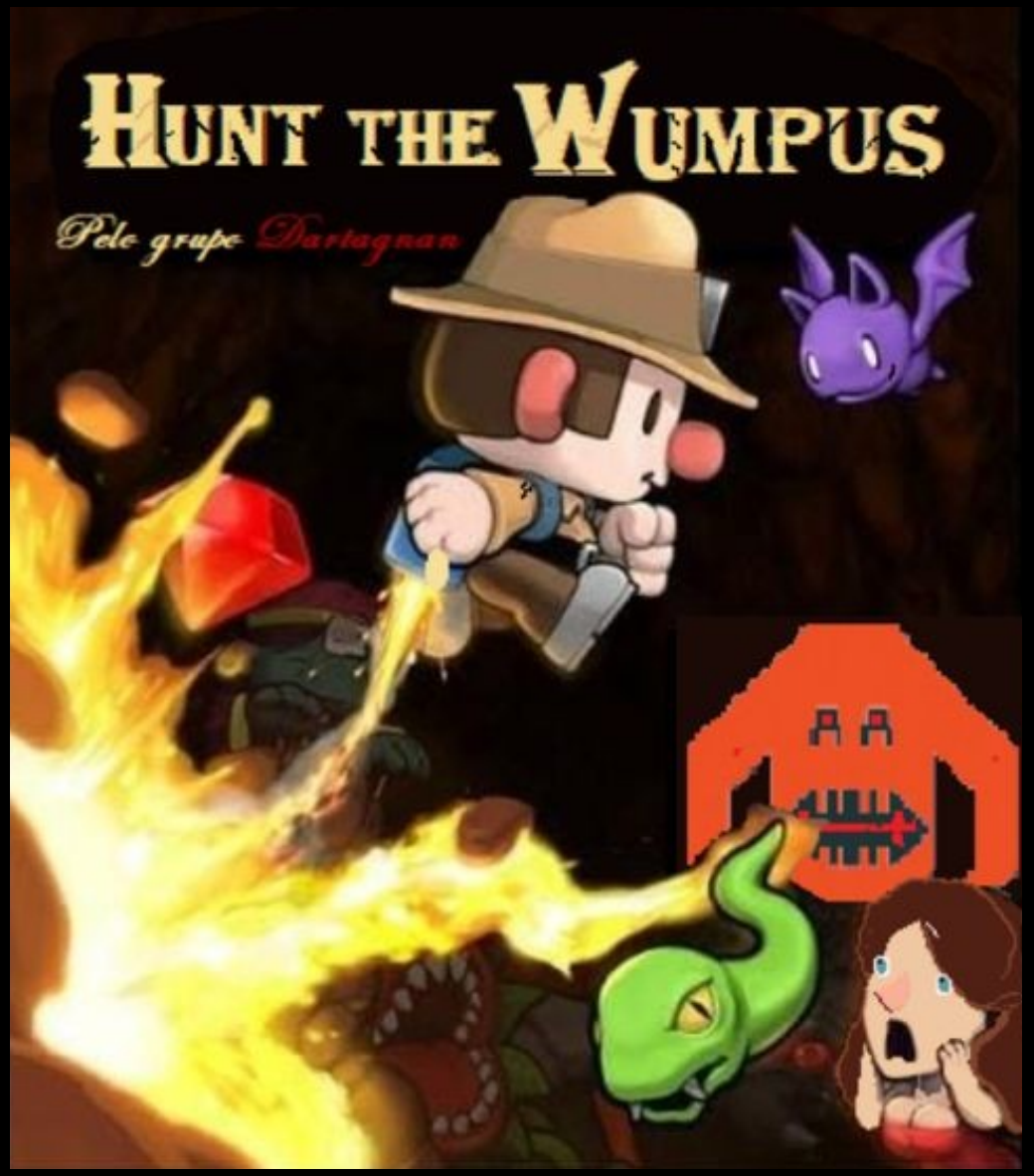


Setembro de 2021

POLI
ESCOLA POLITÉCNICA
DE PERNAMBUCO

UPE
UNIVERSIDADE
DE PERNAMBUCO

Identificação

Universidade de Pernambuco (UPE)
Escola Politécnica de Pernambuco (Poli)
Semestre: 2020.2
Disciplina: Lógica Matemática
Docente: Dr. Ruben Carlo Benante
Grupo 04 - Dartagnan

