Sistema Multiagentes Utilizando a Linguagem AgentSpeak(L) para Criar Estratégias de Armadilha e Cooperação em um Jogo Tipo PacMan

Alisson Rafael Appio Prof. Jomi Fred Hübner - Orientador

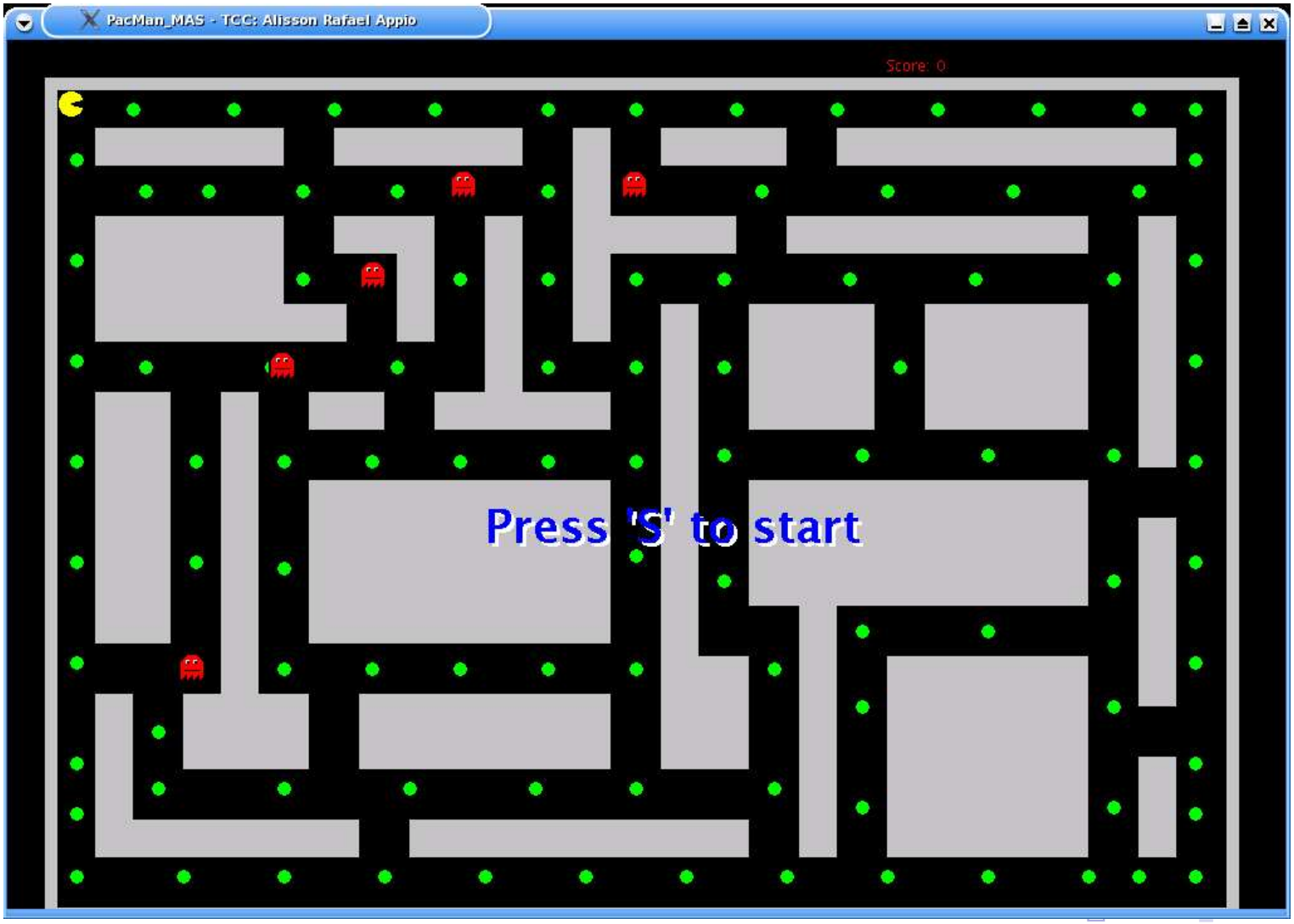
Universidade Regional de Blumenau (FURB)
Centro de Ciências Exatas e Naturais
Departamento de Sistemas e Computação
Curso de Ciências da Computação

13 de dezembro de 2004

Introdução

- Jogos de computadores cada vez mais sofisticados
- Inteligência Artificial (IA)
- Entre os jogos mais populares, destaca-se o jogo PacMan
- Personagens do jogo PacMan: come-come e fantasma
- Criar estratégias de armadilha para jogo PacMan

Jogo PacMan_MAS



Objetivo do Trabalho

O objetivo deste trabalho é verificar a viabilidade do uso de técnicas de cooperação em SMA para resolver o problema de criar estratégias de armadilhas em um jogo do tipo PacMan.

Sistemas Multiagentes - SMA

Conceitos de SMA:

- Estudar o comportamento de um grupo de agentes, normalmente autonomos e distribuídos
- Cooperação

