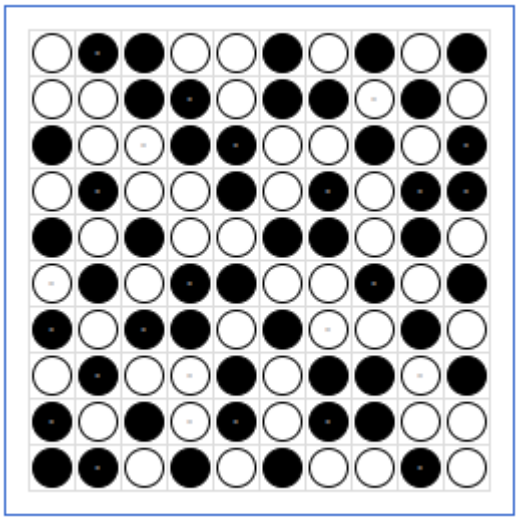
UNIVERSIDAD NACIONAL de MAR DEL PLATA

FACULTAD DE INGENIERIA

Inteligencia Artificial

Trabajo Final – Binairo



Integrantes:

Nucci, Manuel

Pico, Juan Fernando

Fecha de entrega:

29 de noviembre de 2018

Trabajo Final – Binairo

Takuzu, también conocido como Binairo, es un rompecabezas de colocación de números basado en la lógica. Binairo se juega sobre una cuadrícula rectangular sin tamaño estándar con 1s y 0s o, como en nuestro caso, con círculos blancos y negros. Algunas celdas comienzan con los valores ya colocados mientras que el resto se encuentran vacías. El objetivo es colocar círculos en todas las celdas de tal manera que:

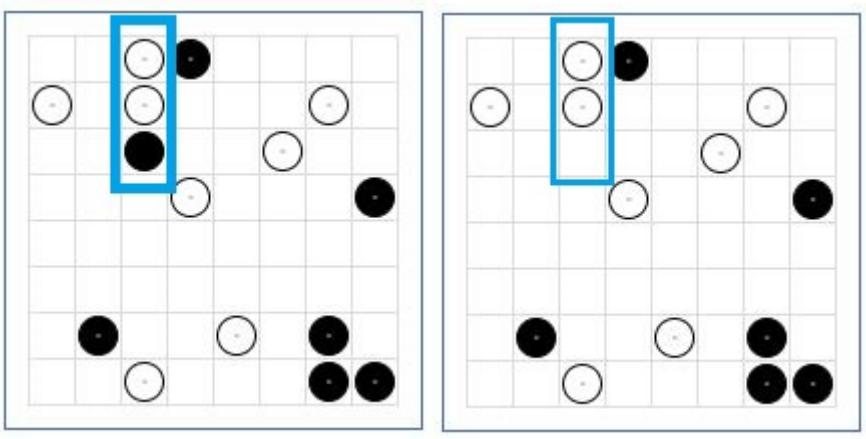
- Cada fila y cada columna debe contener un número igual de círculos blancos y negros.
- Más de dos círculos del mismo color no pueden ser adyacentes.
- Cada fila y columna es única.

Con estas tres simples reglas se debe poder resolver cualquier rompecabezas que se presente. Para ello, en las primeras semanas que se abordó al juego, se dispuso a definir distintas reglas de resolución que pudiesen ser aplicadas en un programa. Desde un primer momento se propuso evitar el uso de fuerza bruta o, lo que es equivalente, introducir un círculo en un casillero ante la imposibilidad de saber que acción seguir a continuación. Para ello era necesario refinar las reglas que permitan resolver al rompecabezas hasta tal punto que siempre se pueda encontrar una acción a ejecutar. Después de mucho esfuerzo y de incontables partidas jugadas se logró armar un listado de pasos ordenado de mayor a menor probabilidad de ocurrencia que puede resolver cualquier tablero que se le presente.

