

Aluno: Mateus Neves Barreto
R.A.: 142358
Disciplina: IA006
Professor: Ricardo R. Gudwin

Relatório – Aula 3

1 Atividade 1

A seguir são apresentados os “prints” da criação da criatura com a regra “move-to-food”.

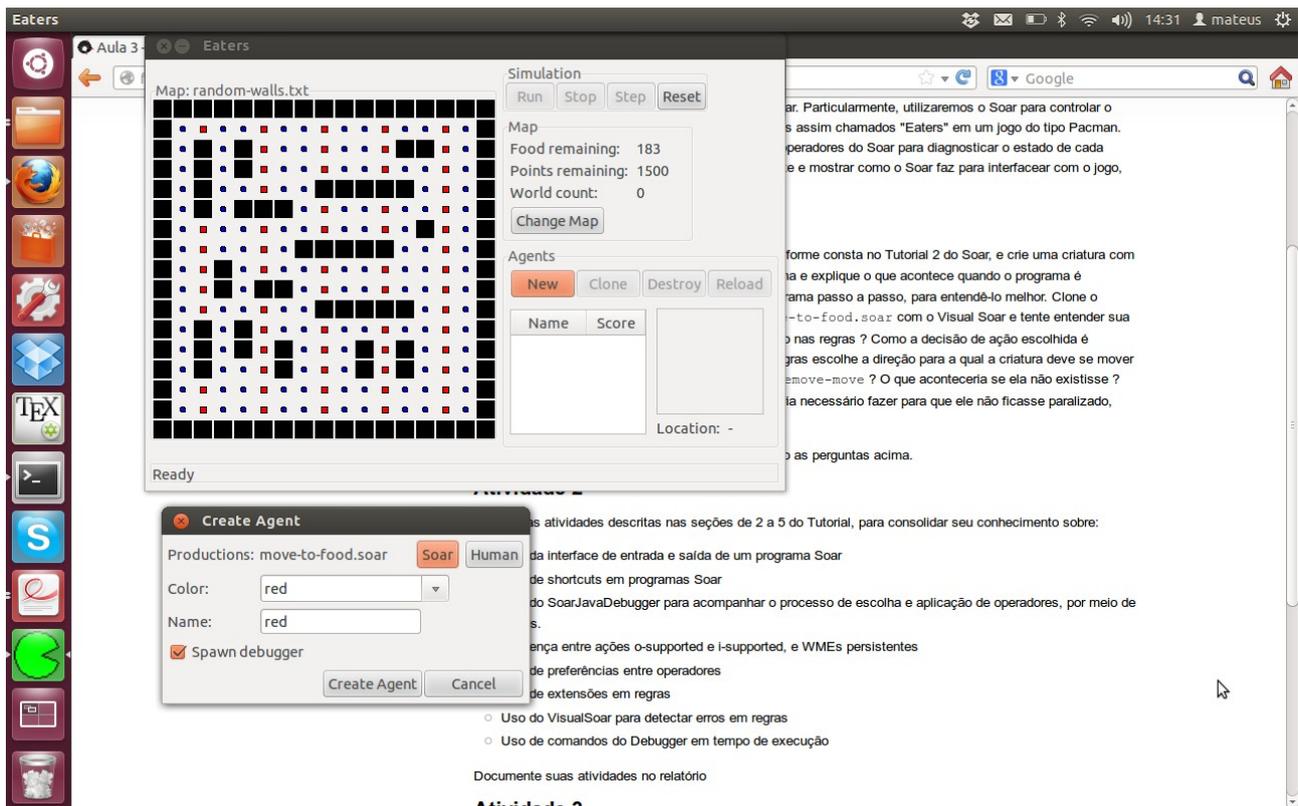


Figura 1 – Criando agente “move-to-food”.

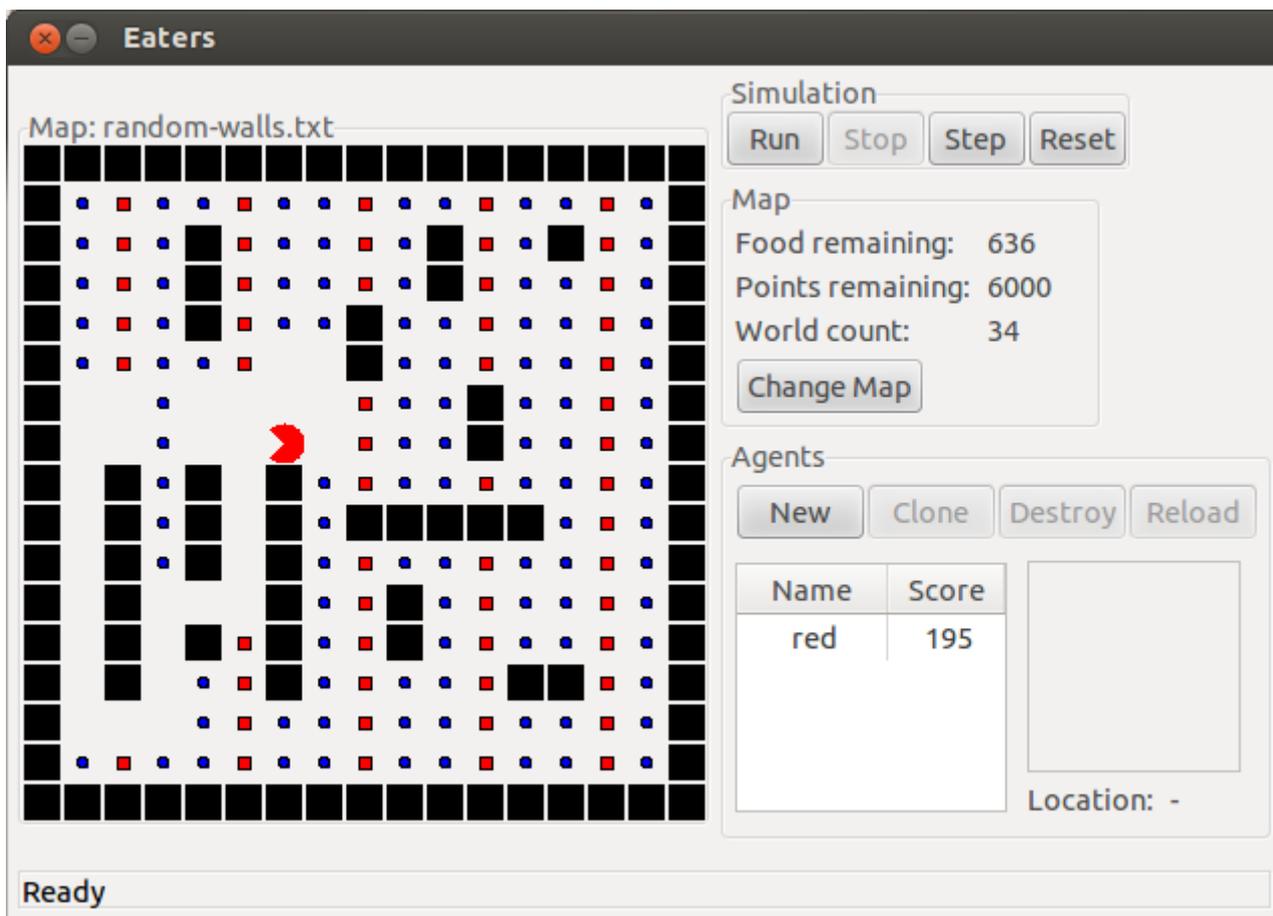


Figura 2 – Executando agente “move-to-food”.