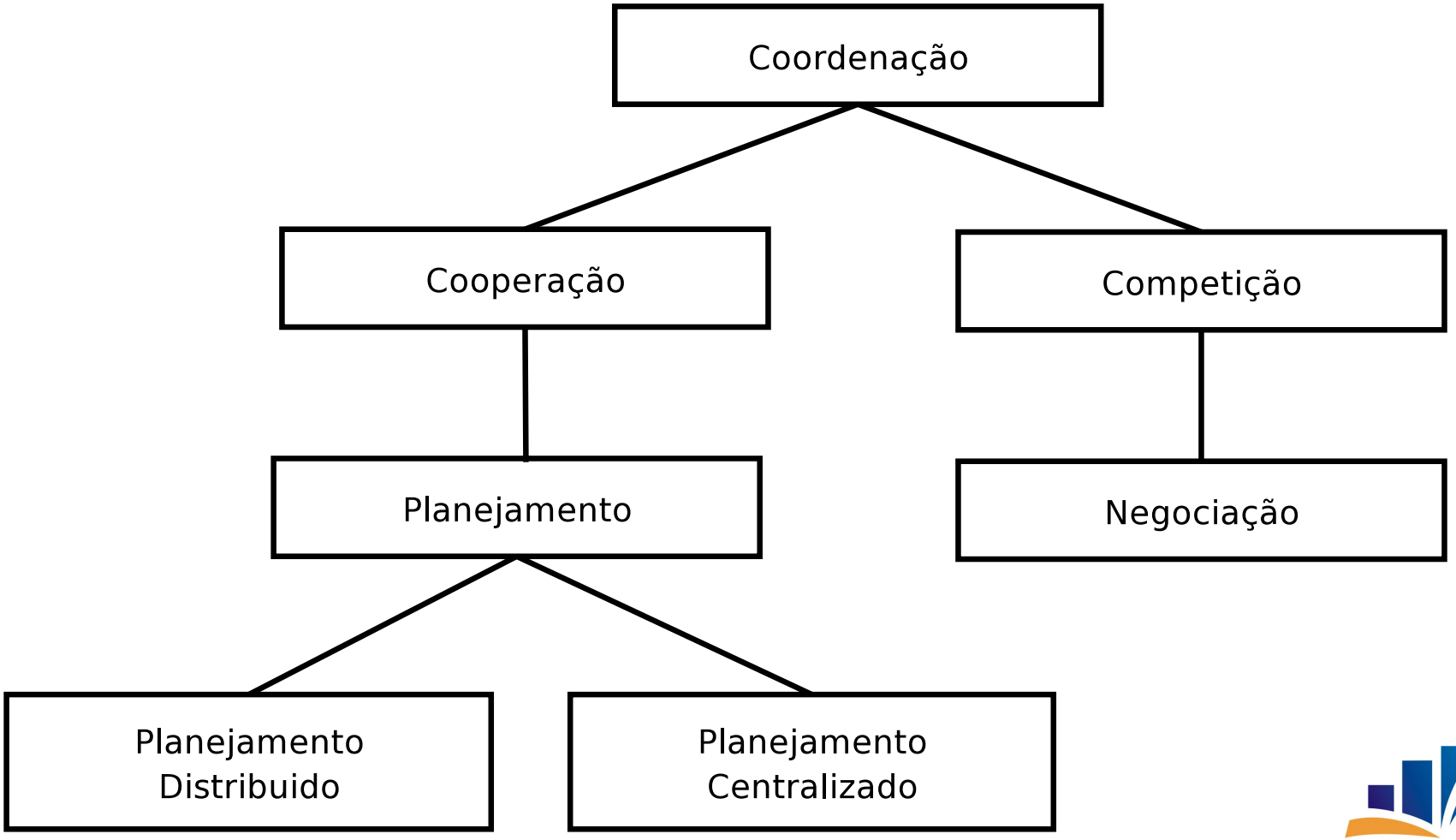
Coordenação

- Dependência entre as ações
- Impossibilidade de resolução de um problema por um único agente
- Coordenação é alcançada por meio da comunicação entre os agentes

Exemplo

Dois robôs precisam mover uma mesa de lugar

Tipos de Coordenação



Cooperação

- Cooperação é um tipo de coordenação entre os agentes que exercem ações com objetivo de atingir um bem social
- Trabalhar em equipes, aumentando a utilidade global do sistema e não sua utilidade individual

Exemplos

- Ambiente de um táxi. Objetivo evitar colisões com outros carros
- Tênis de duplas. Objetivo vencer a partida

Agentes

Um agente é uma entidade real ou virtual, inserida em um ambiente.
Os agentes podem ser divididos em duas categorias:

- Reativos (comportamento estímulo-resposta; não possuem representação interna do mundo)
- Cognitivos

Agentes - Cognitivos

Agentes cognitivos possuem comportamentos complexos onde eles deliberam e negociam suas ações com os outros agentes. Alguns aspectos importantes de agentes cognitivos:

- Percepção
- Ação
- Comunicação
- Raciocínio
- Aprendizagem

Arquitetura BDI

A arquitetura Belief-Desires-Intentions é composta por três atitudes mentais

- Crenças: tudo aquilo que o agente sabe sobre si mesmo, os outros agentes e o ambiente
  - Gerado pela função de revisão de crenças
- Desejos: tudo aquilo que o agente deseja
  - Estados do mundo que são desejados
  - Consideram as crenças para verificar sua viabilidade
- Intenções: seqüência de ações que o agente se compromete a executar para atingir sua meta

Linguagem AgentSpeak(L)

- Crenças
- Planos
- Eventos Ativadores
- Ações Básicas (ações que são executadas no ambiente)
- Ações Internas

Linguagem AgentSpeak(L) - Planos agente barata

Os planos são especificados, como:

- evento ativador: `+perigo`
- contexto: `temLugarEscuro(L)`
- corpo: `fugir(L)`
- `+ perigo: temLugarEscuro(L) ← fugir(L).`

Linguagem AgentSpeak(L) - Planos agente barata

Os planos são especificados, como:

- evento ativador: `+perigo`
- contexto: `temLugarEscuro(L)`
- corpo: `fugir(L)`
- `+ perigo: temLugarEscuro(L) ← fugir(L).`

Linguagem AgentSpeak(L) - Planos agente barata

Os planos são especificados, como:

- evento ativador: `+perigo`
- contexto: `temLugarEscuro(L)`
- corpo: `fugir(L)`
- `+ perigo: temLugarEscuro(L) ← fugir(L).`

Linguagem AgentSpeak(L) - Planos agente barata

Os planos são especificados, como:

- evento ativador: `+perigo`
- contexto: `temLugarEscuro(L)`
- corpo: `fugir(L)`
- `+ perigo: temLugarEscuro(L) ← fugir(L).`

