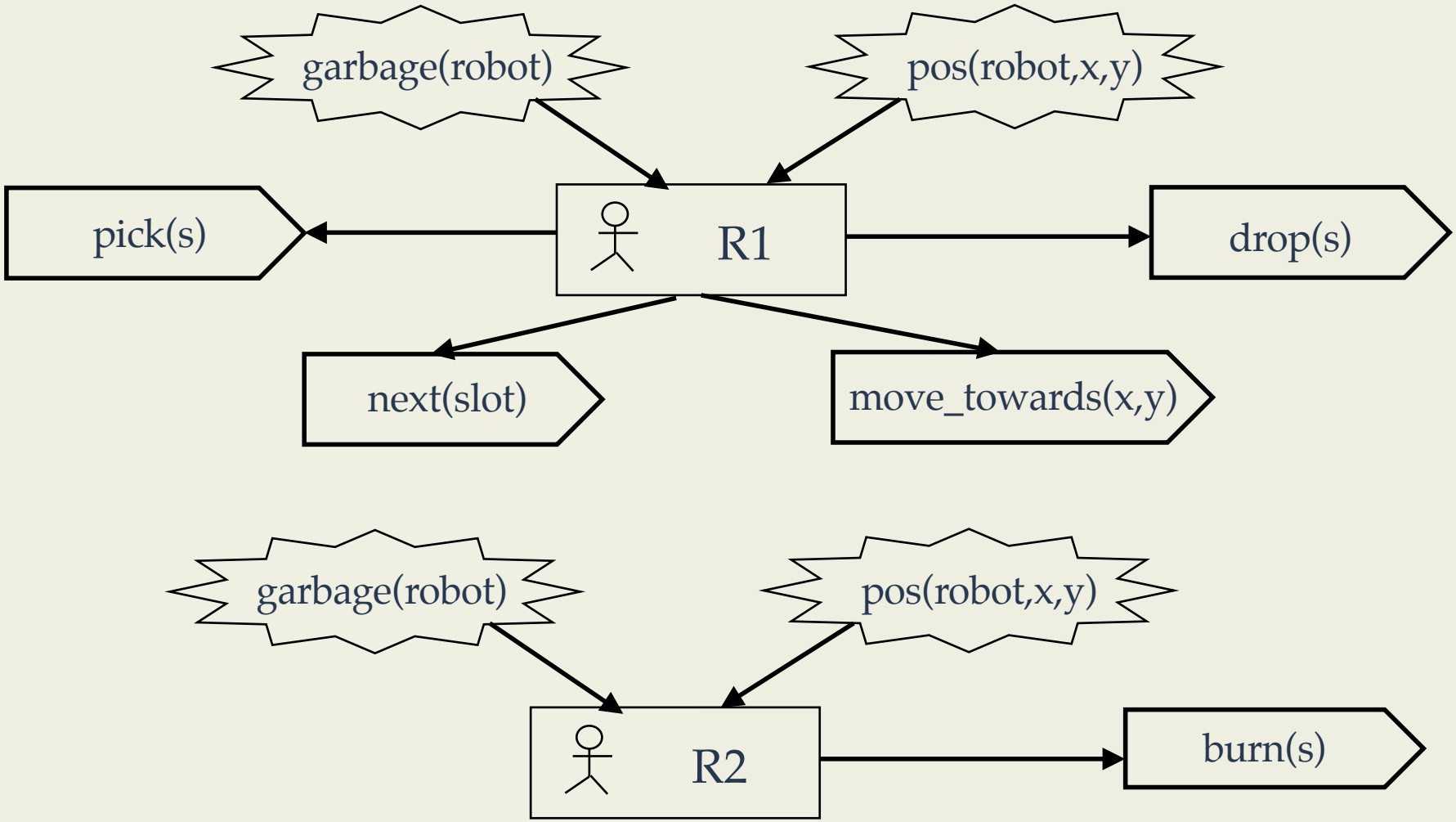
- Tenemos un robot (R1) que busca trozos de basura en toda la cuadrícula, y cuando se encuentra uno, lo lleva a otro robot (R2), ubicado en el centro de la cuadrícula, donde hay un incinerador; el robot en movimiento regresa al lugar donde se encontró el último trozo de basura y continúa la búsqueda desde allí. Se basa en el escenario original que Anand Rao utilizó cuando introdujo el lenguaje
- El robot R1se mueve de izquierda a derecha por la fila en que se encuentra, cuando llega a la última columna de la fila pasa a la primera columna de la siguiente fila (si está en la última fila la siguiente fila es la primera fila.



