UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CURSO DE CIÊNCIAS DA COMPUTAÇÃO

Sistema Multiagente

Matheus Henrique Schaly (18200436)
Pedro Alonso Condessa (19200360)

a) A notificação de um recurso encontrado é realizada através do comando broadcast (imagem 1). Com isso, todos os agentes ficam sabendo (crença) que um recurso foi encontrado (imagem 2). E eles começam a mover-se em direção ao recurso (imagem 3). Ao chegar na posição do recurso, os agentes realizam a mineração dele (imagem 4).

```
10  +!check_for_resources : resource_needed(R) & found(R) & my_pos(X, Y)
11      <- !stop_checking;
12         // Avisa que recurso encontrado, passando local e o recurso, para todos os agentes
13         .println("Hey! I found a new resource at X Y: ", X, " ", Y);
14         .broadcast(tell, resource(X, Y, R));
15         !take(R, boss);
16         !continue_mine.
```

```
36  +resource(X, Y, R) : true
37      // Manda agente para a posição X, Y onde o recurso foi encontrado
38      <- .println("I know that there is a resource at X Y: ", X, " ", Y);
39         .println("The resource type is: ", R);
40         // Manda o agente parar de checar
41         -checking_cells;
42         // Manda o agente ir em direção ao recurso
43         !go(X, Y).
```

```
67  // Vai em direção ao recurso até encontrá-lo
68  +!go(X, Y) : true
69      <- .println("I am moving towards X Y: ", X, " ", Y);
70         .wait(230);
71         // Avisa para parar de checar celulas e ir para onde o recurso foi encontrado
72         move_towards(X, Y);
73         !go(X, Y).
```

```
56  // Acionado quando a posição do agente estiver sobre o recurso
57  +!go(X, Y) : my_pos(X, Y) & resource(X, Y, R)
58     <- .println("FOUND IT!");
59        .wait(230);
60        // Minere o recurso
61        +checking_cells;
62        !stop_checking;
63        !take(R, boss);
64        !continue_mine.
```

b) Para avisar sobre o esgotamento do recurso também utilizamos o comando broadcast, mas usando como argumento o untell (imagem 5).

```
19  +!check_for_resources : resource_needed(R) & not found(R) & my_pos(X, Y)
20     <- .wait(230);
21        move_to(next_cell);
22        // Avisa que recurso não foi mais encontrado, passando local e o recurso, para todos os ag
23        .broadcast(untell, resource(X, Y, R)).
```

c) Para permitir que os agentes guardem informações sobre os recursos localizados que ainda não estão sendo procurados, também utilizamos o comando broadcast. Montamos a lógica fazendo com que o recurso necessário R, não é o mesmo que o recurso encontrado F (imagem 6). Armazenamos este recurso ainda não minerável em future_resource (imagem 7).

```
26  +!check_for_resources : resource_needed(R) & found(F) & my_pos(X, Y)
27     <- .wait(230);
28        // Avisa que resurso encontrado, passando local e o recurso, para todos os agentes
29        // mas nesse caso o recurso necessário não é o mesmo que o recurso encontrado
30        future_resource(X, Y, R);
31        .println("We don't really want resource at: ", X, " ", Y);
32        .broadcast(tell, future_resource(X, Y, F));
33        move_to(next_cell).
```

```
46  +future_resource(X, Y, R) : true
47     // Manda agente para a posição X, Y onde o recurso foi encontrado
48     <- .println("I know that there is resource that we still don't mine at X Y: ", X, " ", Y);
49        .println("The resource type is: ", R).
```