Pode-se observar na Figura 2, que o agente está cercado de espaços sem “comida”. Este estado gerado, impossibilita a escolha de uma regra para as regras propostas no arquivo “move-to-food.soar”, com isso o agente fica estagnado em sua posição não conseguindo mais buscar “comida”.

As regras presentes no arquivo “move-to-food.soar” apenas consideram a criação de propostas se o agente possuir uma “comida comum” ou uma “comida de bônus” em sua vizinhança. A direção que o agente deve tomar caso exista a comida é escolhida aleatoriamente, devido ao operador ser seguido pelo sinal de igualdade “=”. Por exemplo, na existência de apenas uma comida na vizinhança, o agente tomará a direção desta comida, porém na existência de duas ou mais comidas na vizinhança, a direção que o agente tomará será aleatória presente nas direções das comidas vizinhas.

A função da regra “apply*move-to-food*remove-move” é remover o operador da memória de trabalho que já foi “executado”. Aparentemente a remoção desta regra não causa efeito direto na execução do agente, porém é certo que sua memória de trabalho tenderá a crescer de acordo com o tempo de execução, diferente da execução do agente com esta regra presente.

A estratégia para que o agente não fique paralisado após um tempo e a inserção do “estado empty” na regra de proposição, considerando-o com uma preferência menor de escolha. Isso evitaria que o agente estagnasse após um tempo.

2 Atividade 2

2.1 move-to-north

A Figura 3 apresenta o código soar para a criação das regras que o agente se move apenas na direção norte. Todas as proposições e aplicações das ações para os agentes estão sendo realizadas via escrita e leitura na memória de trabalho, pelos comandos de estados de io.input e io.output, entrada e saída respectivamente.

```

go-to-north.soar (~/.Dropbox/Mestrado/IA006/aula3) - gedit
Abrir Salvar Desfazer
go-to-north.soar ✕
sp {propose*move-north
(state <s> ^io.input-link.eater <e>)
(<e> ^x <x> ^y <y>)
-->
(<s> ^operator <o> +)
(<o> ^name move-north)}
sp {apply*move-north
(state <s> ^operator <o>
^io <io>)
(<io> ^output-link <out>)
(<o> ^name move-north)
-->
(<out> ^move <move>)
(<move> ^direction north)}

Texto sem formatação Largura das tabulações: 8 Lin 6, Col 24 INS

```

Figura 3 – Agente para o norte.

A Figura 4 apresenta a análise da execução do código apresentado na Figura 3 até a execução da regra remove-move.

The screenshot shows the Soar Debugger interface. The central log window displays the following execution details:

```

watch 3 --timetags
step
source (/home/mateus/Dropbox/Mestrado/IA006/aula3/go-to-north.soar)***
Total: 3 productions sourced.
--- input phase ---
--- propose phase ---
Firing propose*move-north
11 12 14 102 101
--- decision phase ---
1: 0: 01 (move-north)
step
--- apply phase ---
--- Firing Productions (PE) For State At Depth 1 ---
Firing apply*move-north
152 130 11 13
--- Change Working Memory (PE) ---
--- output phase ---
--- input phase ---
--- propose phase ---
Firing propose*move-north
11 12 14 137 101
Retracting propose*move-north
11 12 14 102 101
--- decision phase ---
2: 0: 02 (move-north)
step
--- apply phase ---
--- Firing Productions (PE) For State At Depth 1 ---
Firing apply*move-north*remove-move
167 165 11 13 133 135
--- Change Working Memory (PE) ---
--- Firing Productions (IE) For State At Depth 1 ---
--- Change Working Memory (IE) ---
--- output phase ---
--- input phase ---
--- propose phase ---
Firing propose*move-north
11 12 14 172 101
Retracting propose*move-north
11 12 14 137 101
--- decision phase ---
3: 0: 03 (move-north)

```

The right window shows the state and a production rule diagram. The state is:

```

(S1 ^epmem E1 ^io I1 ^operator O3 ^operator O3 + ^reward-link R1 ^smem S2
^superstate nil ^type state)

```

The production rule diagram shows a sequence of nodes: I (input) -> P (propose) -> D (decision) -> A (apply) -> O (output). The current state is labeled as WORLD+3.

Figura 4 – Regra remove-move.

As Figuras 5 e 6 apresentam a utilização do break pointer no canto inferior direito.

