

Universidade Federal de São Carlos

Inteligência Artificial:

Trabalho 1: Agente bombeiro capaz de utilizar extintores e apagar focos de incêndio

Alunos:

Leandro Novak, 586927





















Guilherme Neves, 586986

Gustavo Bastos, 551597

Professor:

Murilo Naldi

São Carlos / 2019

2. Conjunto de regras

Para resolver o problema do agente bombeiro, utilizamos as seguintes regras:

```
conteudo(X, Y, Objeto).
```

A regra acima é utilizada para definir os fatos que representam os objetos do ambiente e suas respectivas posições. X e Y seguem um sistema de coordenadas plano, com origem em (1,1), com X variando de 1 a 10 e Y de 1 a 5. O objeto pode assumir qualquer um dos seguintes valores:

- bombeiro;
- entulho;
- escada_inferior;
- escada_superior;
- extintor;
- incendio;
- parede.

```
% Habilita remoção das cláusulas conteudo e carga_extintor
:- dynamic conteudo/3.
:- dynamic carga_extintor/1.
```

Os predicados *dynamic* informam ao interpretador que os predicados *conteudo* e *carga_extintor* são dinâmicos. Ou seja, podem ser adicionados e removidos em tempo de execução.

```
% Métodos para manipulação de listas (Aula Prolog e exercícios)
pertence(Elem,[Elem|_]).
pertence(Elem,[_|Cauda]) :- pertence(Elem,Cauda).
```

Regra que verifica se um elemento pertence a uma lista, checando como caso base se ele é a cabeça da lista, e então verificando a cauda recursivamente.

```
concatena([ ],L,L).
concatena([Cab|Cauda],L2,[Cab|Resultado]) :- concatena(Cauda,L2,Resultado).
```

Regra que concatena duas listas, removendo os elementos da primeira lista e adicionando-os à segunda.

```
inverter([ ],L,L).
inverter([Cab|Cauda],L2,Aux) :- inverter(Cauda,L2,[Cab|Aux]).
```

Regra que inverte uma lista, retirando a cabeça de uma lista e adicionando-a à cabeça da nova lista.

```
% Regras para movimentação do bombeiro
% Sobe escada
permitido(X,Y,cima) :- conteudo(X,Y,escada_inferior), Y2 is Y+1,
    conteudo(X,Y2,escada_superior).
```

Regra que verifica se é possível subir uma escada, checando se a posição atual e a mesma posição no eixo X acima possuem escada inferior e superior, respectivamente. Caso positivo, a posição no eixo Y é incrementada.

```
% Desce escada
permitido(X,Y,baixo) :- conteudo(X,Y,escada_superior), Y2 is Y-1,
    conteudo(X,Y2,escada_inferior).
```

Regra que verifica se é possível descer uma escada, checando se a posição atual e a mesma posição no eixo X abaixo possuem escada superior e inferior, respectivamente. Caso positivo, a posição no eixo Y é decrementada.

```
% Verifica parede ou incêndio
permitido(X,Y,direita) :- X1 is X+1, not(conteudo(X1,Y,entulho)),
    not(conteudo(X1,Y,parede)), not(conteudo(X1,Y,incendio)), X < 10.
permitido(X,Y,esquerda) :- X1 is X-1, not(conteudo(X1,Y,entulho)),
    not(conteudo(X1,Y,parede)), not(conteudo(X1,Y,incendio)), X > 1.
```

Regras que verificam se é possível se movimentar para a direita ou para a esquerda, checando se há algum obstáculo, como parede, entulho ou incêndio, nas coordenadas $X+1$ caso direita, ou $X-1$ caso esquerda, além de checar se o movimento não ultrapassará os limites laterais estabelecidos.

```
% Anda sobre entulho
permitido(X,Y,direita) :- X1 is X+1, conteudo(X1,Y,entulho),
    not(conteudo(X,Y,_)), X2 is X+2, not(conteudo(X2,Y,_)).
permitido(X,Y,esquerda) :- X1 is X-1, conteudo(X1,Y,entulho),
    not(conteudo(X,Y,_)), X2 is X-2, not(conteudo(X2,Y,_)).
```

