Lenguajes de Programación

Solucionando problemas con el paradigma lógico

(Proyecto Segunda Parte)

Ramos López Lizbeth 201749275 lizbeth.ramosl@alumno.buap.mx

Tepoz Romero Belen 201770060 belen.tepoz@alumno.buap.mx

Vargas Arenas Pedro 201734553 pedro.vargasa@alumno.buap.mx

Facultad de ciencias de la computación

15 de Mayo 2021

Problemática a resolver

En este proyecto se plantea la solución para el problema del granjero, el cual está especificado a continuación.

“Hace mucho tiempo un granjero fue al mercado y compró un lobo, una cabra y una col. Para volver a su casa tenía que cruzar un río. El granjero dispone de una barca para cruzar a la otra orilla, pero en la barca solo caben él y una de sus compras. Si el lobo se queda solo con la cabra se la come, si la cabra se queda sola con la col se la come. El reto del granjero era cruzar él mismo y dejar sus compras a la otra orilla del río, dejando cada compra intacta.”

Este problema es un popular juego de lógica y forma parte de los denominados “puzles de cruzar el río”, existen variaciones del acertijo en cuanto a los objetos del mismo, como un zorro, una gallina y unas semillas, pero en todos los casos las reglas son iguales.

Justificación del uso del paradigma declarativo para solucionar el problema

El paradigma declarativo, también conocido como paradigma de programación lógica está basado en implementar relaciones antes que correspondencias, esto hace que la programación lógica a diferencia de la funcional sea potencialmente de un más alto nivel.

Las herramientas declarativas permiten muchas veces un más alto grado de reutilización y de abstracción en tareas repetitivas, permite organizar y simplificar la construcción de un sistema complejo.

En el problema al tratarse de acciones repetitivas, como cruzar a la izquierda, cruzar a la derecha y además contar con ciertas restricciones se adapta completamente a un programa construido de forma declarativa. Al hacerlo de esta forma podemos hacer la separación entre la descripción del problema y las estrategias para encontrar la solución.

Solución en Prolog

Declaración de Hechos

En la implementación se cuentan con Hechos, por ejemplo;

```
estadoFinal(estado(i,i,i,i)).
```

La sentencia anterior nos indica que el estado final, es decir, la solución deseada es que los cuatro actores estén en el mismo lado del río, en este caso el lado izquierdo, la estructura estado debe recibir 4 argumentos, que son las posiciones de cada uno de los personajes, que se tomarán en el siguiente orden, granjero, cabra, lobo y col.

Declaración de Reglas

Dentro del programa se declaran ciertas reglas, a continuación se explicarán algunas de ellas.

```
movimiento(estado(ACTUAL, CABRA, LOBO, COL), estado(DESTINO, CABRA, LOBO, COL)):- cruza(ACTUAL, DESTINO).
```

En la regla anterior se define un movimiento, podemos observar que sólo cambia el estado del granjero, lo que quiere decir que el se transportó el solo de un lado a otro del río, para el resto de los movimientos válidos la estructura es la misma, pero cambia el estado del actor o actores en cada caso. En total existen 4 movimientos válidos;

- El granjero cruza solo.
- El granjero cruza con el lobo.
- El granjero cruza con la cabra.
- El granjero cruza con la col

También se definieron restricciones, ya que en el acertijo se especifica que el lobo y la cabra no pueden quedarse solos ni la cabra con la col. Podemos observar la declaración de una restricción en la siguiente línea de código.

```
restriccion(estado(POSG, POSCL, POSCL, _)):- cruza(POSG, POSCL).
```

Para la otra restricción el formato es el mismo, pero en las posiciones de la cabra y la col.

Listas

La solución del problema se representa en una lista de estados, para evitar repeticiones se declaran las siguientes reglas. para verificar si un estado ya se encuentra en la solución.

```
visitados(X, [X|_]).
visitados(X, [_|C]):- visitados(X, C).
```

Solución

En esta sección del programa se hace uso de la recursividad, por lo que se define un estado base, al que se llegará una vez se encuentre la solución final del acertijo.

```
caminoSolucion([EA | C], [EA | C]) :- estadoFinal(EA).
```

En el caso contrario se definió la regla para añadir a la lista de soluciones un nuevo estado.

```
caminoSolucion([EA | C], CAMINOSOL):- movimiento(EA, EF), not(restriccion(EF)), not(visitados(EF, C)), caminoSolucion([EF, EA | C], CAMINOSOL).
```

Que especifica las siguientes condiciones;

- Debe ser un movimiento permitido (los cuales fueron declarados con anterioridad).
- No debe ser un estado restringido (declarados anteriormente).
- No debe existir el estado dentro de la lista de soluciones.

Funcionamiento del Sistema

La forma en la que se ejecuta el programa es la siguiente;

```
caminoSolucion([estado(d, d, d, d)], L)
```

Dentro de esta instrucción se define el estado inicial de los personajes, todos del lado derecho del río, además se envía una lista donde se van a guardar los estados de la solución.

Al ejecutar la instrucción obtenemos la siguiente salida.

```
L = [estado(i, i, i, i), estado(d, d, i, i), estado(i, d, i, i), estado(d, d, i, d),
estado(i, i, i, d), estado(d, i, d, d), estado(i, i, d, d), estado(d, d, d, d)]
```

La salida nos indica las acciones que realiza el granjero para lograr cruzar al otro lado del río con sus compras intactas.

La solución encontrada define los pasos

- Todos los actores están a la derecha
- El granjero cruza a la izquierda con la cabra.
- El granjero regresa a la derecha solo.
- El granjero cruza a la izquierda con el lobo.
- El granjero regresa la derecha con la cabra.
- El granjero cruza a la izquierda con la col.
- El granjero regresa solo.
- El granjero cruza a la izquierda con la cabra. (Todos los actores han llegado a la izquierda)

Podemos buscar la solución del problema partiendo desde cualquier punto de la solución, por ejemplo.

```
caminoSolucion([estado(d, i, d, d)], L)
```

El resultado es el siguiente.

```
L = [estado(i, i, i, i), estado(d, d, i, i), estado(i, d, i, i), estado(d, d, i, d),
estado(i, i, i, d), estado(d, i, d, d)]
```

Ventajas y Desventajas de la programación declarativa

Ventajas	Desventajas
Permite utilizar un alto nivel de abstracción.	Riesgo de tener un código comprensible sólo por el programador que lo desarrolló.
Ya que permite indicar a la computadora qué hacer no cómo, puede ser resistente a cambios externos.	Se vuelve complejo implementar características individuales.
Se puede realizar con métodos que no son conocidos en el momento de la programación	Está basado en una forma de pensar que no es habitual en las personas

Utilizar la programación declarativa para resolver este problema permitió representar las condiciones en forma comprimida, también hizo que el código fuera más comprensible, y la búsqueda de la solución se obtuviera de forma rápida.