Identificação

- Integrantes do grupo 04 - Dartagnan

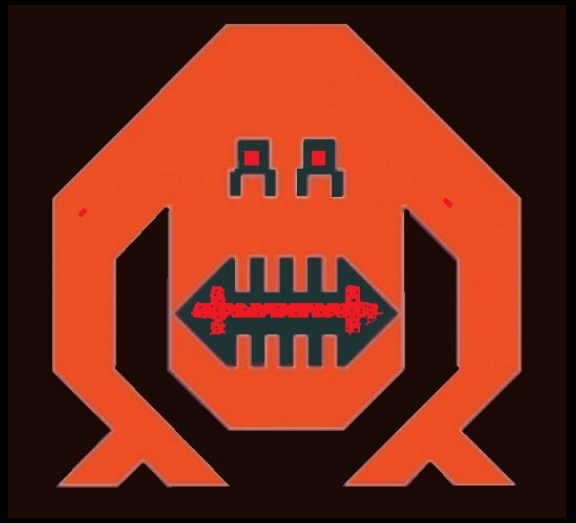
  - Vitor José Alves de Freitas - Lider do grupo -
  - Thiago de Azevedo Cavendish
  - Guilherme Rodrigues Chaves do Nascimento
  - Fernando Cordeiro Frutuoso de Souza Sobrinho
  - Antônio Carlos Lustoza Lemos Filho
  - Matheus Luiz Ferreira dos Santos
  - Kattine Costa Pedrosa
  - Felipe Fonseca Rabelo
  - Victor Schmitz Donato Cavalcanti
  - Rennan Reis Peinado

Introdução

- O Mundo Wumpus

  - Criado por Gregory Yob
  - Objetivo do jogo
  - Obstáculos e eventos
  - Percepções e ações

  - O Monstro:



```
 _____ _____ _____ _____
|     |     |     |  P  |
|_____|_____|_____|_____|
|W    |     G|  P  |     |
|_____|_____|_____|_____|
|     |     |     |     |
|_____|_____|_____|_____|
| A   |     |  P  |     |
|_____|_____|_____|_____|
```

Introdução

- Lista de fatos dinâmicos
  - Fatos que podem ser modificados

```
/*conjunto dinamico que declara o
predicado*/
:- dynamic([
  arrows_owned/1,
  objective/1,
  visited/1,
  pit_locations/1,
  nopit_locations/1,
  wumpus_locations/1,
  nowumpus_location/1,
  current_location/1,
  last_location/1,
  orientation/1,
  last_action/1,
  nactions/1, %numero de acoes.
  destination/1,
  route/1,
  wumpusdeead/0
]).
```

Introdução

- Inicialização:
  - Init_agent:

    Usado para :

    - Poder começar o programa
    - Adicionar e limpar memória
    - Adicionar os dinâmicos

Introdução

- Inicialização:

•Predicados pré-definidos utilizados:
•Assert –Insere novos fatos na base de conhecimento do programa;
•Retract –Possibilita a retirada de predicados e regras já existentes sem sua deleção;
•Cut –Estrutura que interrompe o backtrack quando declarada;
•Subtract –Retira uma lista a partir de outra, resultando numa terceira lista;
•Union –Faz a união de dois conjuntos, resultando num terceiro conjunto;
•Delete –Apaga um elemento de uma lista;
•Intersection –Realiza a interseccão de dois conjuntos, resultando num terceiro.

Introdução

- Inicialização:
  - Init_agent:
  - Declaração de predicados dinâmicos
  - Declara uma série de predicados como sendo dinâmicos, que podem ser alterados durante a execução do programa, informando o número de argumentos que são utilizados

Introdução

- Inicialização:

  Começo do programa:

  Comando ‘$swipl -s agent004.pl’ para iniciar a execução do Prolog

  O simulador do mundo é iniciado com ‘start.’ logo chamando o predicado init_agent

Predicados

- run_agent
- Predicado que, para cada turno, é chamado para realizar a
- ação do agente. A ação é um de seus parâmetros,
- assim como a lista de percepções.

- O primeiro predicado run_agent reage em relação a percepção com GPS, indicando atuais localização e orientação do agente, realiza modificações de memoria que utilizam as percepções com os predicados de final handler , avalia se a quantidade de ações está acima de 70, realiza uma ação com runactions e imprime certas informações do processamento.
- O segundo trata das modificações de memoria com o premapper , passando a não mais reagir ao GPS, e com alguns predicados idênticos ao primeiro.

Predicados

Agente com memória (“inteligência”):
O agente tomará suas decisões baseando-se no conjunto de casas.