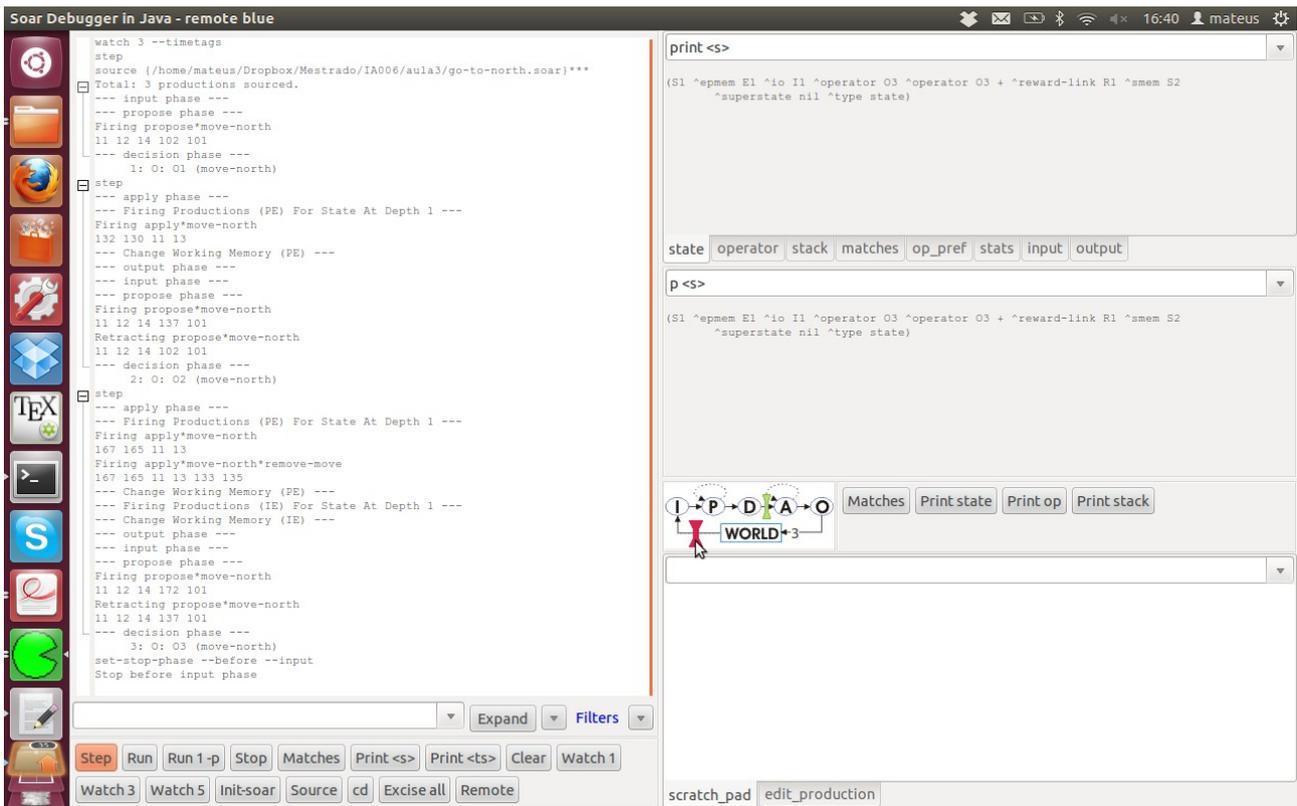


Figura 5 – Utilizando o break pointer - 1.

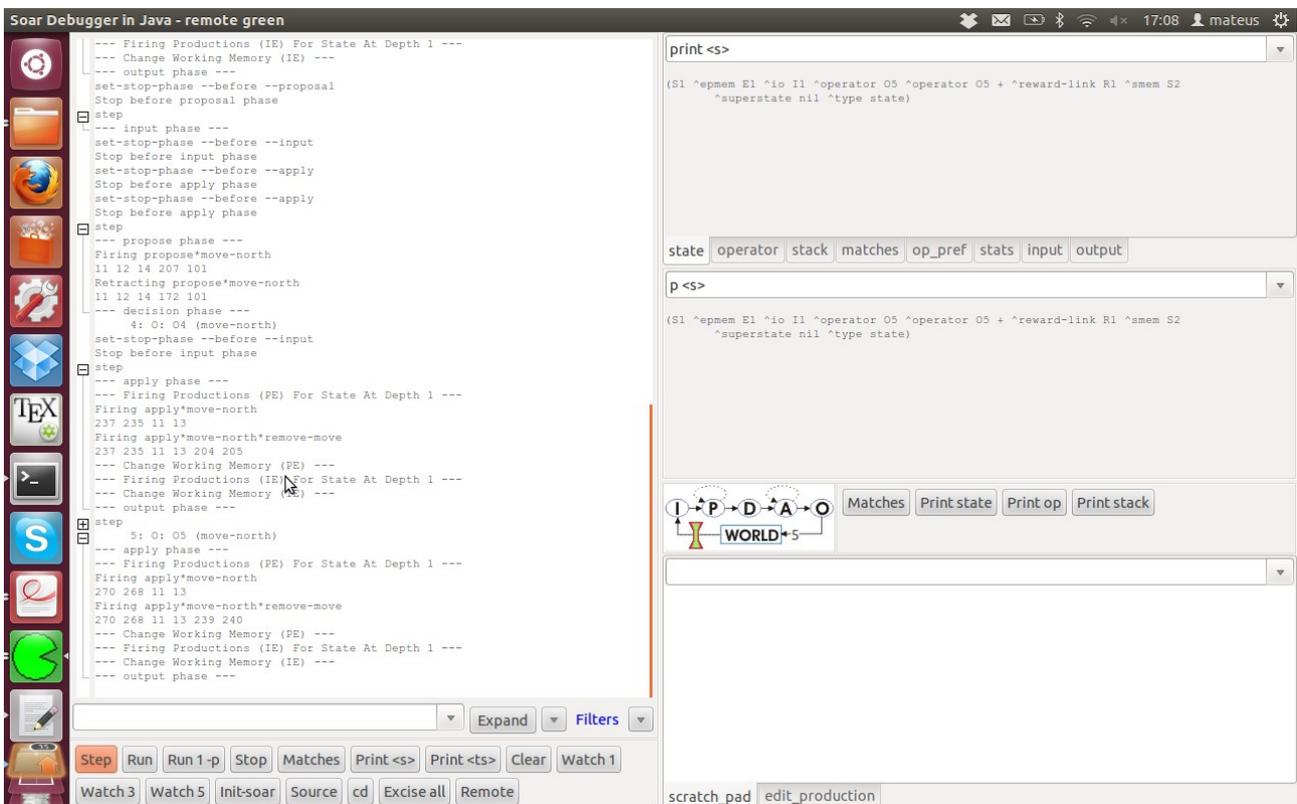
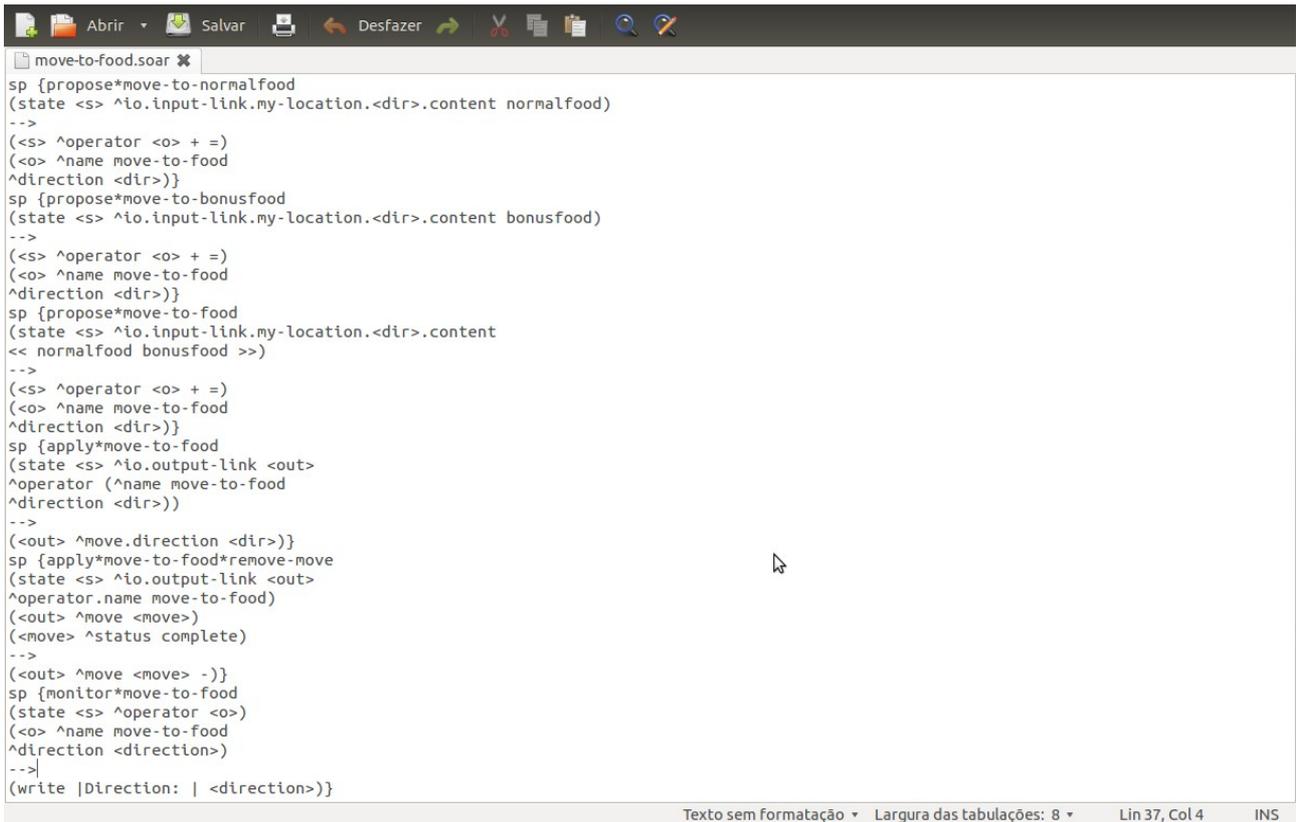