Ejemplo: Cleaning Robots

- **Percepciones:**
  - garbage(robot)
  - pos(robot,x,y)
- **Acciones:**
  - **pick(s):** el robot coge la basura que hay en la celda donde se encuentra.
  - **drop(s):** el robot deposita la basura que transporta en la celda en que se encuentra.
  - **next(celda):** el robot se desplaza a la siguiente celda (teniendo en cuenta su modo de desplazarse)
  - **move_towards(x,y):** el agente se dirige hacía la posición (x,y) por el camino más directo (en diagonal entre su posición y (x,y)).
  - **burn(s):** el robot R2 quema la basura que tiene depositada en su celda.

Ejemplo: Robots de limpieza



Ejemplo: Robots de limpieza

- Creencias

  pos(r1, X, Y)
  garbage(r1)
- Reglas

  at(P) :- pos(P,X,Y) & pos(r1,X,Y)
- Objetivos

  !check(slots)
  !carry_to(R)
  !take(S,L)
  !ensure_pick(S)
- Acciones internas:

  .desire: this internal action takes a literal as parameter and succeeds if that literal is one of the agent's desires.

Ejemplo:Robots de Limpieza (R1)

- Planes (I)

```
+!check(slots) : not garbage(r1)
   <- next(slot);
      !check(slots).

+!check(slots).

@lg[atomic]
+garbage(r1) : not
.desire(carry_to(r2))
  <- !carry_to(r2).
```

```
+!carry_to(R)
   <- // remember where to go back
      ?pos(r1,X,Y);
      -+pos(last,X,Y)
      // carry garbage to r2
      !take(garb,R);
      // goes back and continue to
check
      !at(last);
      !check(slots).
```

Ejemplo:Robots de Limpieza (R1)

- Planes (I)

```
+!take(S,L) : true
   <- !ensure_pick(S);
      !at(L);
      drop(S).

+!ensure_pick(S) : garbage(r1)
   <- pick(garb);
      !ensure_pick(S).

+!ensure_pick(S).

+!at(L) : at(L).

+!at(L) <- ?pos(L,X,Y);
           move_towards(X,Y);
           !at(L).
```

Ejemplo: Robots de Limpieza (R2)

- Planes

  +garbage(r2) : true <- burn(garb).

Ejemplo: Robots de Limpieza

Robots de Limpieza : ¿Cómo se ejecuta?

- Fichero de definición del MAS (extensión .mas2j)

```
MAS mars {

    infrastructure: Centralised

    environment: MarsEnv

    agents: r1; r2;

}
```