O agente trabalha com três objetivos: explorar o mapa, sair e matar o wumpus.

Ele terá que visitar o maior número de casas possível, ser capaz de voltar à posição inicial para sair(se tiver pego o outro) e descoberto a posição e matado o wumpus

Foram criados vários predicados para utilizar o agente com memória

Predicados

- maybeleave

- Avalia se o número de ações do agente ultrapassa 70 ações (valor a partir do qual o agente corre um risco grande entrar em starvation visto que existe o limite de 100 ações) e define o objetivo leave

- Info

- Este predicado foi criado para melhor visualização do processamento, que imprime algumas informações relevantes para a compreensão das decisões tomadas pelo agente. As informações passadas são: Casas seguras, casas inseguras, casas visitadas, casa atual e orientação.

- printobj

- Predicado responsável por imprimir o objetivo atual do agente, quando não há um objetivo, o agente estará automaticamente no modo de explore

Predicados

• safe_locations e nosafe_locations

Estes predicados são utilizados para formar listas de localizações seguras e inseguras baseadas nas listas de posições onde há e não há perigo.

• premapper

Inserido no mapeamento, esse predicado é utilizado
para mapear as posições adjacentes ao agente
quando a última ação foi goforward e não há
percepção de bump realizando trocas de posição
na memória do agente a partir da última posição e
realiza um novo mapeamento com a localização e as percepções atuais.
O segundo predicado é para o caso de ocorrer
percepção de bump.
O terceiro é para a situação da ação passada não ser
goforward.

Predicados

- step ; getstep

Estes predicados servem para inserir a movimentação do agente na memória (sua posição e listas com casas adjacentes) quando ocorre goforward
Inicia se obtendo a posição atual (neste caso a anterior a ação), aplica se à ela a movimentação a partir da orientação (predicados getstep, realizando-se um assert com a nova posição à current_location um assert com a posição anterior à last_location sendo a posição atual adicionada à lista existente em visited

- pit_ handler e trysubp

Recebe a percepção da brisa e a partir disso adiciona as posições adjacentes à lista de posições com ou sem buraco.
O trysubp retira da lista de posições com buraco (pit_locations) as presentes na sem buraco (nopit_locations)

Predicados

- wumpus_handler e trysubw

Semelhante aos handler's de perigo, o wumpus_handler processa a percepção do fedor do wumpus e a partir disso adiciona as posições adjacentes à lista de posições com ou sem wumpus. Contudo, existe um predicado de maior prioridade que trata de considerar a morte do wumpus, que impede demais ações baseadas na percepção de fedor.

O trysubw retira da lista de posições com buraco (wumpus_locations), as presentes na sem buraco (nowumpus_locations)

Predicados

- memory_handler

Este predicado é utilizado para realizar as modificações relacionadas à ação tomada e suas relativas modificações na memória do agente para o próximo turno goforward é tratado pelo premapper pois exige percepção de no[não] para bump e contagem do número de ações (por isso a necessidade de fail nos demais predicados de maior prioridade)

Predicados

• maybekill
Predicado que muda o objetivo do agente para matar o Wumpus quando resta apenas uma possível posição para o mesmo e ele não está morto.

•changeorient
Reage a uma ação de turno e, a depender da orientação do agente, gira o assert de orientação em 90 º para um dos sentidos possíveis e seguros.

•runactions
Predicados responsáveis pela tomada de decisão do agente, são baseados na prioridade de ações com base em objetivos (primeiramente reativo para grab climb e gps, depois matar o wumpus e por fim, explorar)

Predicados

• killit

Predicados responsáveis pela tomada de decisão do agente quando o agente encontra-se configurado para o objetivo kill. Quando se encontra numa posição colinear ao wumpus o agente atira
ou muda de orientação, para ficar de frente para o monstro. Quando não se encontra nessa posição,
é iniciada a procura por uma rota para uma posição colinear ao wumpus

Agente reativo:
Algumas deduções são muito simples de serem feitas, por isso, o agente tem ações pré-definidas para certas percepções. O agente reage a percepção glitter realizando a ação grab
A casa [1, 1] é a inicial, porém após o início do programa, ela não será mais a única casa
visitada e o agente pode voltar para ela. Quando essas condições ocorrerem de forma
simultânea, o agente executará a ação climb.

Caso o agente bata na parede (bump) ele irá executar o GPS para saber seu
posicionamento.