Figura 6 – Utilizando o break pointer - 2.

2.2 move-to-food

Neste sub-tópico serão apresentadas as tarefas realizadas para as ações nos agentes para os mesmos se moverem em direção as comidas, comuns e de bônus. A Figura 7 apresenta o código completo

contendos as regras para realizar tal ação.



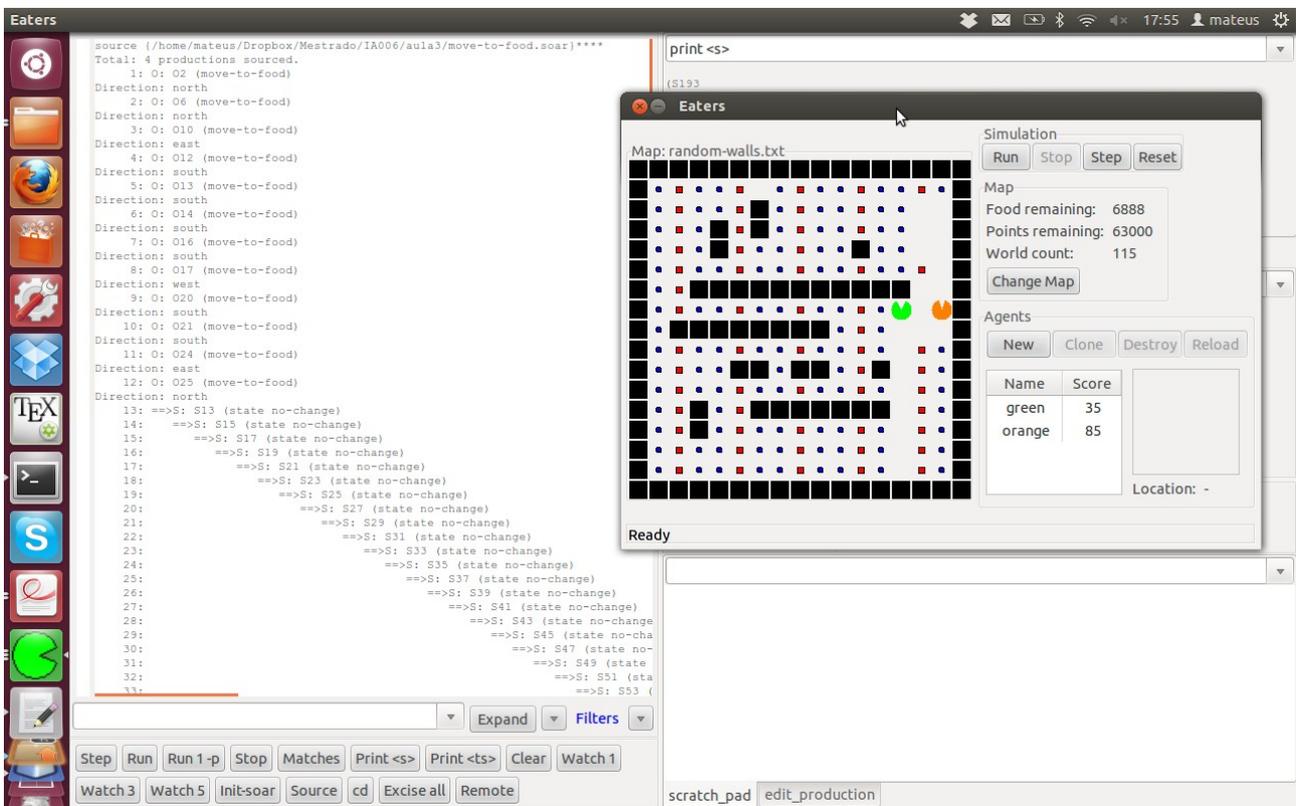
```
move-to-food.soar x
sp {propose*move-to-normalfood
(state <s> ^io.input-link.my-location.<dir>.content normalfood)
-->
(<s> ^operator <o> + =)
(<o> ^name move-to-food
^direction <dir>)}
sp {propose*move-to-bonusfood
(state <s> ^io.input-link.my-location.<dir>.content bonusfood)
-->
(<s> ^operator <o> + =)
(<o> ^name move-to-food
^direction <dir>)}
sp {propose*move-to-food
(state <s> ^io.input-link.my-location.<dir>.content
<< normalfood bonusfood >>)}
-->
(<s> ^operator <o> + =)
(<o> ^name move-to-food
^direction <dir>)}
sp {apply*move-to-food
(state <s> ^io.output-link <out>
^operator (^name move-to-food
^direction <dir>))
-->
(<out> ^move.direction <dir>)}
sp {apply*move-to-food*remove-move
(state <s> ^io.output-link <out>
^operator.name move-to-food)
(<out> ^move <move>)
(<move> ^status complete)
-->
(<out> ^move <move> -)}
sp {monitor*move-to-food
(state <s> ^operator <o>)}
(<o> ^name move-to-food
^direction <direction>)
-->
(write |Direction: | <direction>)}

```

Texto sem formatação ▾ Largura das tabulações: 8 ▾ Lin 37, Col 4 INS

Figura 7 – Código move-to-food.