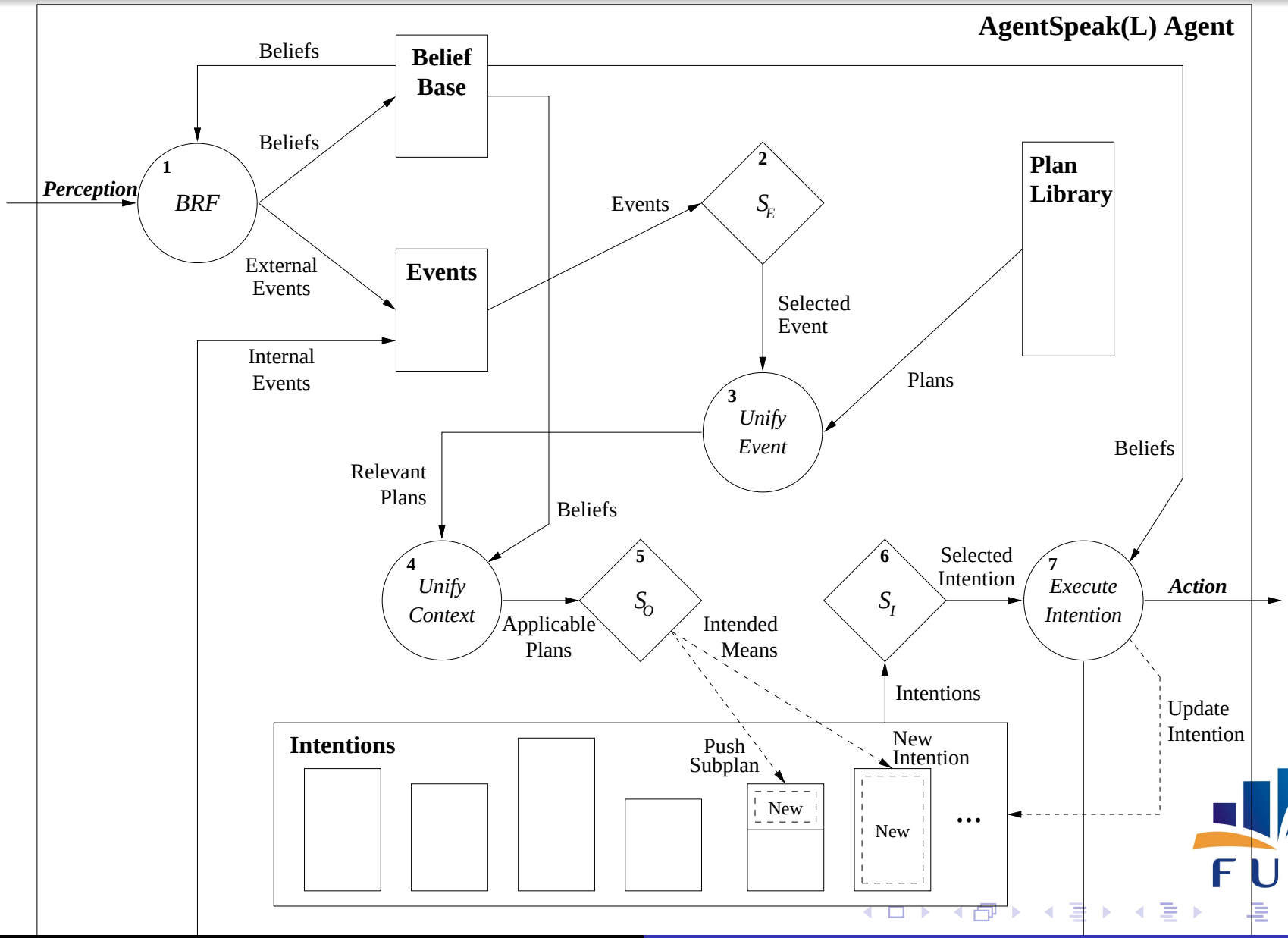
- Tipos de objetivos
  - objetivos de teste: ? `temLugarEscuro(L)`
  - objetivos de realização: ! `pararDeAndar`
- Tipos de eventos
  - evento externo: percebibos pela mudança do ambiente
    + `temLugarEscuro(norte)`
    - `temLugarEscuro(sul)`
  - evento interno: +! `pararDeAndar`
- Ações básicas no ambiente
  - `fugir(L)`
- Ações internas
  - `.somar(1,2,X)`

Planos AgentSpeak(L) - Agente barata

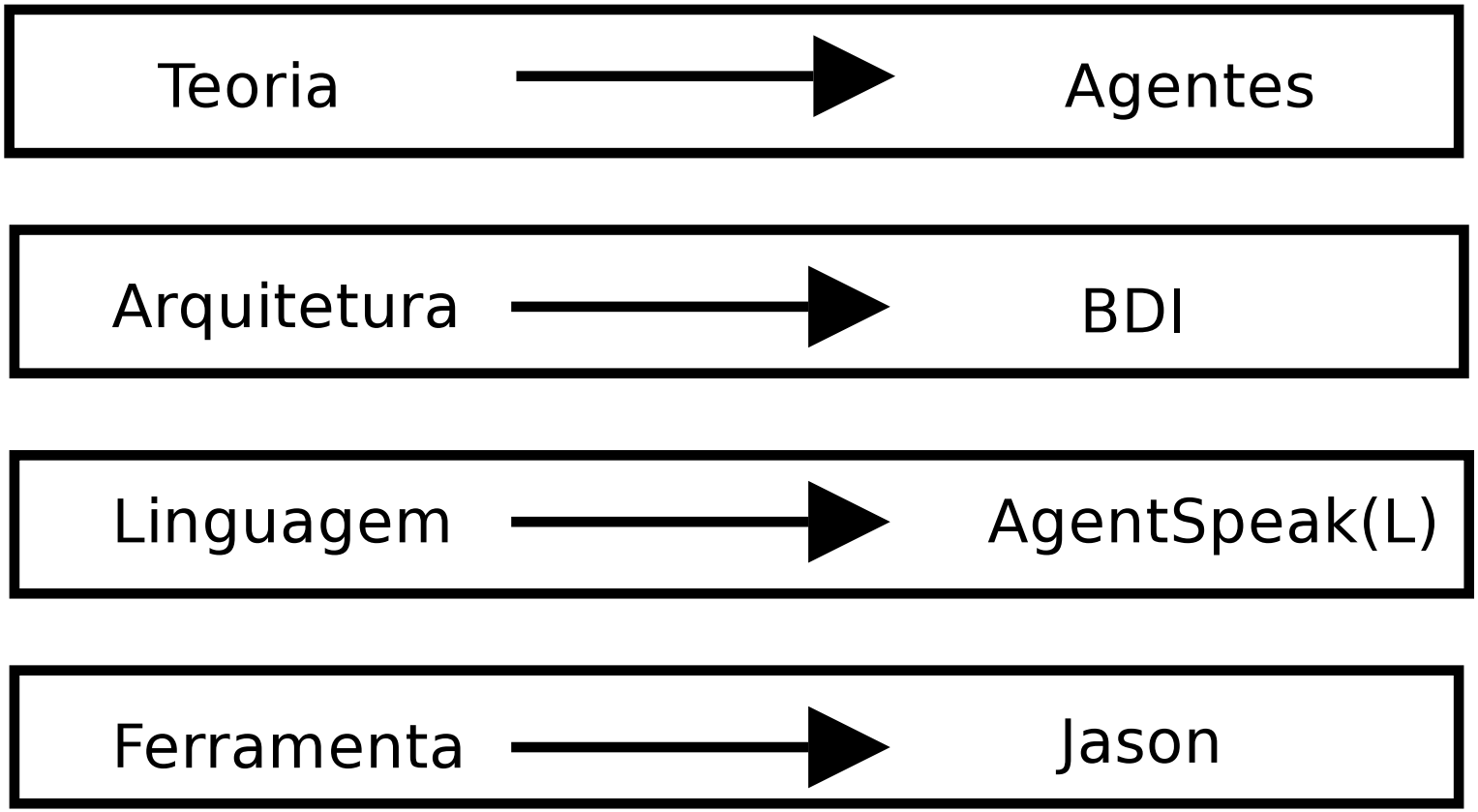
```
+ perigo:  temLugarEscuro(L) ← fugir(L).

+ perigo:  true ← !  pararDeAndar.

+!  pararDeAndar ← fingirMorta.
```