Regras que verificam se é possível se movimentar sobre o entulho, à direita ou à esquerda, verificando se a posição seguinte é um entulho e se ambas as posições adjacentes (X e $X+2/X-2$) não possuem nenhum conteúdo.

```
% Passa por incêndio se possuir carga no extintor
permitido(X,Y,direita) :- X1 is X+1, conteudo(X1,Y,incendio),
    carga_extintor(Carga), Carga > 0.
permitido(X,Y,esquerda) :- X1 is X-1, conteudo(X1,Y,incendio),
    carga_extintor(Carga), Carga > 0.
```

Regras que verificam se é possível se movimentar sobre um incêndio, verificando se o elemento à direita/esquerda é um incêndio e se há carga no extintor.

```
% Gera os sucessores as movimentações (cima, baixo, direita e esquerda)
s([X,Y],[X,Y2]) :- permitido(X,Y,cima), Y2 is Y+1.
s([X,Y],[X,Y2]) :- permitido(X,Y,baixo), Y2 is Y-1.
s([X,Y],[X2,Y]) :- permitido(X,Y,direita), X2 is X+1.
s([X,Y],[X2,Y]) :- permitido(X,Y,esquerda), X2 is X-1.
```

Regras de estado sucessor que utilizam as regras “permitido” anteriormente apresentadas para checar a possibilidade de um movimento e efetuam o movimento para o lado correspondente.

```
% Adaptacao do código de Busca em Largura (bl) disponibilizado no material de
% resolução de problemas por meio de busca. Alterou-se para permitir a busca
% para mais de um tipo de item (incendio, extintor).
solucao_bl(Inicial,Item,Solucao) :- bl([[Inicial]],Solucao,Item).

% Se o primeiro estado de F for meta, então o retorna com o caminho
bl([[Estado|Caminho]|_],[Estado|Caminho],Item) :- meta(Estado,Item).

% Falha ao encontrar a meta, então estende o primeiro estado até seus sucessores
% e os coloca no final da lista de fronteira
bl([Primeiro|Outros], Solucao, Item) :-
    estende(Primeiro,Sucessores),
    concatena(Outros,Sucessores,NovaFronteira),
    bl(NovaFronteira,Solucao, Item).

% Metodo que faz a extensao do caminho até os nós filhos do estado
estende([Estado|Caminho],ListaSucessores):-
    bagof([Sucessor,Estado|Caminho], (s(Estado,Sucessor),
    not(pertence(Sucessor,[Estado|Caminho]))),
    ListaSucessores),!.

% Se o estado não tiver sucessor, falha e não procura mais (corte)
estende(_ ,[]).
```

Regras utilizadas para efetuar as buscas em largura e posteriormente listar o caminho percorrido pelo agente

```
% Define o estado meta, se a posição atual contém o item procurado
meta(Estado,Item) :-
    bagof([X,Y],conteudo(X,Y,Item),Lista), pertenece(Estado,Lista).
```

A regra é auxiliar do processo de busca verificando se a meta da busca foi atingida. Tal regra faz uso da variável “Estado” (posição no ambiente) para verificar se o estado está presente na lista de estados que possuem o conteúdo definido na variável “Item”.

```

% Extintor ainda possui carga, então não efetua a busca
busca_extintor([X,Y,Caminho],[X,Y,Caminho]) :-
    carga_extintor(Carga),
    Carga > 0,!.

% Extintor vazio, inicia busca por um novo extintor
busca_extintor([X,Y,Caminho],[X2,Y2,[X2,Y2|Caminho2]]) :-
    solucao_bl([X,Y],extintor,C),
    concatena(C,Caminho,[X2,Y2|Caminho2]),
    atualiza_extintor(2),
    retract(conteudo(X2,Y2,extintor)),!.

% Atualiza a carga do extintor
atualiza_extintor(Carga) :-
    retractall(carga_extintor(_)),
    assert(carga_extintor(Carga)).

```

As regras acima são responsáveis por fazer a busca pelo extintor. A primeira verifica se o bombeiro ainda possui um extintor com carga e desiste da busca em caso de resultado positivo.

