Rompecabezas de caníbales y misioneros PROLOG

Gerardo Irazabal Monroy

31 de enero del 2024

Abstract

El lenguaje de programación Prolog, que se deriva de la lógica de primer orden, es excelente en la resolución de problemas que involucran relaciones complejas y datos estructurados. Se presta especialmente para aplicaciones en inteligencia artificial y campos que requieren búsqueda heurística y razonamiento lógico. En este informe, se explora la aplicación de Prolog en la solución de un problema clásico de razonamiento: el rompecabezas de misioneros y caníbales. Este rompecabezas plantea el desafío de transportar tres misioneros y tres caníbales a través de un río con un bote que puede llevar como máximo dos personas, bajo la restricción crítica de que los caníbales nunca pueden superar en número a los misioneros en ninguna orilla del río.

Figure 1: Imagen canibales y misioneros.

1 Introducción

Mediante la programación que se hará el prolog SWISH, vamos diseñar y ejecutar un programa en Prolog que resuelva eficazmente este dilema. La naturaleza del problema muestra una representación abstracta de los estados y un conjunto de operaciones que alteran estos estados bajo restricciones bien definidas. La resolución del problema no solo requiere que todos los participantes crucen el río con éxito, sino que también se adhieran a una serie de reglas que eviten cualquier escenario peligroso para los misioneros.

Las reglas son las siguientes, presentadas con notación matemática para enfatizar la naturaleza de las restricciones:

Sea M el número de misioneros y C el número de caníbales en la orilla inicial del río, tal que $M = C = 3$. El objetivo es alcanzar un estado donde $M = C = 0$ en la orilla inicial y $M = C = 3$ en la orilla opuesta, manteniendo la vida de todos los individuos durante el proceso.

El bote, denotado por B, tiene una capacidad máxima $\text{cap}(B)$, donde $\text{cap}(B) = 2$. Además, B no puede operar vacío, estableciendo que $0 < \text{num}(B) \leq \text{cap}(B)$, donde $\text{num}(B)$ es el número de personas en el bote en cualquier travesía.

La condición de seguridad, para cualquier configuración de misioneros y caníbales en ambas orillas, se define como $M \geq C$ o $M = 0$, asegurando que los misioneros nunca sean superados en número por los caníbales.

Este conjunto de reglas forma la base lógica y las restricciones operativas del rompecabezas. La implementación en Prolog del algoritmo de solución explora un espacio de estados bajo estas restricciones, haciendo uso de la recursividad y el backtracking para hallar una secuencia de movimientos válidos que resulte en el tránsito seguro de todos los misioneros y caníbales al otro lado del río.

2 Explicación del código

```
% Estado inicial y final
estado_inicial(estado(3, 3, izq)).
estado_final(estado(0, 0, der)).
```

Figure 2: Fragmento no.1.

Aquí se definen los estados inicial y final del problema. estado(**3, 3, izq**) representa tres misioneros y tres caníbales en la orilla izquierda (**izq**) del río, mientras que **estado(0, 0, der)** representa que todos han cruzado con éxito al lado derecho (**der**). Estos estados son importantes para iniciar la búsqueda de la solución y para saber cuándo se ha alcanzado el objetivo.

```
% Condición de peligro
en_peligro(estado(M, C, _)) :- M < C, M > 0.
en_peligro(estado(M, C, _)) :- M > C, C > 0, M < 3.
```

Figure 3: Fragmento no.2.

en_peligro/1 es una función que evalúa si un estado dado es peligroso. Un estado es peligroso si en alguna orilla los caníbales superan en número a los misioneros y hay al menos un misionero presente. Se utiliza para evitar que el programa realice movimientos que pongan en peligro a los misioneros..

```
% Acciones posibles
% Mover un misionero
accion(estado(M, C, izq), estado(NM, C, der), 'Misionero cruza a derecha') :-
    NM is M - 1, \+ en_peligro(estado(NM, C, der)), NM >= 0.
accion(estado(M, C, der), estado(NM, C, izq), 'Misionero regresa a izquierda') :-
    NM is M + 1, \+ en_peligro(estado(NM, C, izq)), NM =< 3.
```

Figure 4: Fragmento no.3.

En este parte se definen dos acciones relacionadas con el movimiento de un misionero en el juego de los **misioneros y caníbales**. La primera acción, `accion(estado(M, C, izq), estado(NM, C, der), 'Misionero cruza a derecha')`, describe el movimiento de un misionero desde la orilla izquierda (`izq`) a la derecha (`der`), disminuyendo el contador de misioneros en la orilla izquierda (`NM is M - 1`) y verificando que el nuevo estado no sea peligroso (`\+ en_peligro(estado(NM, C, der))`). La segunda acción, `accion(estado(M, C, der), estado(NM, C, izq), 'Misionero regresa a izquierda')`, representa el regreso de un misionero a la orilla izquierda, aumentando el contador de misioneros en esta orilla (`NM is M + 1`) y asegurando que el estado resultante sea seguro (`\+ en_peligro(estado(NM, C, izq))`), lo cual verifican que el número de misioneros se mantenga dentro de límites válidos (`NM >= 0`, `NM =< 3`).

```
% Mover un caníbal
accion(estado(M, C, izq), estado(M, NC, der), 'Caníbal cruza a derecha') :-
    NC is C - 1, \+ en_peligro(estado(M, NC, der)), NC >= 0.
accion(estado(M, C, der), estado(M, NC, izq), 'Caníbal regresa a izquierda') :-
    NC is C + 1, \+ en_peligro(estado(M, NC, izq)), NC =< 3.
```

Figure 5: Fragmento no.4.