Jason

- Implementação de um interpretador AgentSpeak(L)
- Suporte ao desenvolvimento de ambientes
- Comunicação baseada em atos de fala
- Possibilidade de executar o SMA em uma rede (utilizando SACI)
- Biblioteca de ações internas


AgentSpeak(L) Agent
Belief Base
Beliefs
Beliefs
1
BRF
Perception
External Events
Events
Internal Events
2
S_E
Events
Selected Event
Plan Library
Plans
3
Unify Event
Relevant Plans
Beliefs
4
Unify Context
Beliefs
5
S_O
Applicable Plans
Intended Means
6
S_I
Selected Intention
7
Execute Intention
Action
Intentions
Update Intention
Intentions
Push Subplan
New Intention
New
New
...

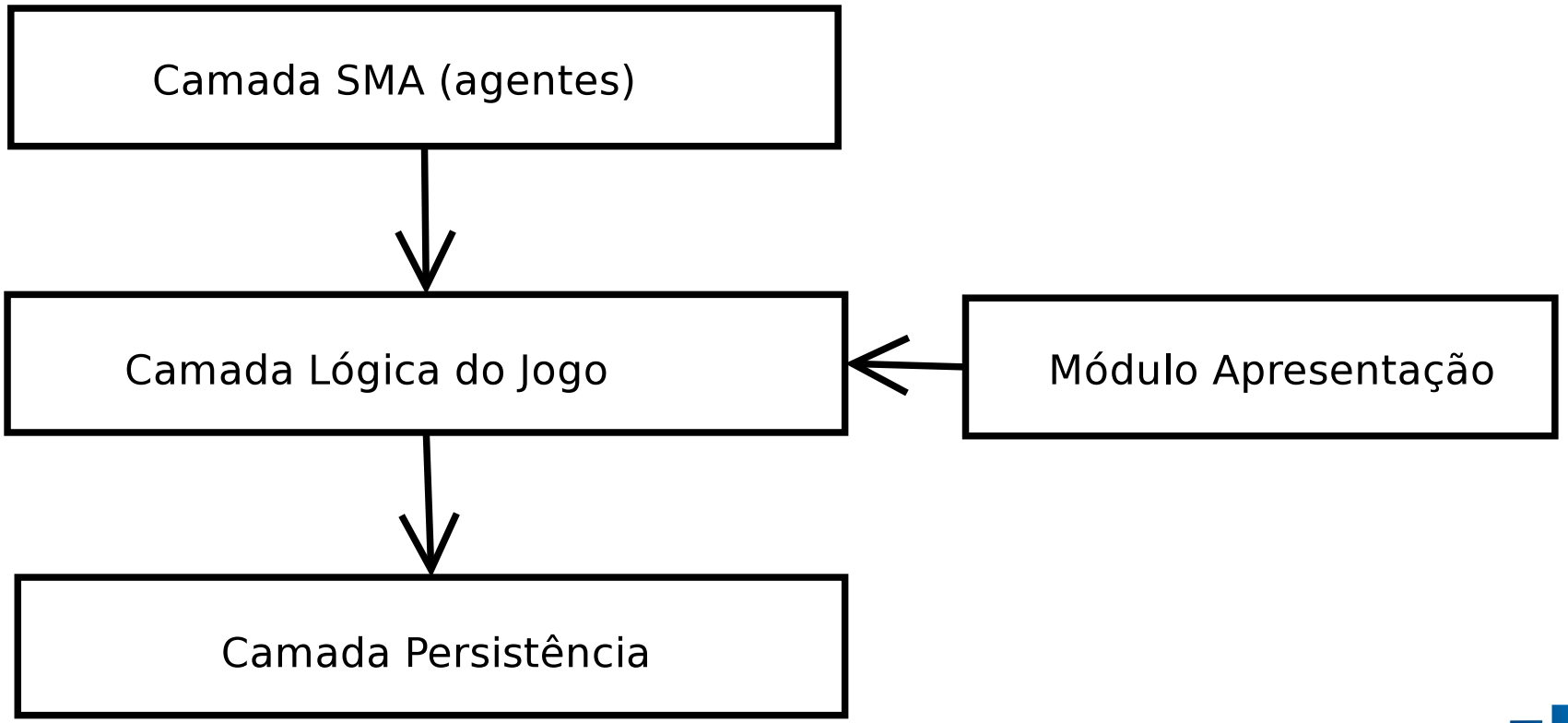
Resumo

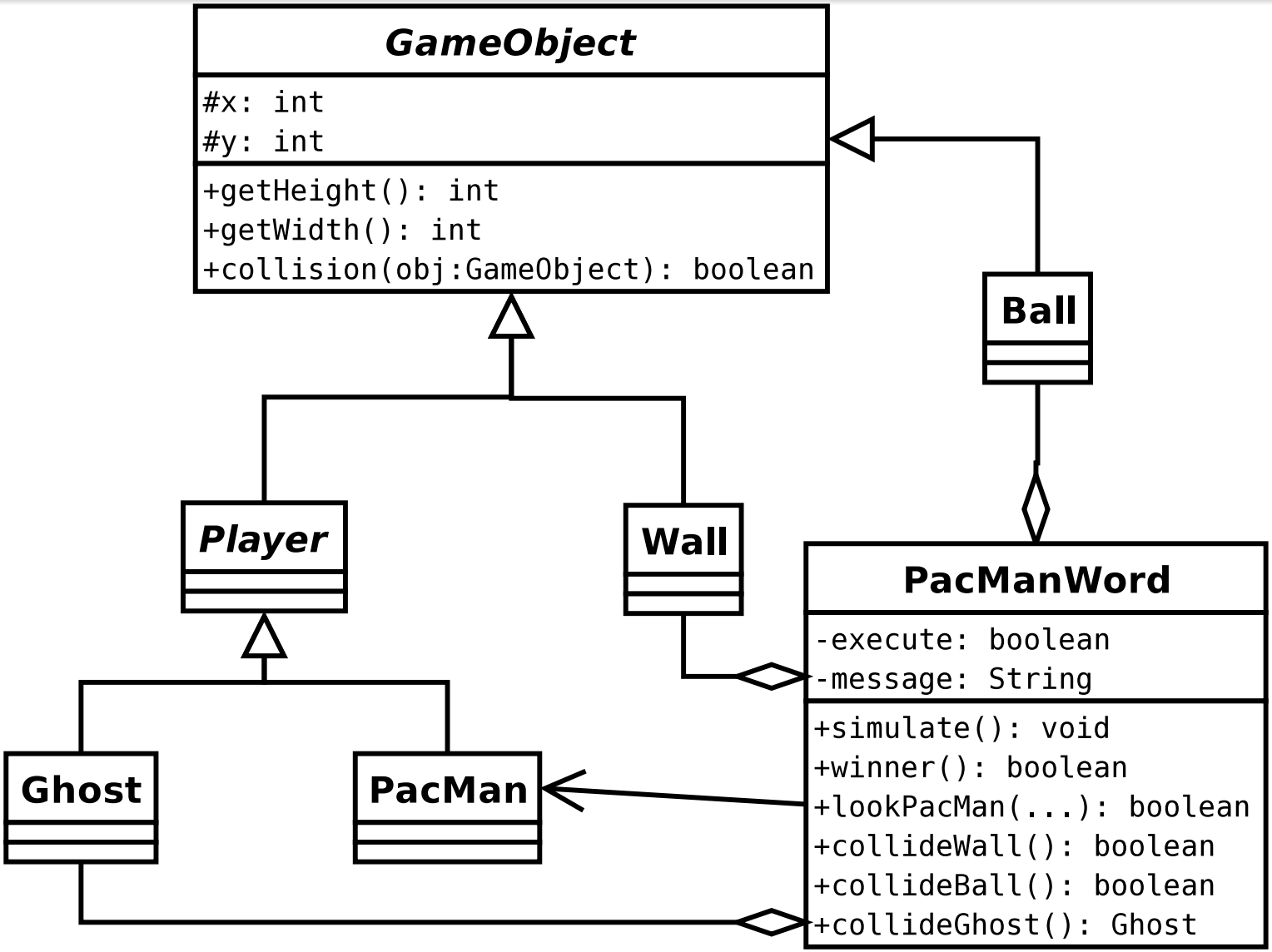

Teoria
Agentes
Arquitetura
BDI
Linguagem
AgentSpeak(L)
Ferramenta
Jason

Requisitos Principais do Jogo

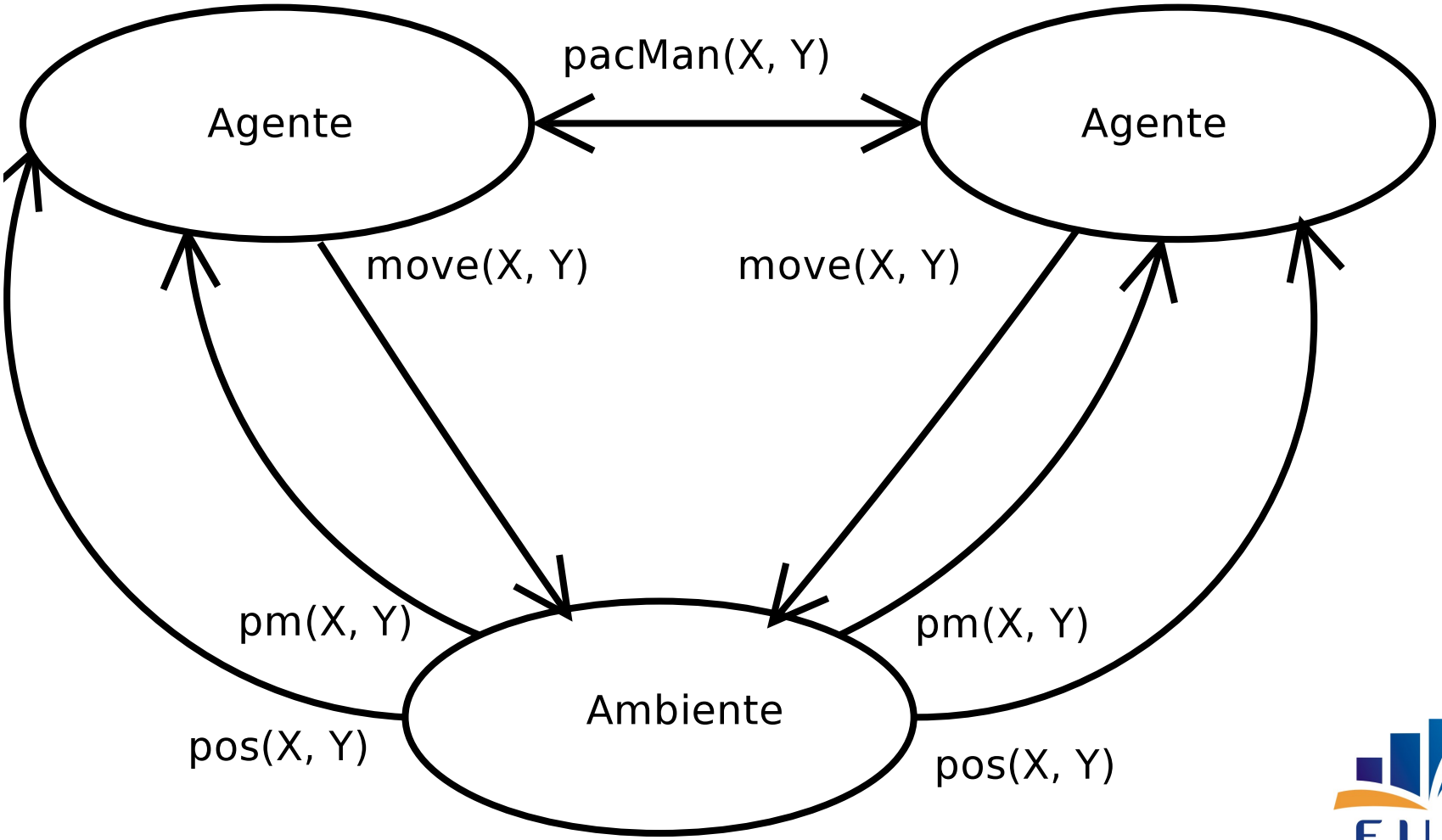
- Cenário do jogo (ambiente) deve ser desenvolvido na linguagem Java
- Utilizar o paradigma de programação orientada a agentes com base em uma linguagem particular chamada AgentSpeak(L)