A continuación, se muestra cada una de las acciones a seguir con una breve explicación de las mismas:

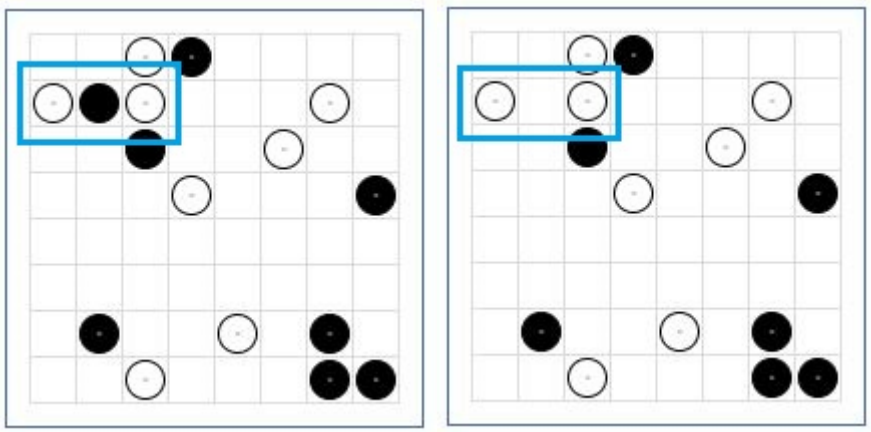
Pasos para resolver el juego Binairo

1. Cuando existen dos símbolos iguales adyacentes, alrededor de ellos se debe colocar un símbolo del color opuesto.



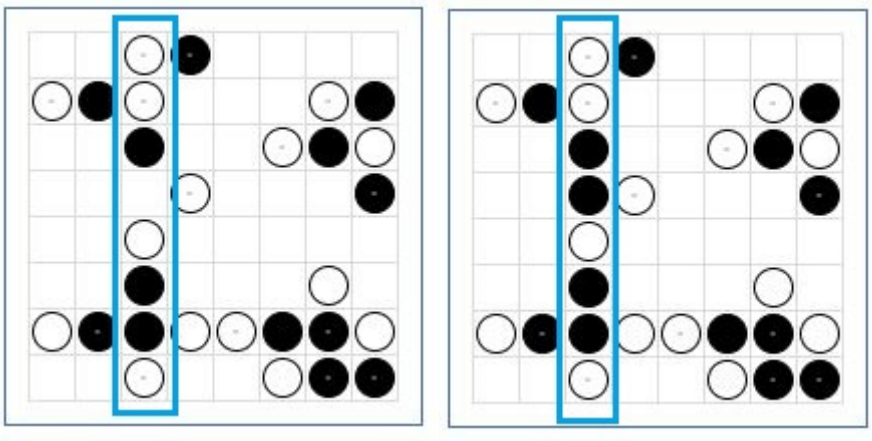
Esta regla junto con la 2 es la que más probabilidades de ocurrencia tiene. Dos círculos del mismo color pueden aparecer en cualquier parte del tablero, tanto en una fila como una columna, motivo por el cual deben analizarse todas estas posibilidades.

2. Dos símbolos del mismo tipo separados por un espacio determinan que el símbolo entremedio es del tipo opuesto.



Al igual que la regla 1, esta regla también es de uso frecuente y puede aparecer tanto en filas como columnas. Como se explicará más adelante, para obtener una mayor eficiencia a la hora de resolver el rompecabezas se dispuso que el ciclo que aplica las reglas 1 y 2 las evalúe en conjunto, con el fin de evitar recorridos extras a la cuadrícula.