A segunda regra utiliza a busca em largura para procurar por um extintor e caso encontre faz uso da regra auxiliar *atualiza_extintor* para atualizar a carga do extintor para o valor total (dois) e remove o extintor encontrado da base de dados.

```

% Sem incêndios para apagar
busca_incendio([X,Y,Caminho],[X,Y,Caminho]) :-
    aggregate_all(count, conteudo(_,_,incendio), Count),
    Count == 0,!.

% Tenta encontrar um incêndio e apagá-lo
busca_incendio([X,Y,Caminho],[X2,Y2,[X2,Y2|Caminho2]]) :-
    solucao_bl([X,Y],incendio,C),
    carga_extintor(Carga),
    NovaCarga is Carga-1,
    atualiza_extintor(NovaCarga),
    concatena(C,Caminho,[X2,Y2|Caminho2]),
    retract(conteudo(X2,Y2,incendio)),!.

```

As regras *busca_incendio* atuam de forma semelhante à *busca_extintor*. A primeira verifica se ainda existem incêndios a serem apagados por meio do comando *aggregate_all* e interrompe a execução caso nenhum incêndio for encontrado.

A segunda regra utiliza a busca em largura para encontrar um incêndio, decrementa em 1 (um) a carga do extintor e remove o incêndio apagado da base de dados.


```

% Sem incêndios para apagar
apaga_incendios([X,Y,Caminho], [X,Y,Caminho]) :-
    aggregate_all(count, conteudo(_,_,incendio), Count),
    Count == 0,!.

% Busca extintor e apaga até dois incêndios
apaga_incendios([X,Y,Caminho], [XF,YF,CaminhoF]) :-
    busca_extintor([X,Y,Caminho], [X2,Y2,Caminho2]),
    carga_extintor(Carga), Carga > 0,
    busca_incendio([X2,Y2,Caminho2], [X3,Y3,Caminho3]),
    busca_incendio([X3,Y3,Caminho3], [X4,Y4,Caminho4]),
    apaga_incendios([X4,Y4,Caminho4], [XF,YF,CaminhoF]).

```

Regras recursivas responsáveis por verificar se todos os incêndios foram apagados, buscar extintor e apagar incêndios. A primeira faz uso do comando *aggregate_all* para contar o número de incêndios ainda definidos na base de dados. Já a segunda busca por um extintor, certificando-se de que existe carga e apaga até dois incêndios utilizando a carga do extintor. Em seguida, utiliza recursão para repetir o procedimento.

```

apaga_todos_os_incendios(Arquivo, Caminho) :-
    carrega_ambiente(Arquivo),
    conteudo(X,Y,bombeiro),
    retract(conteudo(X,Y,bombeiro)),
    apaga_incendios([X,Y,[]],[_,_ ,CaminhoInv]),
    inverter(CaminhoInv,Caminho,[]).

```

Esta é a principal regra do programa, é a partir dela que se inicia a busca e são chamadas todas as outras regras.

```

% Carrega o ambiente de um arquivo externo
% base: https://www.swi-prolog.org/FAQ/ReadDynamicFromFile.html
carrega_ambiente(File) :-
    retractall(conteudo(_,_,_)),
    retractall(carga_extintor(_)),
    assert(carga_extintor(0)),
    set_prolog_flag(answer_write_options,[max_depth(0)]),
    open(File, read, Stream),
    call_cleanup(carrega_ambiente(Stream,_,_),
    close(Stream)).

carrega_ambiente(_, [], T) :- T == end_of_file, !.

carrega_ambiente(Stream, [T|X], _) :-
    read(Stream, T),
    assert(T),
    carrega_ambiente(Stream,X,T).

```

Regras auxiliares que permitem a separação da lógica de resolução do problema, da definição dos fatos que representam o ambiente.

A primeira regra utiliza o comando interno *retractall* para remover fatos da base e o comando *assert* para inserir um novo fato. Além disso, o comando *set_prolog_flag* é utilizada para permitir a exibição de resultados longos. O restante da primeira regra, e as seguintes percorrem o arquivo declarando na base de dados os fatos e regras presentes no arquivo.