Ejercicios

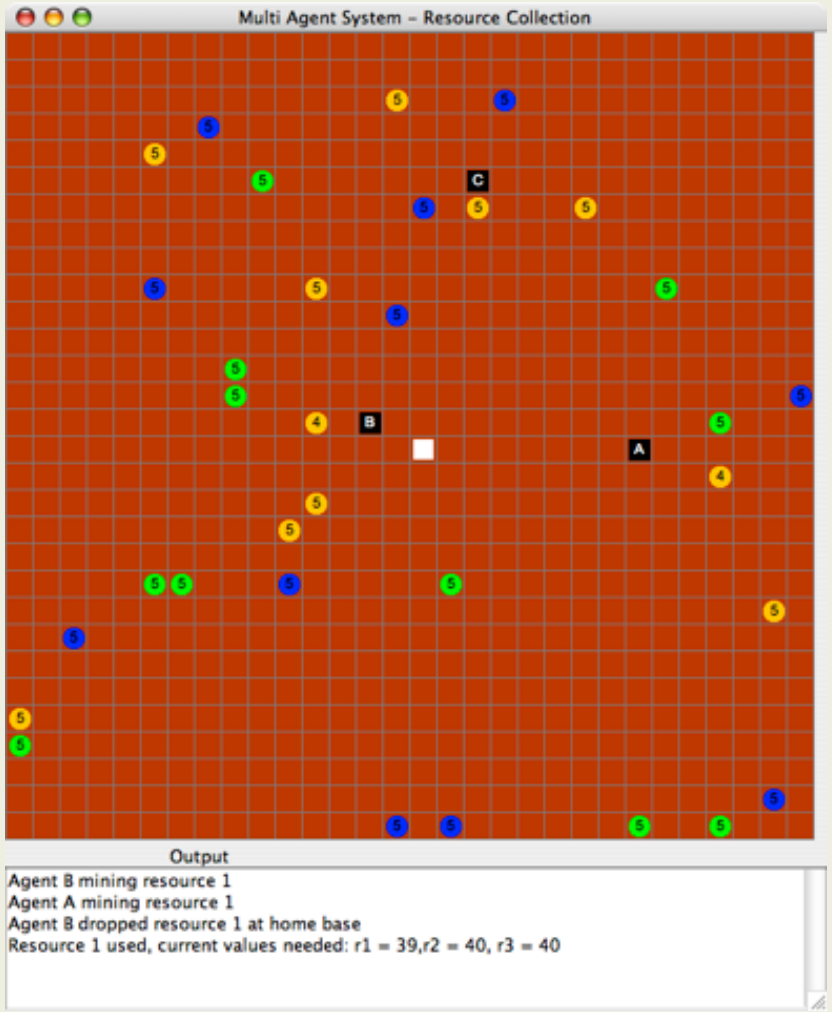
- Modificar el ejemplo de recogida para incluir comunicación:

  Añadir COORDINACIÓN → necesidad de mensajes

  - 1º Añadir un agente "Jefe" que le diga al robot r1 cuando empezar y de forma periódica le diga que pare o que siga.
    - Emplear las acciones internas: .send, .wait, .at, .println,
    - Descripción en: http://jason.sourceforge.net/api/jason/stdlib/package-summary.html
  - 2º Añadir un nuevo comportamiento en el robot r1 y el agente jefe de forma que:
    - El robot R1 avise al robot jefe de que ha terminado de recorrer la matriz.
    - El robot jefe recibe el mensaje y le ordena al robot r1 que se vuelva al punto de origen.
    - Tras un tiempo de espera el jefe ordena que se mueran los agentes r1 y r2
  - 3º Incluir dos agentes que quemen la basura (estilo r2)
  - 4º Cambiar r1 para llevar la basura al robot quemador más cercano

Ejemplo: Mining Robots
Tipos de agente

- Agente “builder":
  - Situado en el centro de la tabla (que representa algún entorno espacial).
  - Está realizando algunos trabajos de construcción para los que necesita 3 tipos diferentes de recursos.
- Agente “Miner”:
  - Cuadrado negro.
  - Percibe un recurso en una celda sólo cuando está sobre esa celda. También percibe el tipo de recurso.
  - Si el recurso es del tipo que está buscando, coge una unidad, se la lleva al BUILDER y vuelve a donde encontró la mina.
- Cada mina tiene 5 unidades de un tipo de recurso (tipo 1 –naranja-, tipo 2 –verde-, tipo 3 –azul-).



Ejemplo: Mining Robots (tipos de agente)