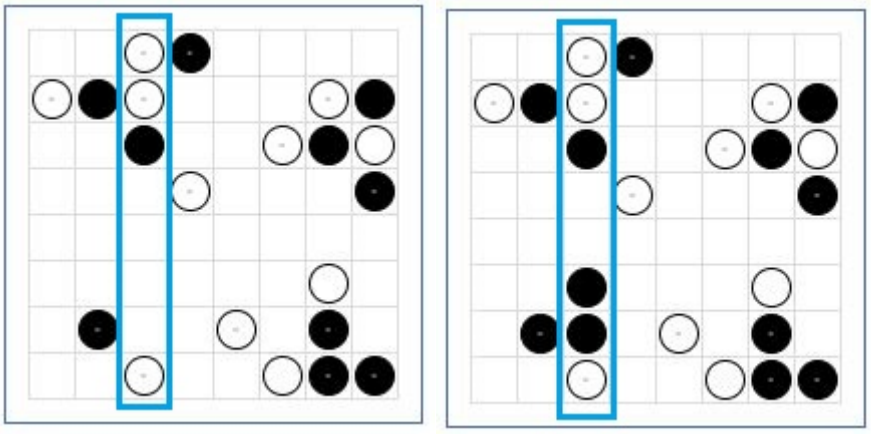
3. Si existe una fila en la que la mitad de los elementos son de un mismo tipo, entonces los símbolos restantes serán del símbolo opuesto.



Una regla simple pero muy poderosa. La aplicación de las reglas 1 y 2 no tiene en muchos de los casos el potencial de descubrir cuál es el símbolo faltante en una fila o columna, tal como se muestra en el ejemplo. Es aquí cuando entra la regla 3 y permite salvar este inconveniente. Para su aplicación se recorren todas las filas y luego todas las columnas.

Hasta este punto las reglas enunciadas son triviales, surgen de la lectura de los requerimientos que nos impone el juego y de pensar un poco en cómo solucionarlos. La regla 4 junto con sus subreglas es la clave de la resolución del rompecabezas. Su enorme potencial permite realizar ciertas deducciones lógicas que a priori no son evidentes, pero luego, cuando uno se dispone a analizar detenidamente la situación, encuentra el por qué un círculo debería ir en tal casillero.

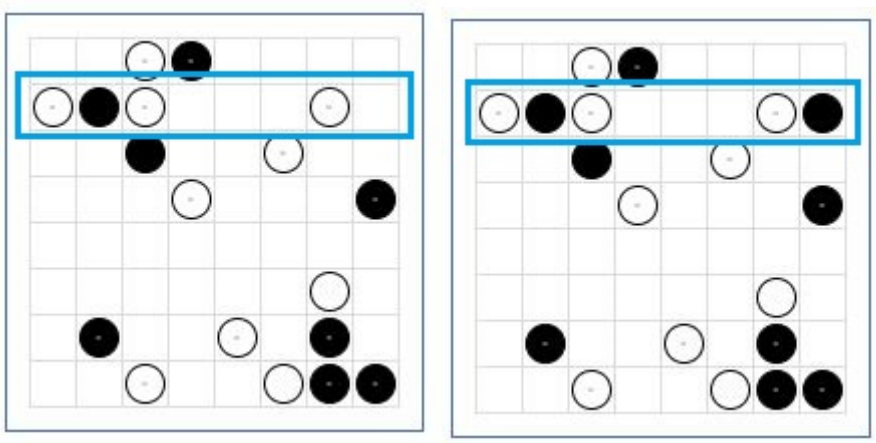
4. Cuando falta un símbolo de tipo A:
   4.1. Y faltan ***dos o más*** de B tiene que haber un B que cuente con dos espacios libres adyacentes a él (pueden ser consecutivos o estar rodeado por ellos). Si esto se da entonces sabemos que en uno de esos lugares estará el símbolo faltante de A. En consecuencia, el ***resto*** de los lugares que sobran se completarán con B's.



En el ejemplo puede observarse que a la columna le falta un único círculo blanco. A su vez, existe un círculo negro que tiene dos lugares adyacentes libres a él. Como uno de los requisitos del juego es que nunca existan tres símbolos del mismo tipo consecutivos, deducimos que el símbolo blanco estará en alguna de dichas posiciones. En consecuencia, ubicado el símbolo blanco allí, podemos afirmar que el resto de los lugares libres se completarán con círculos negros (en este caso).

La restricción de que falten dos o más símbolos de B es impuesta debido a que si sólo faltase uno de él y se cumpliese que tiene dos lugares adyacentes libres, entonces se completarían los lugares restantes con círculos negros los cuales no existirían (puesto que hay dos espacios libres y faltan sólo un símbolo A y otro de B en la fila o columna).