3. Execução

A execução do projeto se dá por meio da ferramenta `swipl`, conforme mostrado na figura a seguir:

```
novak:src$ swipl firefighter.pl
Welcome to SWI-Prolog (threaded, 64 bits, version 8.0.2)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.
```

```
For online help and background, visit http://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).
```

```
?- █
```

Em seguida faz-se o uso da regra *apaga_todos_os_incendios* para resolver os ambientes, tendo como entrada o nome do arquivo que contém o ambiente (no exemplo, “ambiente1.pl”) envolto em aspas simples e a variável que receberá o resultado da execução (no exemplo, Resultado):

```
?- apaga_todos_os_incendios('ambiente1.pl', Resultado).
Resultado = [[1,1],[2,1],[3,1],[4,1],[5,1],[5,2],[6,2],[7,2],[8,2],[9,2],
,[9,3],[10,3],[10,4],[9,4],[8,4],[7,4],[6,4],[5,4],[5,5],[4,5],[3,5],[3,
4],[2,4],[1,4],[1,3],[2,3],[2,3],[1,3],[1,4],[2,4],[3,4],[3,5],[4,5],[5,
5],[5,4],[6,4],[7,4],[8,4],[9,4],[9,5],[10,5],[10,5],[9,5],[9,4],[10,4],
[10,3],[9,3],[9,2],[8,2],[7,2],[6,2],[5,2],[5,1],[6,1],[7,1],[8,1],[9,1]
].
```

Em caso de falha por não ser possível solucionar o ambiente, caso do ambiente 2, a consulta retornará falso:

```
?- apaga_todos_os_incendios('ambiente2.pl', Resultado).
false.
```


4. Códigos

Abaixo seguem os códigos para cada um dos ambientes disponibilizados para teste e o código capaz de resolver o problema.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Ambiente Completo 1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% dynamic permite remover conhecimento da base
:- dynamic conteudo/3.

% Define a posição inicial do bombeiro
conteudo(1,1,bombeiro).

% Define os objetos e sua posições
conteudo(5,1,escada_inferior).
conteudo(9,1,incendio).

conteudo(4,2,entulho).
conteudo(5,2,escada_superior).
conteudo(7,2,entulho).
conteudo(9,2,escada_inferior).

conteudo(1,3,escada_inferior).
conteudo(2,3,extintor).
conteudo(3,3,parede).
conteudo(9,3,escada_superior).
conteudo(10,3,escada_inferior).

conteudo(1,4,escada_superior).
conteudo(3,4,escada_inferior).
conteudo(4,4,entulho).
conteudo(5,4,escada_inferior).
conteudo(7,4,entulho).
conteudo(9,4,escada_inferior).
conteudo(10,4,escada_superior).
```

```

conteudo(3,5,escada_superior).
conteudo(5,5,escada_superior).
conteudo(6,5,entulho).
conteudo(7,5,parede).
conteudo(9,5,escada_superior).
conteudo(10,5,incendio).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%
% Ambiente Completo 2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%

% dynamic permite remover conhecimento da base
:- dynamic conteudo/3.

% Define a posição inicial do bombeiro
conteudo(1,1,bombeiro).

% Define os objetos e sua posições
conteudo(3,1,entulho).
conteudo(5,1,escada_inferior).
conteudo(8,1,escada_inferior).
conteudo(9,1,incendio).
conteudo(10,1,incendio).

conteudo(1,2,escada_inferior).
conteudo(5,2,escada_superior).
conteudo(6,2,entulho).
conteudo(7,2,entulho).
conteudo(8,2,escada_superior).

conteudo(1,3,escada_superior).
conteudo(2,3,extintor).
conteudo(3,3,extintor).
conteudo(4,3,parede).
conteudo(10,3,escada_inferior).

conteudo(3,4,escada_inferior).

```

```
conteudo(4,4,entulho).
conteudo(5,4,escada_inferior).
conteudo(7,4,entulho).
conteudo(9,4,escada_inferior).
conteudo(10,4,escada_superior).
```