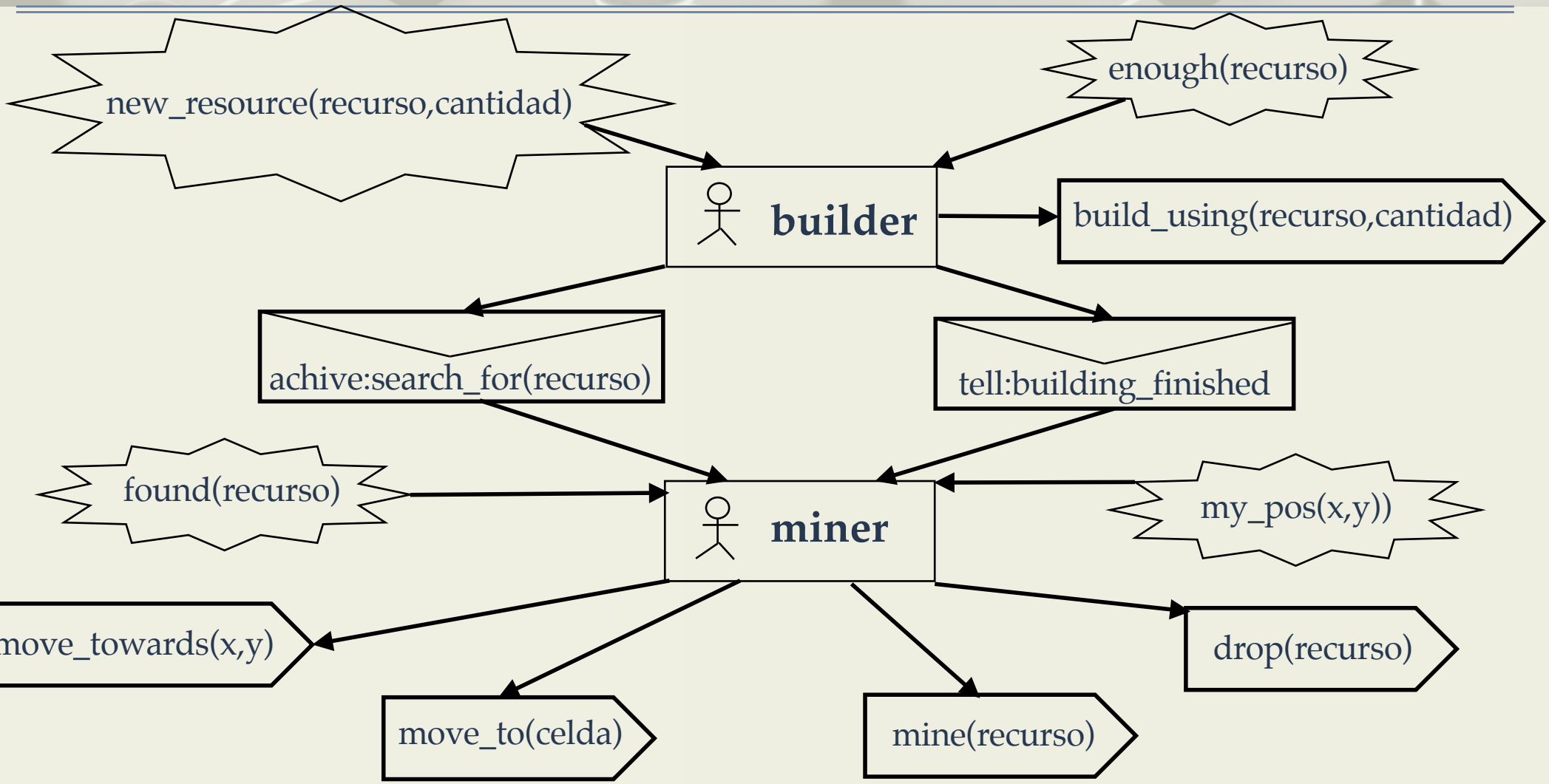
- Agente 'miner':
  - Celdas negras.
  - A: se mueve de izquierda a derecha por la fila en que se encuentra, cuando llega a la última columna de la fila pasa a la primera columna de la siguiente fila (si está en la última fila la siguiente fila es la primera fila.
  - B: se mueve de derecha a izquierda por la fila en que se encuentra, cuando llega a la primera columna de la fila pasa a la última columna de la fila anterior (si está en la primera fila la fila anterior es la última fila).
  - C: se mueve de arriba a bajo por la columna en que se encuentra, cuando llega a la última fila de la columna pasa a la primera la fila de la columna siguiente(si está en la última columna la columna siguiente es la primera columna).

- Agente ‘miner’.
  - **Percepciones:**
    - **my_pos(x,y):** los agentes ‘miner’ perciben la posición en que se encuentran (columna,fila).
    - **found(recurso):** los agentes ‘miner’ perciben el tipo de recurso que se encuentra en su posición.
  - **Acciones:**
    - **move_to(celda):** se mueve a la siguiente celda (la siguiente celda dependerá si es un agente de tipo A, B o C).
    - **mine(recurso):** el agente coge el recurso de la mina donde se encuentra situado y disminuye la cantidad de recursos de esa mina.
    - **drop(recurso):** el agente deposita el recurso que transporta en la posición donde se encuentra.
    - **move_towards(x,y):** el agente se dirige hacía la posición (x,y) por el camino más directo (en diagonal entre su posición y (x,y)).