   4.2. Si la condición anterior no se cumple, pero faltan ***3 o más símbolos*** de B, contando con ***tres o más*** celdas libres ***consecutivas*** en algún lugar, entonces sabemos que ahí irá el símbolo A. En consecuencia, afirmamos que **el resto de los lugares se completarán con símbolos de tipo B**.

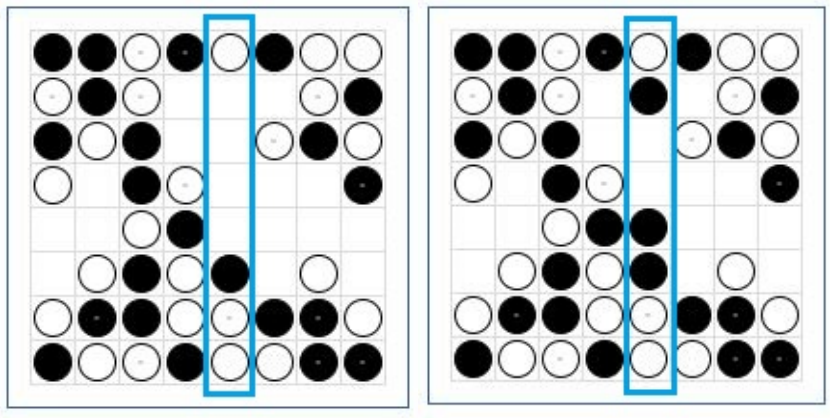


Observamos en el ejemplo que falta un símbolo blanco en la fila. A su vez, notamos que faltan tres símbolos negros, tal como impone la regla 4.2. Al encontrar tres lugares **libres consecutivos** podemos afirmar que allí deberá ser situado el círculo blanco con el fin de evitar el posicionamiento de tres símbolos negros consecutivos, lo cual invalidaría uno de los requerimientos del Binairo.

Como consecuencia, al haber sido determinado el intervalo de lugares en donde podría ser ubicado el círculo blanco, podemos deducir que el resto de los lugares libres que existan en la fila o columna deberán ser completados con símbolos negros como es este caso.

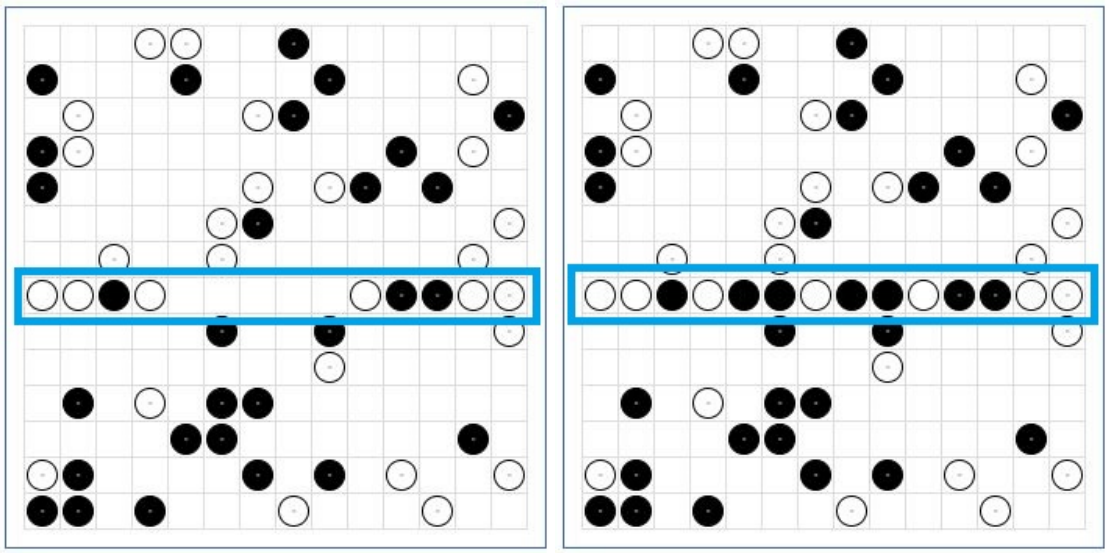
Nuevamente recalcamos el hecho de que deben faltar tres o más símbolos de B ya que si faltasen dos de B y uno de A y hubiese tres espacios libres consecutivos, se reunirían todas las condiciones, pero no habría espacios extras libres que completar en la fila o columna, conduciendo a un desperdicio de recursos y tiempo al tomarse el trabajo de evaluar la situación de la fila o columna.