- Desenvolver agentes BDI que devem criar e executar estratégias de armadilhas para pegar o come-come.

Esboço Geral do Sistema

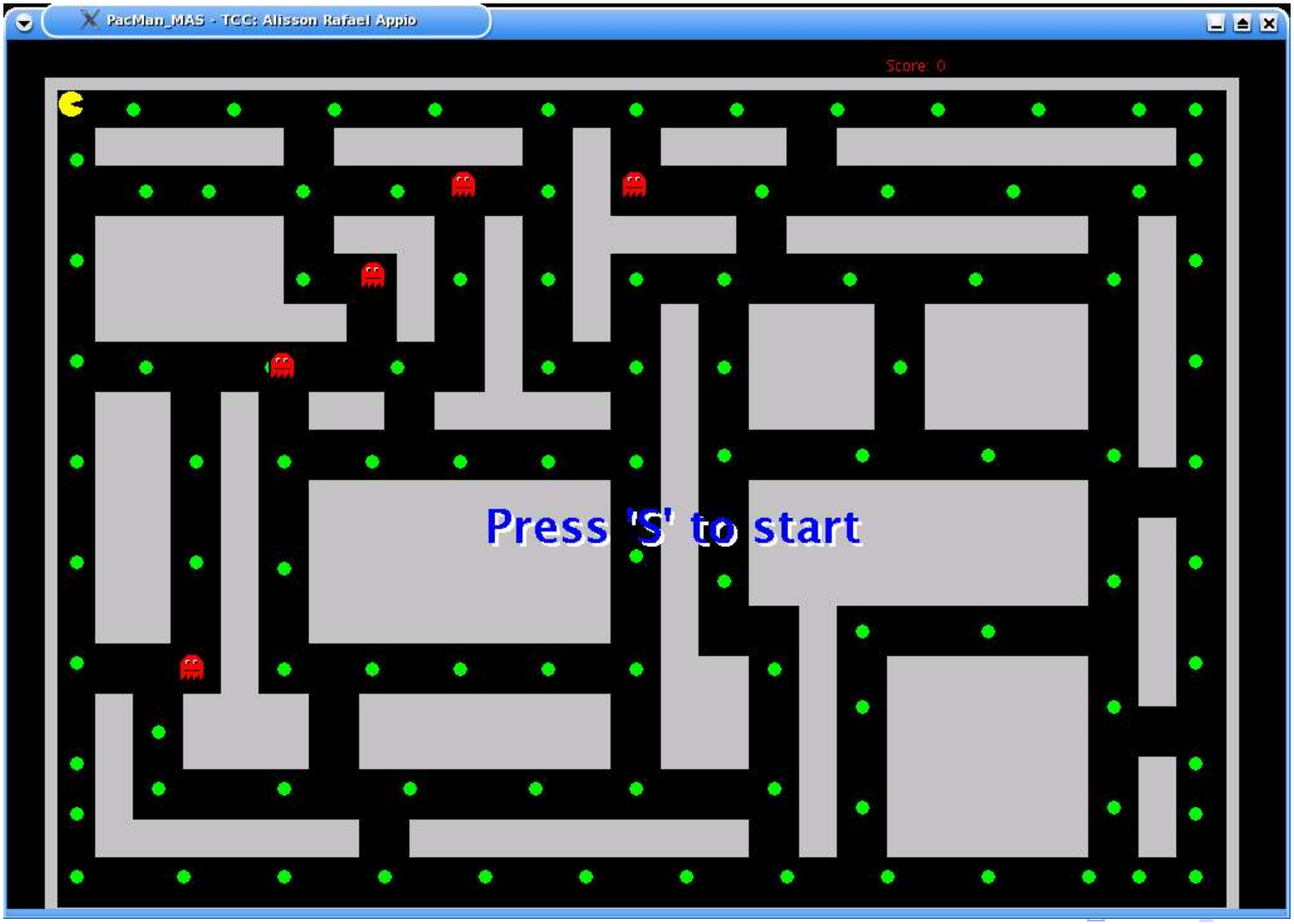



GameObject
#x: int
#y: int
+getHeight(): int
+getWidth(): int
+collision(obj:GameObject): boolean
Ball
Player
Wall
PacManWord
-execute: boolean
-message: String
+simulate(): void
+winner(): boolean
+lookPacMan(...): boolean
+collideWall(): boolean
+collideBall(): boolean
+collideGhost(): Ghost
Ghost
PacMan