Ejemplo: Mining Robots (tipos de agente)

- BUILDER agent:
  - Inicialmente necesita recursos de tipo 1 –naranja-.
  - Después de recibir un nº aleatorio de muestras de tipo 1, pide a todos los mineros (*brodcast*) recursos de tipo 2 –verde- (*achive(search_for(2))*.
  - Por último, después de recibir un nº aleatorio de muestras de tipo 2 pide a todos los mineros (*brodcast*) recursos de tipo 3 –azul- (*achive(search_for(3))* .
  - Construcción Finalizada: informar a todos los mineros (*brodcast*) que van a la posición del BUILDER (*tell(building_finished))*
  - **Percepciones:**
    - **new_resource(recurso,cantidad):** cuando se deposita un recurso en su celda el agente 'builder' percibe el tipo de 'recurso' depositado y la 'cantidad' de recursos de ese tipo que tiene.
    - **enough(recurso):** el agente ´builder´ percibe si ya tiene suficientes recursos del tipo 'recurso'.
  - **Acciones:**
    - **build_using(r,c):** construye utilizando el recurso de tipo 'r' del que dispone de una cantidad 'c'.

Ejemplo: Mining Robots



Ejemplo: Mining Robots

```
MAS resourceCollection {
    infrastructure: Centralised

    environment: env.planetEnv

    agents:
        col #3;
        builder;
}
```

Ejemplo: Mining Robots

Builder Agent

```
// Beliefs
resource_needed(1).

// Plans

// a resource has been dropped at site so build site further using that resource
//@pnr[breakpoint]
+new_resource(R,ID) : resource_needed(R)
  <- build_using(R,ID).

// a resource is not needed any more, inform collectors to search for another resource
+enough(R): true
  <- -resource_needed(R);
     +resource_needed(R+1);
     .broadcast(achieve, search_for(R+1)).

// just tell collectors that I finished the building
+building_finished: true
  <- .broadcast(tell,building_finished).
```

Ejemplo: Mining Robots

Miner Agent

```
pos(boss,15,15).
checking_cells.
resource_needed(1).

// Plans

+my_pos(X,Y)
  : checking_cells & not building_finished
  <- !check_for_resources.

+!check_for_resources
  : resource_needed(R) & found(R)
  <- !stop_checking;
     !take(R,boss);
     !continue_mine.

+!check_for_resources
  : resource_needed(R) & not found(R)
  <- move_to(next_cell).
```

Ejemplo: Mining Robots

Miner Agent

```
+!stop_checking : true
  <- ?my_pos(X,Y);
     +pos(back,X,Y);
     -checking_cells.

+!take(R,B) : true
  <- mine(R);
     !go(B);
     drop(R).

+!continue_mine : true
  <- !go(back);
     -pos(back,X,Y);
     +checking_cells;
     !check_for_resources.
```

Ejemplo: Mining Robots

Miner Agent

```
+!go(Position)
  :  pos(Position,X,Y) & my_pos(X,Y)
  <- true.

+!go(Position) : true
  <- ?pos(Position,X,Y);
     move_towards(X,Y);
     !go(Position).

@psf[atomic]
+!search_for(NewResource) : resource_needed(OldResource)
  <- +resource_needed(NewResource);
     -resource_needed(OldResource).

@pbf[atomic]
+building_finished : true
  <- .drop_all_desires;
     !go(boss).
```

Ejercicios

- Modificar el ejemplo de recogida para incluir comunicación:

  Añadir COORDINACIÓN → necesidad de mensajes

  - 1º Cuando un MINER agent descubre un recurso, le comunica su posición a todos los MINER agents, que se dirigen a esa posición. Cuando no queda más recurso en esa posición, vuelven a la posición donde estaban cuando recibieron esta comunicación, y a realizar su recorrido habitual de búsqueda.

  - 2º Incrementar los mineros hasta 6, y haced que el minero que encuentra el recurso, le comunique su posición al BUILDER. Éste haga un Contract-Net con el resto de mineros, y se quede con los 4 más cercanos, que irán a la posición del recurso, cargarán y lo llevarán al BUILDER, para después volver a su posición original.