```
conteudo(2,5,incendio).
conteudo(3,5,escada_superior).
conteudo(5,5,escada_superior).
conteudo(6,5,entulho).
conteudo(7,5,parede).
conteudo(9,5,escada_superior).
conteudo(10,5,incendio).
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%
```

```
% Ambiente Completo 3
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%
```

```
% dynamic permite remover conhecimento da base
:- dynamic conteudo/3.
```

```
% Define a posição inicial do bombeiro
conteudo(1,1,bombeiro).
```

```
% Define os objetos e sua posições
```

```
conteudo(3,1,entulho).
conteudo(5,1,escada_inferior).
conteudo(6,1,parede).
conteudo(9,1,escada_inferior).
conteudo(10,1,incendio).
```

```
conteudo(1,2,escada_inferior).
conteudo(3,2,entulho).
conteudo(5,2,escada_superior).
conteudo(6,2,entulho).
conteudo(7,2,entulho).
conteudo(8,2,escada_inferior).
conteudo(9,2,escada_superior).
```

```
conteudo(1,3,escada_superior).
conteudo(4,3,escada_inferior).
conteudo(5,3,parede).
conteudo(6,3,extintor).
conteudo(8,3,escada_superior).
conteudo(10,3,escada_inferior).
```

```
conteudo(1,4,extintor).
conteudo(3,4,escada_inferior).
conteudo(4,4,escada_superior).
conteudo(6,4,entulho).
conteudo(9,4,escada_inferior).
conteudo(10,4,escada_superior).
```

```
conteudo(2,5,incendio).
conteudo(3,5,escada_superior).
conteudo(6,5,entulho).
conteudo(8,5,incendio).
conteudo(9,5,escada_superior).
conteudo(10,5,incendio).
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%
```

```
% Ambiente Completo 4
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%
```

```
% dynamic permite remover conhecimento da base
:- dynamic conteudo/3.
```

```
% Define a posição inicial do bombeiro
conteudo(1,1,bombeiro).
```

```
% Define os objetos e sua posições
```

```
conteudo(2,1,entulho).
conteudo(4,1,escada_inferior).
conteudo(5,1,incendio).
conteudo(6,1,parede).
conteudo(7,1,extintor).
```

```
conteudo(9,1,escada_inferior).
```

```
conteudo(1,2,escada_inferior).
```

```
conteudo(3,2,entulho).
```

```
conteudo(4,2,escada_superior).
```

```
conteudo(7,2,escada_inferior).
```

```
conteudo(9,2,escada_superior).
```

```
conteudo(10,2,entulho).
```

```
conteudo(1,3,escada_superior).
```

```
conteudo(2,3,incendio).
```

```
conteudo(4,3,escada_inferior).
```

```
conteudo(5,3,parede).
```

```
conteudo(6,3,extintor).
```

```
conteudo(7,3,escada_superior).
```

```
conteudo(10,3,escada_inferior).
```

```
conteudo(3,4,escada_inferior).
```

```
conteudo(4,4,escada_superior).
```

```
conteudo(6,4,entulho).
```

```
conteudo(9,4,escada_inferior).
```

```
conteudo(10,4,escada_superior).
```

```
conteudo(2,5,entulho).
```

```
conteudo(3,5,escada_superior).
```

```
conteudo(6,5,entulho).
```

```
conteudo(8,5,incendio).
```

```
conteudo(9,5,escada_superior).
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%
```

```
% Firefighter Prolog
```

```
% Inteligência Artificial - Turma A - 2019/2
```

```
%
```

```
% Professor:
```

```
%   - Murilo Naldi
```

```
% Alunos:
```

```
%   - Leandro Novak           586927
```

```
%   - Guilherme Neves       586986
```



```

% - Gustavo Bastos          551597
%
% Uso: apaga_todos_os_incendios('ambientex.pl', Caminho).
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Habilita remoção das cláusulas conteudo e carga_extintor
:- dynamic conteudo/3.
:- dynamic carga_extintor/1.