4.2.1. Si la cantidad de celdas consecutivas es exactamente 4 entonces, además de haber completado los lugares que determinó la regla 4.2, en los dos extremos de las celdas consecutivas se deben colocar símbolos de tipo B.



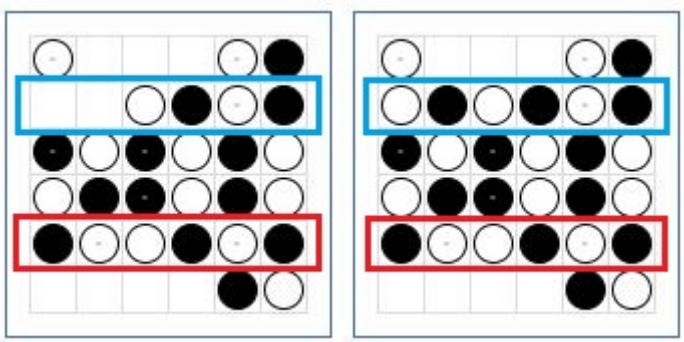
Al encontrarnos con 4 lugares libres adyacentes y faltar un único símbolo blanco, debemos preguntarnos que podría ocurrir si es posicionado en cualquiera de dichos lugares. Si es colocado en alguno de los dos casilleros centrales vemos que no ocurre ningún problema ya que las demás celdas podrían ser completadas con uno y dos símbolos negros, o viceversa. Luego, si el símbolo blanco es colocado en alguno de los extremos se encuentra que se produciría una violación de los requerimientos del Binairo puesto que deberían ser colocados tres círculos negros consecutivos. Es por esta razón que el símbolo blanco no puede ser colocado en ninguna de las puntas. En conclusión, es posible afirmar que en dichos lugares deberán ser colocados dos símbolos negros (o de tipo B hablando genéricamente).

4.2.2. Si la cantidad de celdas consecutivas es exactamente 5 entonces dichos espacios se completan siguiendo una secuencia fija de la forma B B A B B.



Al faltar un único símbolo de tipo A y existir cinco lugares libres adyacentes se encuentra que la única posibilidad es que dicho símbolo sea ubicado en el centro, evitando de esta forma que queden 3 o 4 símbolos de tipo B adyacentes y que invalidarían al rompecabezas.

5. Si le faltan dos símbolos a una fila o columna comprobar si no existe alguna otra fila o columna que ***esté completa*** y que salvo esas dos posiciones sea ***idéntica*** a la primera. En este caso, completarla con los símbolos ***opuestos***.



El último requerimiento que nos impone el Binairo es el siguiente:

"Cada fila y columna es única."

Para garantizar el mismo es necesario que, una vez que han sido aplicadas las reglas de la 1 a la 4 sin resultados, entonces se verifique lo propuesto por la regla 5.

Un dato extra a recalcar es el hecho de que uno podría pensar que, en el caso que encuentre dos filas o columnas a las que ambas les falten dos símbolos en las mismas posiciones también podría ser completadas con el opuesto. Sin embargo, aquí nos encontramos con que podrían ser completadas de la forma AB – BA o de la forma opuesta BA – AB. Esto conduce a que la colocación de los círculos no sea determinista y con un 100% de seguridad, con lo que podría darse el caso de que la elección de una u otra alternativa conduzca a que cuando el tablero sea completado íntegramente surjan inconsistencias en alguna de las filas o columnas por haber dado un paso que no se estaba completamente seguro. Es por esta razón que dicha situación no se tuvo en cuenta y, a pesar de ello, la regla 5 es suficientemente poderosa para encontrar alguna celda a completar cuando todas las reglas anteriores han fallado.