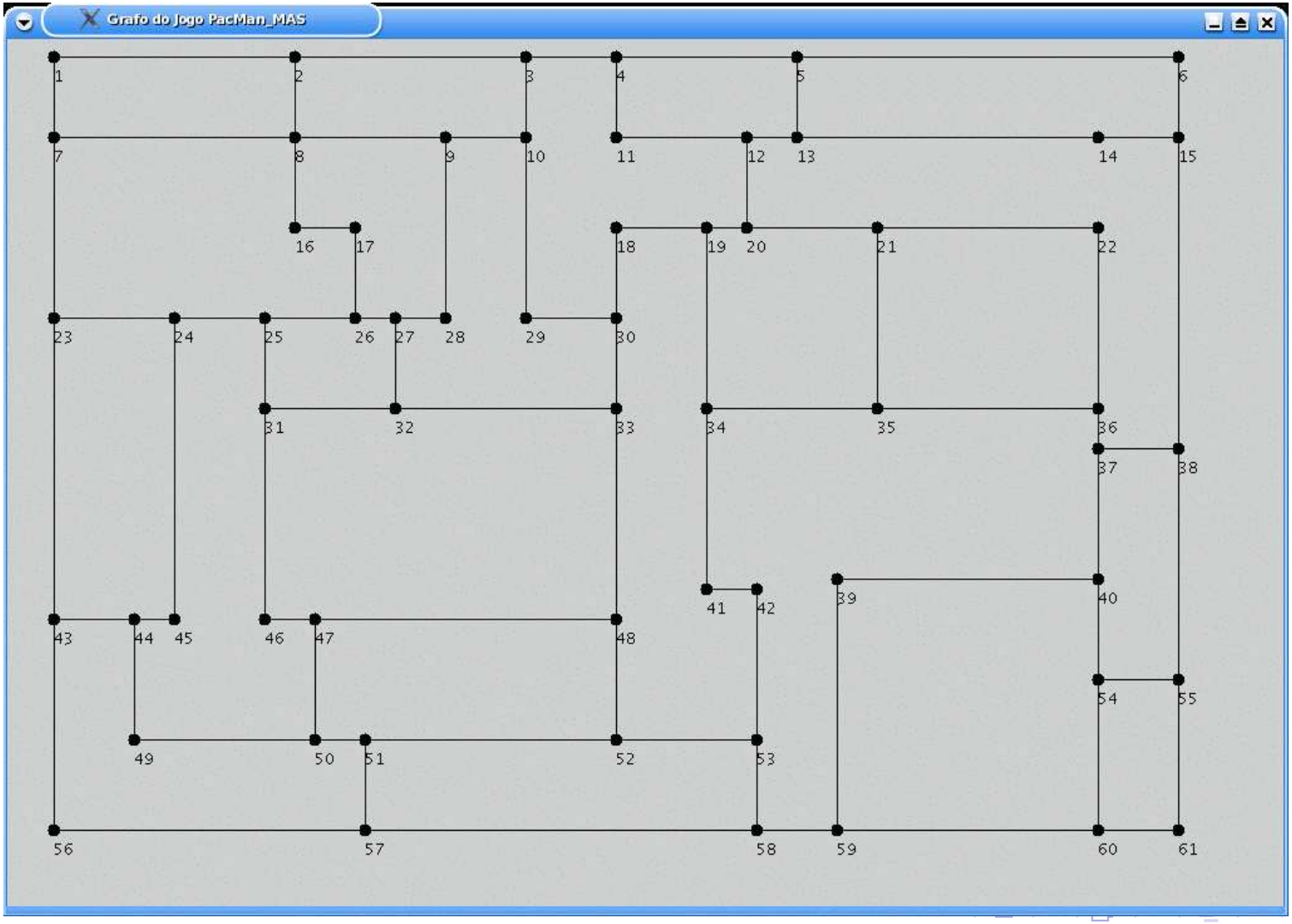
Percepções do agente



Jogo - Grafo



Jogo - Grafo


Grafo do Jogo PacMan_MAS
1 2 3 4 5 6
7 8 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29 30
31 32 33 34 35 36
37 38
39 40 41 42
43 44 45 46 47 48
54 55
49 50 51 52 53
56 57 58 59 60 61