% Métodos para manipulação de listas (Aula Prolog e exercícios)
pertence(Elem,[Elem|_]).
pertence(Elem,[_|Cauda]) :- pertence(Elem,Cauda).

concatena([],L,L).
concatena([Cab|Cauda],L2,[Cab|Resultado]) :-
concatena(Cauda,L2,Resultado).

inverter([],L,L).
inverter([Cab|Cauda],L2,Aux) :- inverter(Cauda,L2,[Cab|Aux]).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Regras para movimentação do bombeiro
% Sobe escada
permitido(X,Y,cima) :- conteudo(X,Y,escada_inferior), Y2 is Y+1,
    conteudo(X,Y2,escada_superior).

% Desce escada
permitido(X,Y,baixo) :- conteudo(X,Y,escada_superior), Y2 is Y-1,
    conteudo(X,Y2,escada_inferior).

% Verifica parede ou incêndio
permitido(X,Y,direita) :- X1 is X+1, not(conteudo(X1,Y,entulho)),
    not(conteudo(X1,Y,parede)), not(conteudo(X1,Y,incendio)), X < 10.
permitido(X,Y,esquerda) :- X1 is X-1, not(conteudo(X1,Y,entulho)),
    not(conteudo(X1,Y,parede)), not(conteudo(X1,Y,incendio)), X > 1.

% Anda sobre entulho
permitido(X,Y,direita) :- X1 is X+1, conteudo(X1,Y,entulho),

```

```

    not(conteudo(X,Y,_)), X2 is X+2, not(conteudo(X2,Y,_)).
permitido(X,Y,esquerda) :- X1 is X-1, conteudo(X1,Y,entulho),
    not(conteudo(X,Y,_)), X2 is X-2, not(conteudo(X2,Y,_)).

% Passa por incêndio se possuir carga no extintor
permitido(X,Y,direita) :- X1 is X+1, conteudo(X1,Y,incendio),
    carga_extintor(Carga), Carga > 0.
permitido(X,Y,esquerda) :- X1 is X-1, conteudo(X1,Y,incendio),
    carga_extintor(Carga), Carga > 0.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Gera os sucessores as movimentações (cima, baixo, direita e esquerda)
s([X,Y],[X,Y2]) :- permitido(X,Y,cima), Y2 is Y+1.
s([X,Y],[X,Y2]) :- permitido(X,Y,baixo), Y2 is Y-1.
s([X,Y],[X2,Y]) :- permitido(X,Y,direita), X2 is X+1.
s([X,Y],[X2,Y]) :- permitido(X,Y,esquerda), X2 is X-1.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Adaptacao do código de Busca em Largura (bl) disponibilizado no
material de
% resolução de problemas por meio de busca. Alterou-se para permitir a
busca
% para mais de um tipo de item (incendio, extintor).
solucao_bl(Inicial,Item,Solucao) :- bl([[Inicial]],Solucao,Item).

% Se o primeiro estado de F for meta, então o retorna com o caminho
bl([[Estado|Caminho]|_],[Estado|Caminho],Item) :- meta(Estado,Item).

% Falha ao encontrar a meta, então estende o primeiro estado até seus
sucessores
% e os coloca no final da lista de fronteira
bl([Primeiro|Outros], Solucao, Item) :-
    estende(Primeiro,Sucessores),
    concatena(Outros,Sucessores,NovaFronteira),
    bl(NovaFronteira,Solucao, Item).

% Metodo que faz a extensao do caminho até os nós filhos do estado
estende([Estado|Caminho],ListaSucessores):-

```

```

    bagof([Sucessor,Estado|Caminho], (s(Estado,Sucessor),
    not(pertence(Sucessor,[Estado|Caminho]))),
    ListaSucessores),!.

% Se o estado não tiver sucessor, falha e não procura mais (corte)
estende( _ ,[]).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Define o estado meta, se a posição atual contém o item procurado
meta(Estado,Item) :-
    bagof([X,Y],conteudo(X,Y,Item),Lista), pertenece(Estado,Lista).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Extintor ainda possui carga, então não efetua a busca
busca_extintor([X,Y,Caminho],[X,Y,Caminho]) :-
    carga_extintor(Carga),
    Carga > 0,!.