A Figura 8 apresenta a execução do agente com o código move-to-food.



The screenshot shows the Eaters simulation environment. On the left is a vertical toolbar with icons for various functions. The main window is divided into several sections:

- Terminal:** Displays the execution log of the SOAR agent, showing production rules being fired and their states. It starts with "Total: 4 productions sourced." and lists 13 production rules with their respective directions (north, south, east, west) and states (state no-change).
- Simulation Panel:** Contains a "Map: random-walls.txt" showing a grid world with black walls, blue and red food pellets, and two agents: a green one and an orange one. It includes "Run", "Stop", "Step", and "Reset" buttons.
- Agents Table:** Lists the current agents in the simulation:

Name	Score
green	35
orange	85
- Buttons:** At the bottom, there are buttons for "Step", "Run", "Run 1-p", "Stop", "Matches", "Print <s>", "Print <ts>", "Clear", "Watch 1", "Watch 3", "Watch 5", "Init-soar", "Source", "cd", "Excise all", and "Remote".

Figura 8 – Execução move-to-food.

Pode-se observar na Figura 9 a impressão do operador escolhido e sua respectiva direção, pelo comando “(write |Direction: | <direction>)”.

The screenshot shows a Prolog IDE interface. On the left, a 'watch -P' window displays a list of steps and their corresponding directions: west, south, east, and south. The steps are numbered 1 through 7. Below the watch window are buttons for 'Step', 'Run', 'Run 1-p', 'Stop', 'Matches', 'Print <s>', 'Print <ts>', 'Clear', and 'Watch 1'. On the right, a 'print -depth 2 i3' window shows the current state: (I3 ^move M7) and (M7 ^direction east ^status complete). Below this, a 'state' window shows the current state: ==>S: S1 and 0: 0: 031 (move-to-food). At the bottom, there is a 'p-stack' window showing the stack: ==>S: S1 and 0: 0: 031 (move-to-food). A small diagram of a world with a robot and a goal is visible in the center.

Figura 9 – Impressão da direção do operador selecionado.

A Figura 10 apresenta a impressão da direção do operador selecionado concatenado com a variável “type”/tipo da comida, esse impressão é feita com o comando “(write (crLf) | Propose move | <dir> |, for | <type>)”.

The screenshot shows a Prolog IDE interface. On the left, a 'watch -P' window displays a list of steps and their corresponding directions concatenated with food types: west, west, south, west, north, west, west, south, west, north. The steps are numbered 14 through 24. Below the watch window are buttons for 'Step', 'Run', 'Run 1-p', 'Stop', 'Matches', 'Print <s>', 'Print <ts>', 'Clear', and 'Watch 1'. On the right, a 'print -depth 2 i3' window shows the current state: (I3 ^move M23) and (M23 ^direction north ^status complete). Below this, a 'state' window shows the current state: ==>S: S1, ==>S: S13 (state no-change), and ==>S: S15 (state no-change). At the bottom, there is a 'p-stack' window showing the stack: ==>S: S1, ==>S: S13 (state no-change), and ==>S: S15 (state no-change). A small diagram of a world with a robot and a goal is visible in the center.

Figura 10 – Impressão da direção e o tipo do operador selecionado.

2.3 Debugging commands

Neste sub-tópico é apresentado o debbuging da execução do agente com o código move-to-food.soar. Entre as análises, pode-se observar prints, com profundidades da árvore de execução; análises da memória de trabalho pelo comando wmes; prints de posições específicas da memória; matches: listas de regras prontas para serem executadas separadas por três resultados, o-assertions (que serão operadores), i-assertions (que criarão os argumentos de suporte para os operadores) e retractions (que excluirão os argumentos de suporte). Nas Figuras 11 e 12 pode-se observar estas análises.

The screenshot displays a SOAR debugging environment. On the left, a code editor shows the execution of a 'Propose move south, for normalfood' rule. The code includes various assertions, matches, and retractions. The right panel shows the current state of the system, including a 'p-stack' with elements like 'S1' and '(0: 05 (move-to-food))'. Below the code editor, there is a control panel with buttons for 'Step', 'Run', 'Run 1-p', 'Stop', 'Matches', 'Print <s>', 'Print <ts>', 'Clear', and 'Watch 1'. At the bottom, there are additional buttons for 'Watch 3', 'Watch 5', 'Init-soar', 'Source', 'cd', 'Excise all', and 'Remote'. The bottom status bar shows 'scratch_pad' and 'edit_production'.

Figura 11 – Debugging commands- 1.

