Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación
Lenguajes de Programación (6204)
Semestre 1-2024

Proyecto #2 – Prolog
Supervivencia en The Walking Dead

En este proyecto, te encontrarás en el mundo post-apocalíptico de The Walking Dead. Tu objetivo es navegar a través de un laberinto lleno de zombies, buscando suministros y encontrando a otros supervivientes. El proyecto se implementará en Prolog, utilizando su poderoso sistema de backtracking para simular la lógica del juego.

Parte 1: Representar el laberinto

Para representar un laberinto de forma dinámica en Prolog, puedes utilizar hechos y reglas que se puedan modificar en tiempo de ejecución. Se desea implementar el predicado `generarLaberinto` que permita representar las conexiones del laberinto, zombies, suministros y supervivientes.

```
generar_laberinto(Conexiones, Zombies, Suministros, Supervivientes).
```

Donde:

- Conexiones es una lista de listas que indican como está conectado el laberinto. Ejemplo: [[a, b], [b, c], [c, d], [d, e]].
- Zombies es una lista de posiciones en donde se encuentran los n zombies. Ejemplo: [c,e].
- Suministros es una lista de listas que posee el nombre del suministro y la posición en el laberinto. Ejemplo: [[arma, b), [comida, a]].
- Supervivientes es una lista de listas que indican el nombre de cada superviviente y su posición en el laberinto. Ejemplo [[rick, d]].

Con los ejemplos anteriores se espera que la base de conocimientos quede de la siguiente manera:

```
% Conexiones entre ubicaciones

conectado(a, b).

conectado(b, c).

conectado(c, d).

conectado(d, e).


% Ubicaciones de los zombies

zombie(c).

zombie(e).
```

```
% Ubicaciones de los suministros
suministro(arma, b).
suministro(comida, a).

% Ubicaciones de los supervivientes
superviviente(rick, d).
```

Ejemplos:

```
1) ?- generar_laberinto([[a, b], [b, c], [c, d], [d, e]], [c, e], [[comida, d]], [[rick, d], [daryl, b]]).
```

false.

```
2) ?- generar_laberinto([[a, b], [b, c], [c, d], [d, e]], [e], [[comida, a], [arma,  c]], [[carol, d], [negan, b]]).
```

true.

```
3) ?- generar_laberinto([[a, b], [b, c], [c, d], [d, e]], [c], [[arma,  a]], [[gleen, b]]).
```

true.

```
4) ?- generar_laberinto([[a, b], [b, c], [c, d], [d, e]], [a, e], [[medicina, c], [agua, a]], [[maggie, d], [michonne, b]]).
```

false.

Notas:

1. Se debe validar que cada lista de entrada no esté vacía.
2. Verificar que en cada posición solo pueda existir un zombie, un suministro o un superviviente.
3. Si los datos de entrada son correctos se debe retornar **true** como respuesta, sino se debe eliminar toda la base de conocimientos y retornar **false**.

Parte 2: Encontrar un camino seguro

Implementa un predicado que encuentra un camino a través del laberinto que evita a los zombies, recibiendo como entrada el punto de partida y llegada.

```
camino_seguro(Inicio, Fin, Camino).
```

Ejemplos:

1) Tomando en consideración la base de conocimientos generada por el **ejemplo 2** de la parte anterior:

```
?- camino_seguro(a, d, Camino).
```

```
Camino = [a, b, c, d].
```

2) Tomando en consideración la base de conocimientos generada por el **ejemplo 3** de la parte anterior:

```
?- camino_seguro(a, d, Camino).
```

false.

Parte 3: Encontrar camino con suministros

Se necesita implementar un predicado que permita hallar un camino desde un punto de inicio hasta un superviviente, recogiendo todos los suministros y evitando los zombies.

```
camino_con_suministros(Inicio, NombreSuperviviente, Camino).
```

Ejemplos:

Tomando en consideración la base de conocimientos arrojada por

```
?- generar_laberinto([[a, b], [b, c], [c, d], [d, e], [e, f], [f, g], [b, h], [h, i], [i, j], [j, k], [k, l], [l, m], [m, n], [n, g]], [c, e], [[comida, b], [agua, h], [medicina, m]], [[rick, g]]).
```

1)
```
?- camino_con_suministros(a, rick, Camino).
```

```
Camino = [a, b, h, i, j, k, l, m, n, g].
```

2)
```
?- camino_con_suministros(b, rick, Camino).
```

```
Camino = [b, h, i, j, k, l, m, n, g].
```

3)
```
?- camino_con_suministros(d, rick, Camino).
```

false.

4)
```
?- camino_con_suministros(a, daryl, Camino).
```

false.

Consideraciones para la entrega

- Debe utilizar SWI-Prolog.
- Los predicados definidos deben estar dentro de un archivo llamado **`Proyecto2.pl`**. Puede definir los predicados auxiliares con cualquier nombre que describa su propósito, pero los predicados descritos arriba deben mantener su nombre.
- Documente la solución (representación, hechos definidos, estrategia utilizada, etc.) en un archivo en formato *markdown* (**`README.md`**). No están permitidos documentos en ningún otro formato.
- No está permitido utilizar soluciones de terceros (incluyendo soluciones generadas por LLMs). Cualquier material consultado para ideas de solución o documentación debe ser referenciado en una sección del **`README.md`**, indicando para qué sirvió.
- El proyecto se puede realizar en grupos de **hasta dos estudiantes** y la fecha de entrega será el **martes 25/06/2024**, hasta las 11:59 pm (VET).
- El proyecto se debe adjuntar por e-mail a la dirección `ldpucv@gmail.com` usando como asunto:
  `“[Proyecto Prolog]” <DatosEstudiante> [; <DatosEstudiante>]`
  donde `<DatosEstudiante> ::= <Apellidos><Nombres>,<CI>`.

José Yvimas, junio 2024