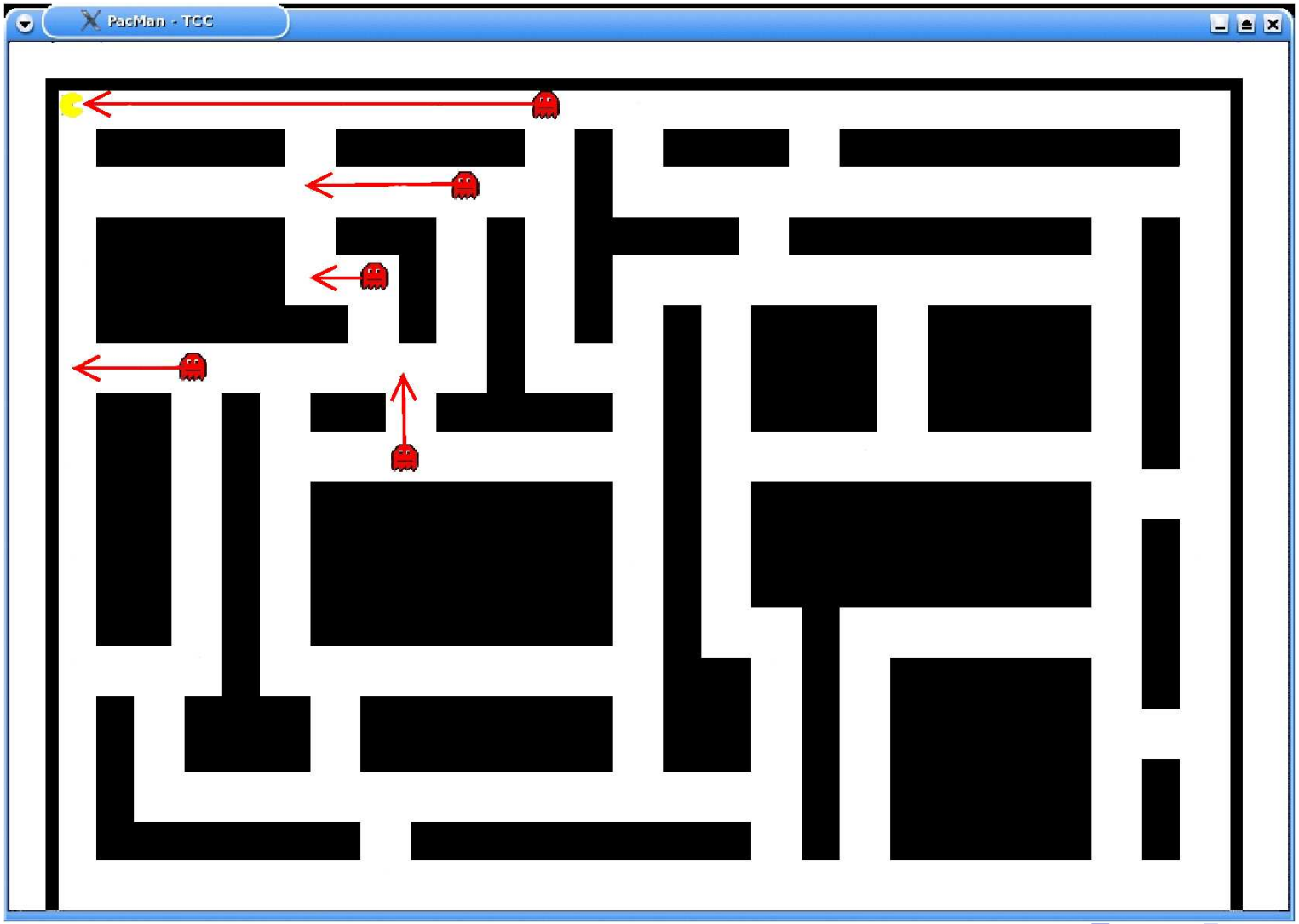
Planos de movimentação do agente

```
+ pos(X, Y) : not(moving) & not(cooperate) ←
      .nodoMaisProximo(X, Y, null, null,
          Xresult, Yresult, CornerGoal);
          + moving;
          !go(Xresult, Yresult, CornerGoal).
+ pos(X, Y) : cooperate ←
      ?  pacMan(Xpm, Ypm);
      .nodoMaisProximo(X, Y, Xpm, Ypm, Xresult,
          Yresult, CornerGoal);
      - cooperate;
      + moving;
      !  go(Xresult, Yresult, CornerGoal).
```

Planos de movimentação do agente

```
+! go(X, Y, CornerGoal):  pos(X, Y) ←
        - moving;
        + pos(X, Y).

+! go(X, Y, CornerGoal):  true ←
        ?  pos(Xpos, Ypos);
        .nextPosition(Xpos, Ypos, CornerGoal,
            Xresult, Yresult);
        move(Xresult, Yresult);
        !go(X, Y, CornerGoal).
```

Estratégia de Armadilha no Jogo



Plano da percepção do come-come

```
+ pm(X, Y) : true ←
      ?  pos(Xg, Yg);
      .nodoMaisLonge(Xg, Yg, X, Y, Xresult,
          Yresult, GoalEsquina);
      + moving;
      !  sendCoordinatePacMan(X, Y, ghost, 1);
      !  go(Xresult, Yresult, GoalEsquina).
```

Plano para entrar em modo de cooperação

```
+ pacMan(X,Y) : true
          ← + cooperate.
```

Algoritmo do loop principal do Jogo - método simulate

```
1  while não é fim de jogo do
2      manda pacman andar;
3      if pacman colidir com uma bola then
4          muda estado da bola para invisível;
5          incrementa pontos pacman;
6      if todas as bolinhas estão invisíveis then
7          pacman venceu o jogo
8      if pacman colidir com parede then
9          manda pacman voltar;
10
11     else if pacman colidir com fantasma then
12         pacman perdeu o jogo;
```

Resultados

- Os agentes cooperam para criar armadilha. Porém quando os agentes estão muito longe do come-come, a armadilha não tem sucesso.
- Controle dos agentes descentralizado. Não existe um módulo gerente
- Fazer os agentes cooperarem não é uma tarefa trivial. Porém, é uma maneira mais sofisticada de evitar que o come-come vença o jogo
- Jogo ficou um pouco lento

Resultados

- Uma das maiores implementações com a ferramenta Jason
- A comunicação entre os agentes é feita de maneira transparente
- Ferramenta Jason em desenvolvimento, encontrados alguns bugs

Conclusões

- Jogos sofisticados podem ser concebidos através da arquitetura BDI
- Comportamento de cada agente é simples não tem plano de armadilha. Através da interação entre eles surgem as estratégias de armadilhas
- AgentSpeak(L) alto nível de abstação na especificação do agente

Extensões

- Desenvolver outras técnicas de armadilhas para este jogo
- Desenvolver jogos mais sofistivados
- Desenvolver personagens autônomos em um ambiente virtual
- Desenvolver sistemas de simulação de sociedade de seres humanos (interagindo uns com os outros)

Demonstração do Jogo

Demonstração do jogo PacMan_Mas

Bibliografia


Rodrigo Machado and Rafael H. Bordini.
Running AgentSpeak(L) agents on SIM_AGENT.
In *Intelligent Agents VIII – Proceedings of the Eighth International Workshop on Agent Theories, Architectures, and Languages (ATAL-2001)*, 2002.


Jorge A. R. Torres, Luciana P. Nedel, and Rafael H. Bordini.
Using the BDI architecture to produce autonomous characters in virtual worlds.
In *Proceedings of the Fourth International Conference on Interactive Virtual Agents (IVA 2003)*, 2003.


Gerhard Weiß.
Multiagent Systems.
MIT Press, 2000.