Definimos acciones para el movimiento de un caníbal. La acción `accion(estado(M, C, izq), estado(M, NC, der), 'Caníbal cruza a derecha')` describe el traslado de un caníbal de la orilla izquierda (`izq`) a la derecha (`der`), reduciendo el número de caníbales en la orilla izquierda (`NC is C - 1`) y asegurando que el estado resultante no sea peligroso (`\+ en_peligro(estado(M, NC, der))`). De manera similar, `accion(estado(M, C, der), estado(M, NC, izq), 'Caníbal regresa a izquierda')` representa el retorno de un caníbal a la orilla izquierda, incrementando el contador de caníbales en dicha orilla (`NC is C + 1`) y comprobando que no resulte en un estado peligroso (`\+ en_peligro(estado(M, NC, izq))`), asegurando de que el número de caníbales se mantenga dentro de los límites válidos (`NC >= 0`, `NC =< 3`).

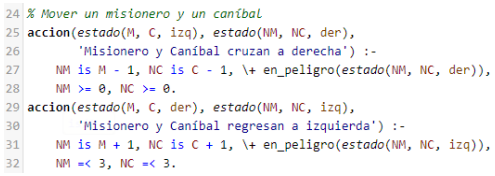

Figure 6: Fragmento no.5.

Aquí se aborda las acciones combinadas. La acción `accion(estado(M, C, izq), estado(NM, NC, der), 'Misionero y Caníbal cruzan a derecha')` permite que un misionero y un caníbal crucen juntos del lado izquierdo (`izq`) al derecho (`der`). Se realiza disminuyendo tanto el contador de misioneros (`NM is M - 1`) como el de caníbales (`NC is C - 1`) en la orilla izquierda y verificando que no se genere un estado peligroso (`\+ en_peligro(estado(NM, NC, der))`). La acción inversa, `accion(estado(M, C, der), estado(NM, NC, izq), 'Misionero y Caníbal regresan a izquierda')`, describe el regreso de ambos al lado izquierdo, incrementando sus respectivos contadores (`NM is M + 1`, `NC is C + 1`) y asegurando que la orilla izquierda permanezca segura. Estas acciones se encargan de mantener los contadores de misioneros y caníbales dentro de los límites seguros y válidos.

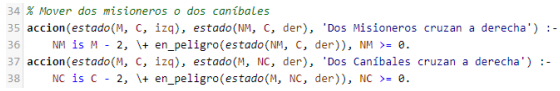

Figure 7: Fragmento no.6.

Ahora, aquí nos centramos en el movimiento de dos personajes a la vez. La acción `accion(estado(M, C, izq), estado(NM, C, der), 'Dos Misioneros cruzan a derecha')` permite que dos misioneros crucen juntos desde la orilla izquierda (`izq`) a la derecha (`der`), disminuyendo el contador de misioneros en la orilla izquierda (`NM is M - 2`) y asegurándose de que el nuevo estado no sea peligroso (`\+ en_peligro(estado(NM, C, der))`). Por otro lado, la acción `accion(estado(M, C, izq), estado(M, NC, der), 'Dos Caníbales cruzan a derecha')` describe el movimiento de dos caníbales de la misma manera, reduciendo el contador de caníbales en la orilla izquierda (`NC is C - 2`) y verificando que el estado resultante sea seguro y salvaguardando por mantener la integridad y seguridad del grupo.

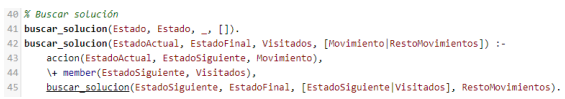

Figure 8: Fragmento no.7.

Definimos el predicado `buscar_solucion`, para encontrar una secuencia de movimientos que resuelva el rompecabezas. El caso base `buscar_solucion(Estado, Estado, _, [])` indica que la solución se ha encontrado cuando el estado actual coincide con el estado final. La parte recursiva `buscar_solucion(EstadoActual, EstadoFinal, Visitados, [Movimiento|RestoMovimientos])` explora los movimientos posibles desde el estado actual. Utiliza el predicado `accion` para generar un `EstadoSiguiente` y añade este estado a la lista de `Visitados` para evitar ciclos. Esta estrategia de búsqueda recursiva continúa hasta encontrar una secuencia de movimientos que lleve del estado inicial al estado final.

