NOME: GABRIEL ALESSI POSONSKI RA: 2259583 DATA: 25/06/2023

Lógica para Computação - 2023.1
AC-13 - Implementação de Agentes em JaCaMo

Essa implementação de agentes tem como objetivo simular a comunicação de um agente controlador de trânsito (CT) que envia sinais para um agente de veículo autônomo (VA). O cenário seria um cruzamento que inicialmente se encontra ocupado e impede a passagem do VA. Como não foi utilizado um ambiente, a simulação foi feita através da manipulação das crenças e utilização de impressões no console do sistema.

Inicialmente temos o código do arquivo .jcm:

```
14
15  mas ac13 {
16
17      agent ct
18
19      agent va
20
21
22
23  }
24
```

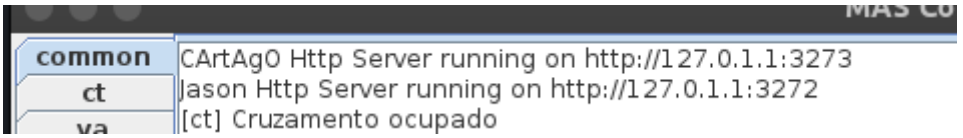
Arquivo ac13.jcm

Primeiramente o agente CT é inicializado e realiza todos os procedimentos programados. No seu código temos as seguintes instruções:

```
1   // Agent ct in project ac13
2
3   /* Initial beliefs and rules */
4
5   cruzamento(ocupado).
6
7   /* Initial goals */
8
9   !verificarSensor.
10
11  /* Plans */
12
13  +!verificarSensor : cruzamento(ocupado) <-
14                          .print("Cruzamento ocupado");
15                          .send(va, achieve, frear);
16                          -cruzamento(ocupado);
17                          +cruzamento(livre);
18                          !verificarSensor.
19
20  +!verificarSensor : cruzamento(livre) <-
21                          .print("Cruzamento livre");
22                          .send(va, achieve, acelerar).
23
24  { include("$jacamoJar/templates/common-cartago.asl") }
25  { include("$jacamoJar/templates/common-moise.asl") }
```

Arquivo ct.asl

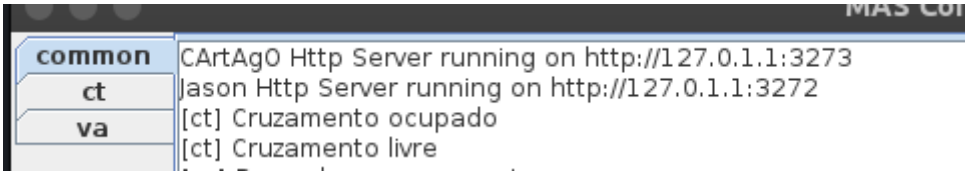
Inicialmente o agente crê que o cruzamento está ocupado, e seu objetivo é o de verificar a situação do sensor, que definirá qual é a ação a ser tomada pelo VA. Há 2 planos para 2 situações diferentes: na linha 13, o plano é ativado se houver a crença “cruzamento(ocupado)”, o que é o nosso caso, então se executam as primeiras linhas de instruções:



impressão no Console MAS

Em seguida, um sinal é enviado para o VA, solicitando que o mesmo freie, porém devido às instruções do arquivo .jcm, o programa primeiro executará por completo o arquivo do agente “ct” e só então partirá para o “va”. Como são poucas instruções, iremos finalizar as execuções deste agente e depois partiremos para o outro.

Após isso, nas linhas 16 e 17 o agente sofre alteração nas suas crenças, deixando de acreditar que o cruzamento está ocupado e passando a crer que ele está livre, isto é nada mais do que uma simulação do ambiente que o agente estaria sondando. Em seguida, ocorre uma chamada do plano “verificarSensor”, porém agora, com o agente tendo uma crença diferente da inicial, o que acarreta na execução da segunda opção do plano, que faz algo semelhante ao primeiro, porém com uma impressão diferente e enviando o sinal de “acelerar” para o VA:



impressão no Console MAS

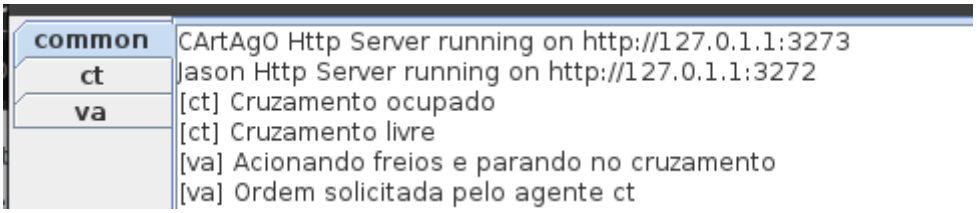
Então se encerram as instruções do agente “ct” e passamos a analisar o agente “va”, que tem as seguintes instruções:

```
1  // Agent va in project ac13
2
3  /* Initial beliefs and rules */
4  posicao(cruzamento).
5
6  /* Initial goals */
7
8
9  /* Plans */
10
11 +!acelerar[source(Ag)] : posicao(cruzamento) <-
12         .print("Desativando freio e acelerando");
13         .print("Avancando no cruzamento");
14         .print("Ordem solicitada pelo agente ", Ag).
15
16 +!frear[source(Ag)] : posicao(cruzamento) <-
17         .print("Acionando freios e parando no cruzamento");
18         .print("Ordem solicitada pelo agente ", Ag).
19
20 { include("$jacamoJar/templates/common-cartago.asl") }
21 { include("$jacamoJar/templates/common-moise.asl") }
22
```

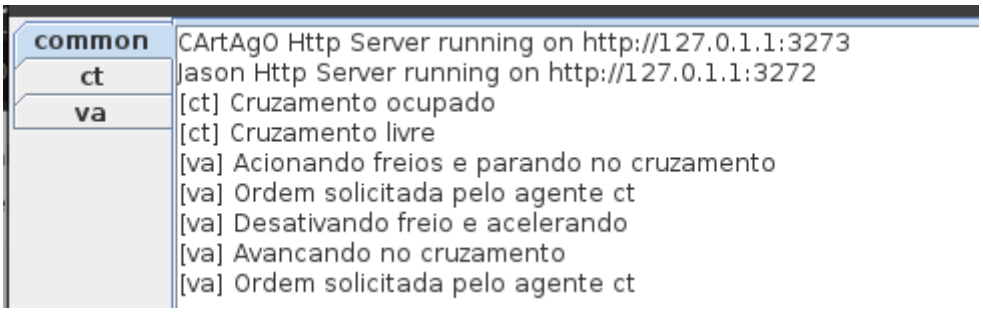
Arquivo va.asl

O agente inicia tendo a crença de que está no cruzamento, que é o que define se ele irá executar os objetivos frear e acelerar, além de ser necessário

chamar esses objetivos. Anteriormente o agente ct já realizou isto, e as instruções do “va” ficaram armazenadas na memória do sistema, que agora irá executá-las. Primeiro, os comandos de frear:



E finalmente, o comando para acelerar:



A execução poderia ser simultânea se ambos estivessem no mesmo arquivo, porém decidi como forma de desafio, tentar realizar os comandos em arquivos diferentes.