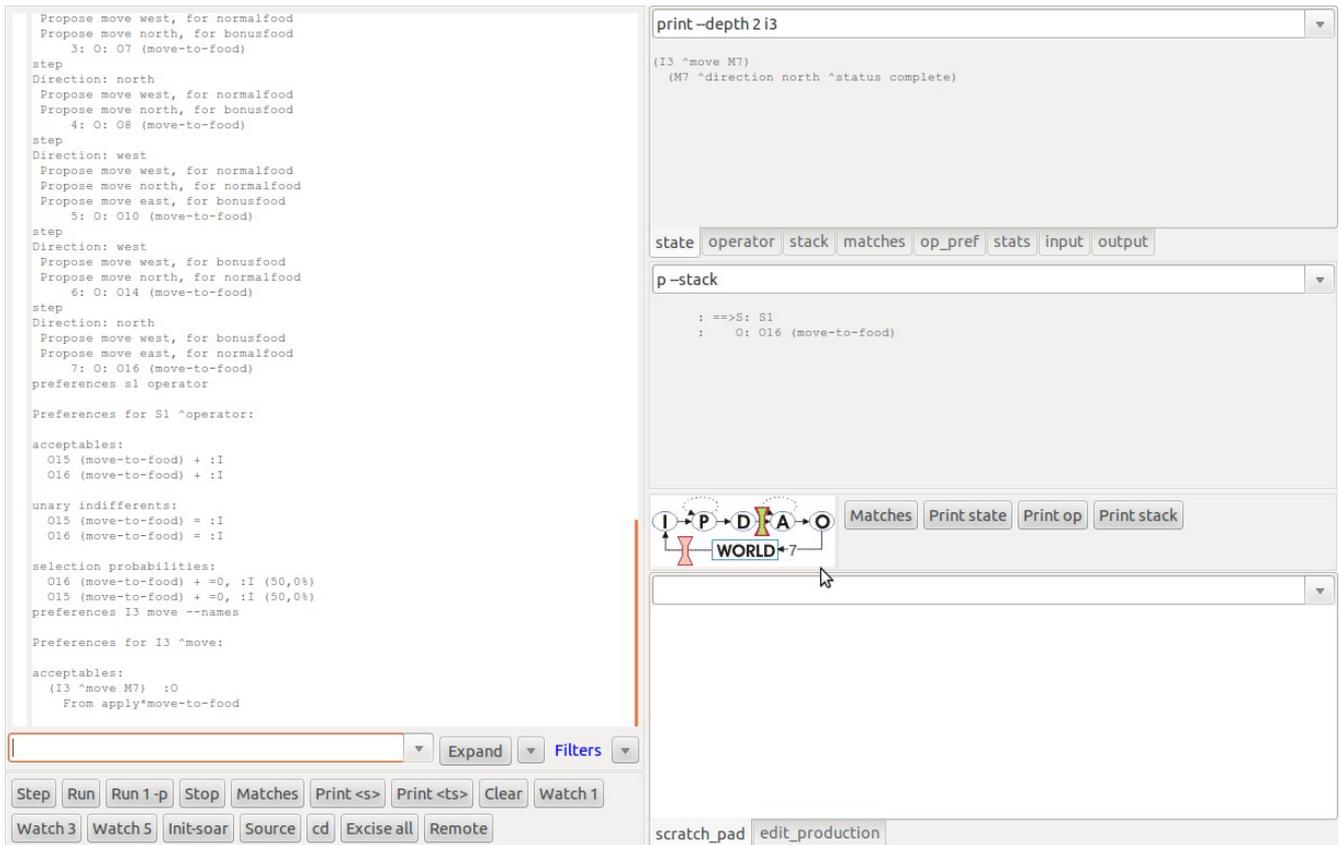


Figura 12 – Debugging commands- 2.

2.4 Semantic errors

Neste sub-tópico serão apresentadas as análises direcionadas aos erros semânticos. A Figura 13 apresenta o código que será analisado.

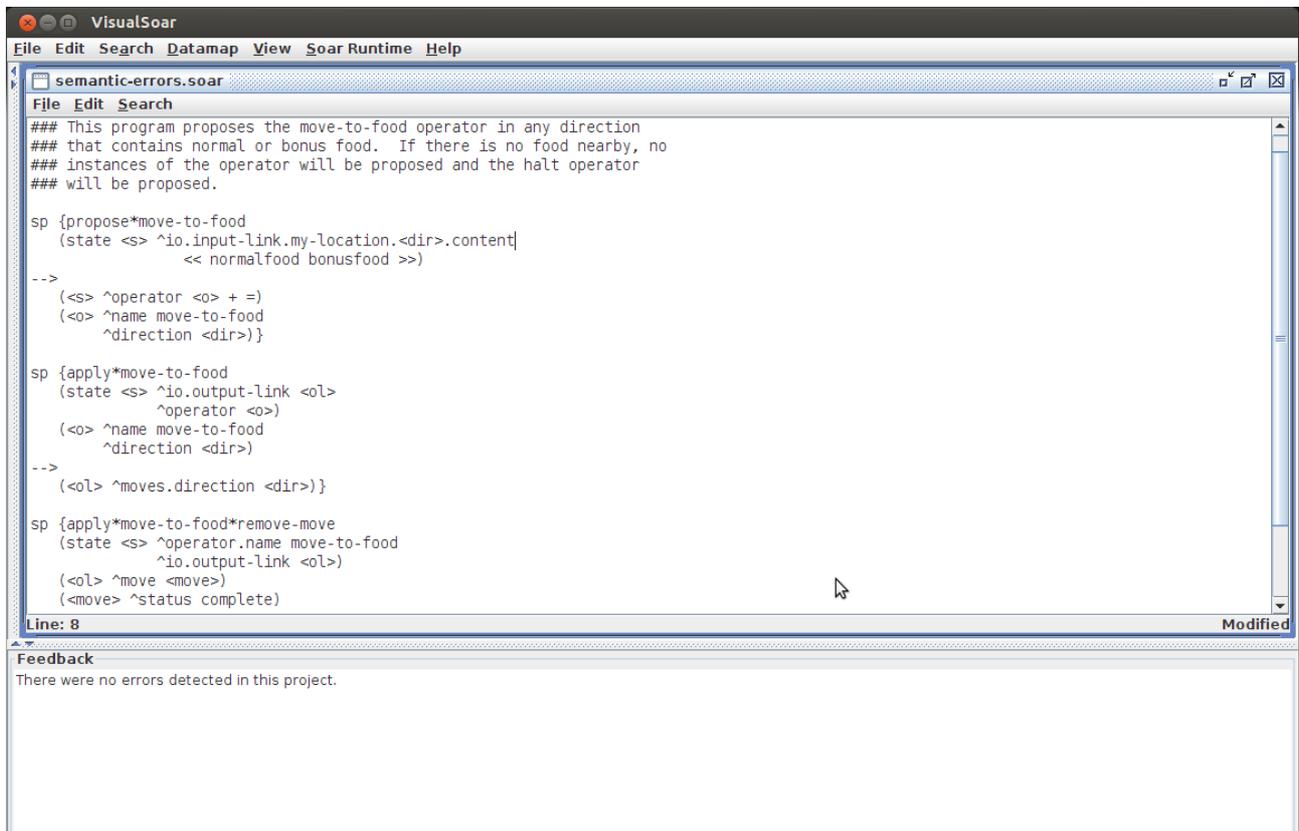


Figura 13 – Código erro semântico.

No caso específico foi necessário apenas a correção de dois comandos que foram trocados: o `contant` por `content` e o `moves` por `move`. Observamos o erro gerrado na Figura 14 que não seleciona nenhum operador para realizar alguma ação e o erro corrigido na Figura 15.

```

step
source {/home/mateus/Dropbox/Mestrado/IA006/SoarTutorial_9.3.2-Linux_64bit/Agem
Total: 3 productions sourced.
1: ==>S: S13 (state no-change)
matches propose*move-to-food
1 (state <s> ^io <i*1>)
1 (<i*1> ^input-link <i*2>)
1 (<i*2> ^my-location <m*1>)
6 (<m*1> ^<dir> <d*1>)
>>> (<d*1> ^contant { << normalfood bonusfood >> <c*1> })
0 complete matches.