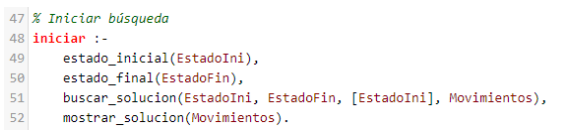

Figure 9: Fragmento no.8.

El predicado `iniciar` marca el comienzo de la solución del rompecabezas de los **misioneros y caníbales**. Este predicado establece los estados inicial y final a través de `estado_inicial(EstadoIni)` y `estado_final(EstadoFin)`, respectivamente. Posteriormente, invoca `buscar_solucion` con el estado inicial, el estado final y una lista inicial de estados visitados que contiene solo el estado inicial. `buscar_solucion` retorna los movimientos necesarios para resolver el juego, `mostrar_solucion(Movimientos)` se encarga de imprimir esta secuencia de movimientos.

```
54 % Mostrar solución
55 mostrar_solucion([]) :- writeln('Todos han cruzado el río con éxito.').
56 mostrar_solucion([M|Ms]) :-
57     writeln(M),
58     mostrar_solucion(Ms).
```

Figure 10: Fragmento no.9.

Finalmente, `mostrar_solucion` se utiliza para exhibir la secuencia de movimientos que resuelven el rompecabezas. Este predicado se maneja recursivamente: si la lista de movimientos está vacía (`mostrar_solucion([])`), imprime un mensaje de éxito indicando que todos los personajes han cruzado el río con seguridad.

3 Resultados

Para poder correr el código, en la terminal, colocamos el fragmento **iniciar**, tal y como se muestra en la imagen a continuación, para posteriormente, dar click en **run**

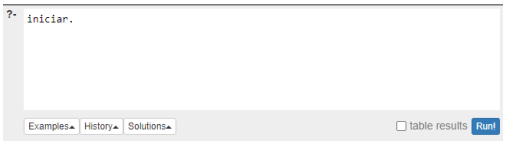

Figure 11: Terminal "iniciar.".

Y el resultado que obtuvimos fue la siguiente.

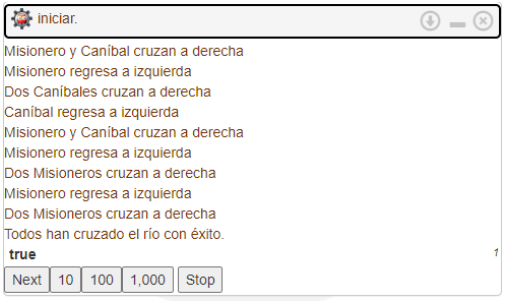

Figure 12: Resultados de la terminal.

Interpretemos los resultados.

- **Misionero y Caníbal cruzan a derecha**: Al principio, un misionero y un caníbal cruzan el río hacia la derecha. Esto cumple con la restricción de no tener más caníbales que misioneros en ningún lado.

- **Misionero regresa a izquierda**: Luego, uno de los misioneros regresa solo hacia la izquierda. Esto asegura que nunca haya más caníbales que misioneros en el lado izquierdo.

- **Dos Caníbales cruzan a derecha**: Dos caníbales cruzan hacia la derecha. Aunque ahora hay más caníbales que misioneros en el lado derecho, es seguro porque hay un misionero en ese lado también.

- **Caníbal regresa a izquierda**: Uno de los caníbales regresa a la izquierda para mantener el equilibrio en ese lado.

- **Misionero y Caníbal cruzan a derecha**: Un misionero y un caníbal cruzan nuevamente hacia la derecha.

- **Misionero regresa a izquierda**: Uno de los misioneros regresa a la izquierda.

- **Dos Misioneros cruzan a derecha**: Dos misioneros cruzan hacia la derecha. Ahora, hay más misioneros que caníbales en ambos lados.

- **Misionero regresa a izquierda**: Uno de los misioneros regresa a la izquierda para mantener el equilibrio.

- **Dos Misioneros cruzan a derecha**: Finalmente, los dos misioneros restantes cruzan hacia la derecha. Con esto, todos los misioneros y caníbales han cruzado el río con éxito, y la restricción de no tener más caníbales que misioneros en ningún lado se mantiene en todo momento.

4 Conclusión

Bajo esta interpretación correcta, la secuencia de movimientos cumple con todas las reglas establecidas:

- **T**odos los misioneros y caníbales son movidos de una orilla a la otra.
- **E**n ningún momento se mueve el bote vacío y no se excede su capacidad de dos personas.

Por lo tanto, podemos concluir que la secuencia de movimientos resuelve el rompecabezas de forma segura y efectiva, cumpliendo con todas las restricciones establecidas.

References

[1] Prolog Tutorials. *Misioneros y caníbales con Prolog.* Recuperado de https://www.youtube.com/watch?v=QKP0lb7PHzs