Há uma reação para que o agente atire de acordo com o fedor do Wumpus.

Predicados

• adjacent
• Trata-se de casas adjacentes, é a análise dessas casas para fazer o mapeamento do formato do mundo. Logo, são consideradas adjacentes as casas que estão a esquerda, a direita, acima ou abaixo do agente e
que fazem parte do mapa.

• walker
O predicado walker é responsável por buscar um destino seguro, criar uma rota para este e realizar a ação baseada naquela rota. Em sua estrutura
também foram adicionados write para acompanhamento do seu processamento, para melhor compreensão da atitude do agente.

Predicados

• init_search / search
Um predicado inicia o outro e, atuando em conjunto, procuram uma casa não visitada e segura a partir da casa [1, 1] da esquerda para a direita e de baixo para cima. O init_search precisa do set_destination/1. Caso não seja encontrada uma posição segura, o objetivo será sair.

•set_destination
Seleciona uma casa destino. Caso o objetivo seja sair, o destino será
a casa [1, 1]. Se o número de ações for menor que 2 ele estabelece a casa [2, 1] como destino. Pode ser chamado para qualquer posição do mapa e, se executado, verifica se a posição em que foi chamado é uma casa segura não visitada . Satisfeita essa condição, a casa é estabelecida como destino.

Predicados

- first_step ; set_route

Há a criação de caminho para o destino do agente, de forma direta, comparando posições de saída. No first_step há a análise da primeira ação necessária para ir até o destino.

- first_step ; set_route

O cálculo da rota nova é feito pelo set_route escolhendo o caminho adjacente mais seguro e liberando um novo destino para o agente.

- closestadj

Predicados closestadj buscam uma casa adjacente para melhor caminho ao destino informado ou chamam outros predicados de seleção de casa quando não encontrada. Inicialmente verificam se a casa a ser pesquisada está fora dos limites do mundo, então buscam alguma casa adjacente capaz de prover um caminho mais direto possível ao destino. Caso não seja encontrada, são chamados os demais predicados de seleção de casa para a rota.

Predicados

- longpath

O predicado longpath busca uma casa adjacente na direção oposta ao mais perto, quando não são encontrados outros caminhos para o destino.

- diag (x/y)

Predicados diag (X/Y) buscam uma casa adjacente na direção de um destino diagonal (quando nenhuma das coordenadas deste destino são iguais a referência).

Predicados

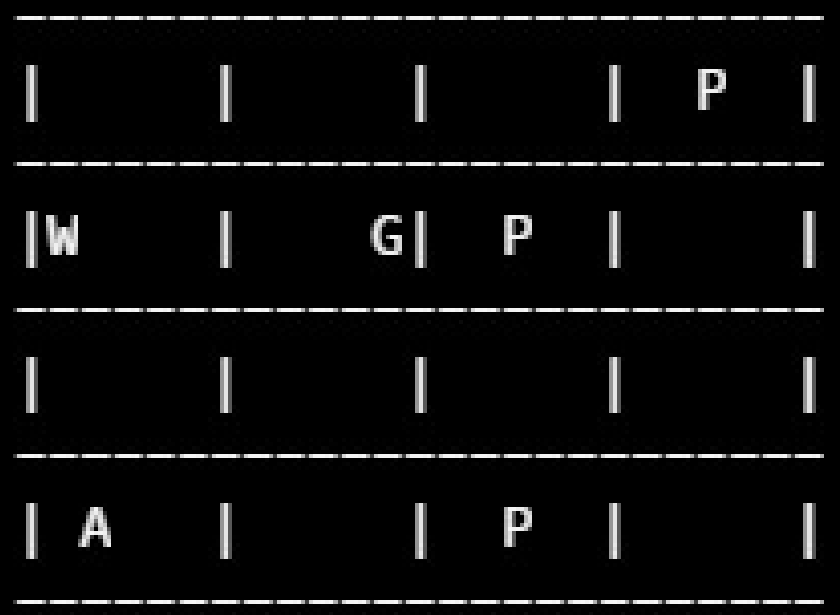
- tryperpendicular

O predicado tryperpendicular buscam uma casa perpendicular ao percurso do destino até a referência.

- aim_cell

Recebe duas posições e realizam a ação de
rotação quando não está na orientação correta para a segunda ou a ação de goforward quando não for o caso.

Conclusão



POLI ESCOLA POLITÉCNICA DE PERNAMBUCO

UPE UNIVERSIDADE DE PERNAMBUCO