```

print -depth 2 i3

state operator stack matches op_pref stats input output

p-stack

: ==>S: S1
: ==>S: S13 (state no-change)

WORLD 1

scratch_pad edit_production

Figura 14 – Erro semântico - 1.

```

step
source {/home/mateus/Dropbox/Mestrado/IA006/aula3/semantic-errors.soar}***
Total: 3 productions sourced.
1: 0: 04 (move-to-food)
step
2: 0: 07 (move-to-food)
step
3: 0: 08 (move-to-food)
matches propose*move-to-food
1 (state <s> ^io <i*1>)
1 (<i*1> ^input-link <i*2>)
1 (<i*2> ^my-location <m*1>)
6 (<m*1> ^<dir> <d*1>)
3 (<d*1> ^content { << normalfood bonusfood >> <c*1> })
3 complete matches.
matches
0 Assertions:
  apply*move-to-food*remove-move [S1]
  apply*move-to-food [S1]
1 Assertions:
Retractions:

```

print -depth 2 i3

(I3 ^move M3)
(M3 ^direction south ^status complete)

state operator stack matches op_pref stats input output

p-stack

: ==>S: S1
: 0: 08 (move-to-food)

WORLD 3

scratch_pad edit_production

Figura 15 – Erro semântico - 2.

3 Atividade 3

Neste t3pico s3o utilizadas t3cnicas para evitar que o agente busque comida com mais efici3ncia. Os argumentos de prefer3ncias auxiliam nesta etapa. Estes operadores s3o: +, -, >, < e =; respectivamente aceit3vel, rejeitar, melhor/maior, pior/menor e indiferente (utilizado para fun33o rand3mica).

As Figuras 16 e 17 apresentam a execu33o dos agentes com a utiliza33o dos argumentos de prefer3ncia, para a otimiza33o de busca por comidas com maior recompensa, ou seja, comida b3nus. Se n3o possuir comida b3nus na vizinha3a o agente busca por comida normal e assim por diante.

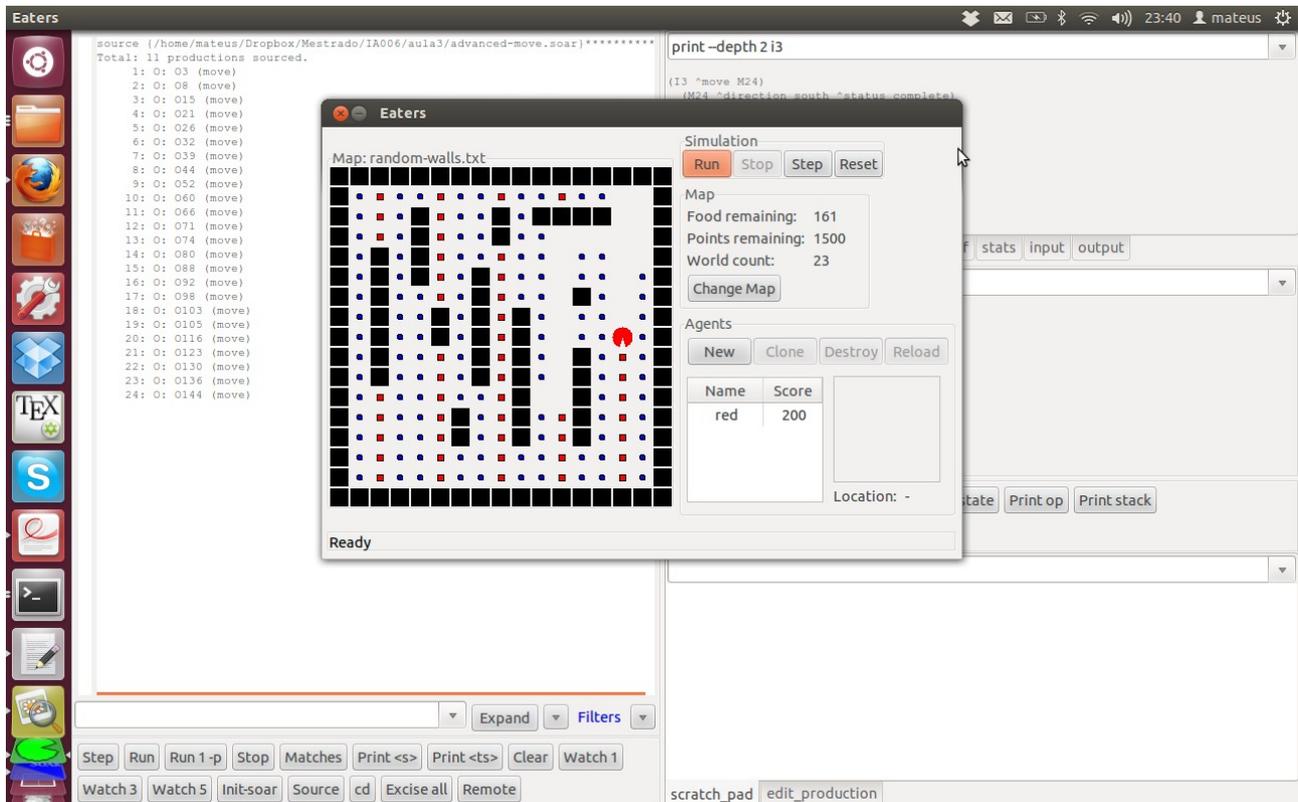


Figura 16 – Operador com movimento avan3ado - 1.