% Extintor vazio, inicia busca por um novo extintor
busca_extintor([X,Y,Caminho],[X2,Y2,[X2,Y2|Caminho2]]) :-
    solucao_bl([X,Y],extintor,C),
    concatena(C,Caminho,[X2,Y2|Caminho2]),
    atualiza_extintor(2),
    retract(conteudo(X2,Y2,extintor)),!.

% Atualiza a carga do extintor
atualiza_extintor(Carga) :-
    retractall(carga_extintor(_)),
    assert(carga_extintor(Carga)).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Sem incêndios para apagar
busca_incendio([X,Y,Caminho],[X,Y,Caminho]) :-
    aggregate_all(count, conteudo(_,_,incendio), Count),
    Count == 0,!.

% Tenta encontrar um incêndio e apagá-lo

```

```

busca_incendio([X,Y,Caminho], [X2,Y2,[X2,Y2|Caminho2]]) :-
    solucao_bl([X,Y],incendio,C),
    carga_extintor(Carga),
    NovaCarga is Carga-1,
    atualiza_extintor(NovaCarga),
    concatena(C,Caminho,[X2,Y2|Caminho2]),
    retract(conteudo(X2,Y2,incendio)),!.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%
% Sem incêndios para apagar
apaga_incendios([X,Y,Caminho], [X,Y,Caminho]) :-
    aggregate_all(count, conteudo(_,_,incendio), Count),
    Count == 0,!.

% Busca extintor e apaga até dois incêndios
apaga_incendios([X,Y,Caminho], [XF,YF,CaminhoF]) :-
    busca_extintor([X,Y,Caminho],[X2,Y2,Caminho2]),
    carga_extintor(Carga), Carga > 0,
    busca_incendio([X2,Y2,Caminho2],[X3,Y3,Caminho3]),
    busca_incendio([X3,Y3,Caminho3],[X4,Y4,Caminho4]),
    apaga_incendios([X4,Y4,Caminho4],[XF,YF,CaminhoF]).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%
apaga_todos_os_incendios(Arquivo, Caminho) :-
    carrega_ambiente(Arquivo),
    conteudo(X,Y,bombeiro),
    retract(conteudo(X,Y,bombeiro)),
    apaga_incendios([X,Y,[]],[_,_,CaminhoInv]),
    inverter(CaminhoInv,Caminho,[]).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%
% Carrega o ambiente de um arquivo externo
% base: https://www.swi-prolog.org/FAQ/ReadDynamicFromFile.html
carrega_ambiente(File) :-
    retractall(conteudo(_,_,_)),
    retractall(carga_extintor(_)),
    assert(carga_extintor(0)),

```

```
set_prolog_flag(answer_write_options,[max_depth(0)]),
open(File, read, Stream),
call_cleanup(carrega_ambiente(Stream, _, _),
close(Stream)).

carrega_ambiente(_, [], T) :- T == end_of_file, !.

carrega_ambiente(Stream, [T|X], _) :-
    read(Stream, T),
    assert(T),
    carrega_ambiente(Stream,X,T).
```


5. Referências Bibliográficas

Trabalho 1 - Agente Bombeiro. Material disponibilizado pelo professor Dr. Murilo Naldi.

Aula 5 – Prolog (Parte 2). Material disponibilizado pelo professor Dr. Murilo Naldi.

Aula 06 – Resolução de problemas. Material disponibilizado pelo professor Dr. Murilo Naldi.

Prolog count the number of times a predicate is true. Disponível em:

<<https://stackoverflow.com/questions/6060268/prolog-count-the-number-of-times-a-predicate-is-true>>. Acesso em: 20 Out. 2019.

Reading dynamic data from a file. Disponível em:

<<https://www.swi-prolog.org/FAQ/ReadDynamicFromFile.html>>. Acesso em 20 Out. 2019.

SWI-Prolog - show long list. Disponível em:

<<https://stackoverflow.com/questions/8231762/swi-prolog-show-long-list>>. Acesso em 21 Out. 2019.

Managing (dynamic) predicates. Disponível em:

<<https://www.swi-prolog.org/pldoc/man?section=dynpreds>>. Acesso em 21 Out. 2019.