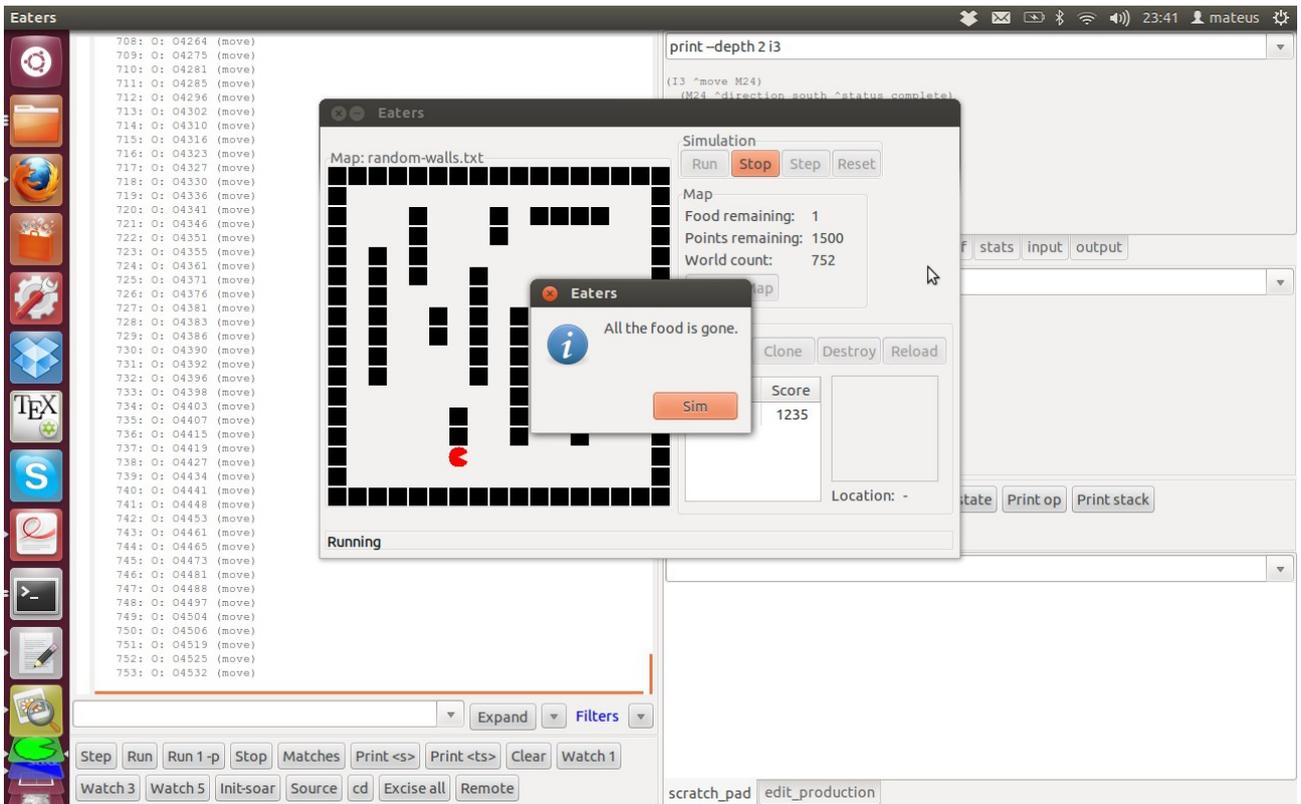


Figura 17 – Operador com movimento avançado - 2.

A Figura 18 apresenta o agente com operador de salto. Cada salto tem o mesmo custo que uma comida normal.

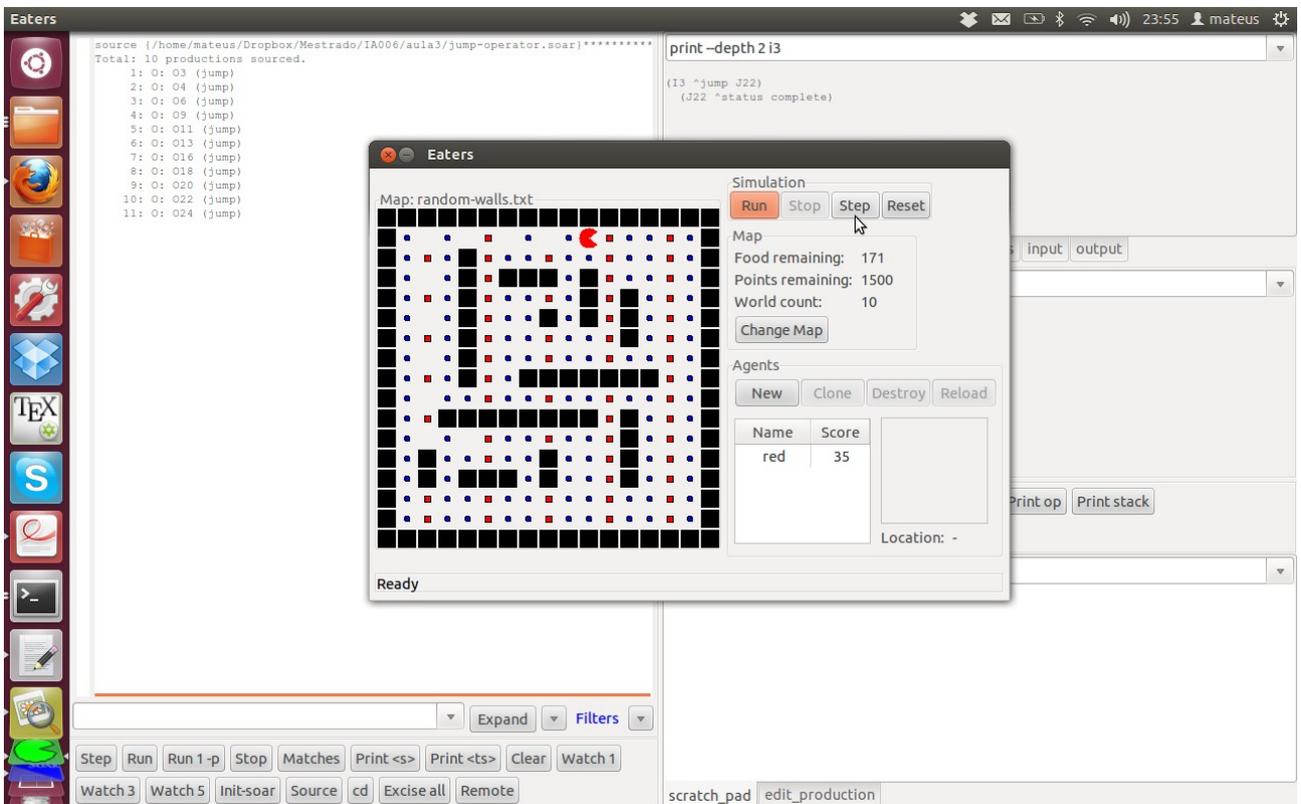


Figura 18 – Agente com o operador Jump.

4 Atividade 4

Para realizar esta tarefa, foi testado duas alternativas sem sucesso. A primeira foi, tentar forçar o agente a preferência por casas horizontais, caso as melhores opções sejam casas com o conteúdo vazio (empty):

```
sp {select*move*prefer-horizontal
(state <s> ^operator <o> +)
(<o> ^name move
^content empty
^direction <<west east>>)
-->
(<s> ^operator <o> >)}
```

Esta alternativa falhou, pois assim que o agente atinge paredes (wall) ele é obrigado a subir e se na subida ele também encontra um limitador, isso acaba gerando a sua volta, demorando mais tempo para o mesmo sair das regiões vazias. Consequentemente ficando com menos pontos competindo com os agentes avançados.

A segunda tentativa, frustrada também, foi realizar o *merge* entre os agentes que possuem salto e movimento com os agente que guardam a última direção. Porém, assim que é gravado a última direção, as proposições de ações deixam de ser escolhidas, gerando estate-no-change. Para tentar contornar esse problema, foi tentado gravar o objeto inteiro no output-link, porém foi encontrada uma dificuldade de se apagar o último e manter apenas um operador gravado na memória. Essa tarefa, não estava presente em nenhum tutorial até o momento. A tarefa de apagar um estado criado em <s> foi possível, porém não funcionava para o *merge*.