Con la definición y programación de estas cinco reglas y sus casos particulares en la 4 es posible resolver cualquier tablero que se le presente al programa. Han sido probados rompecabezas de 6x6, 8x8, 10x10, 14x14, 20x20 y hasta 30x30, todos en dificultad difícil, obteniendo resultados satisfactorios.

Implementación del programa en Prolog

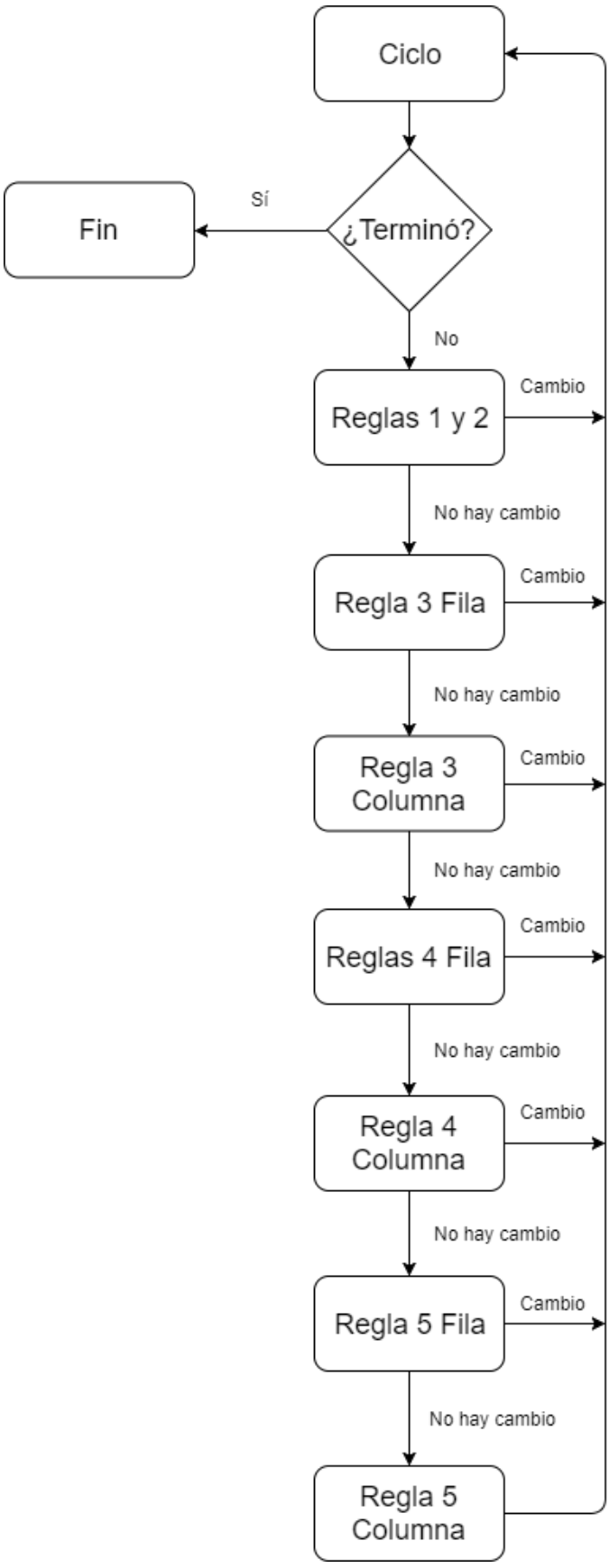
Uno de los primeros desafíos que se encontró a la hora de encarar la resolución del juego fue determinar qué estructura de datos utilizar para representar el conocimiento inicial que se tenía del mismo, conformado por el tablero inicial y las reglas del juego. Debe tenerse en cuenta que a este conocimiento base se le sumaron nuevas reglas y pautas basadas en patrones que fueron determinados en función de la experiencia que se adquirió a través del juego de múltiples partidas.

En consecuencia, fue necesario determinar una estructura de datos propicia para poder recuperar el conocimiento generado, ayudando de esta manera a la resolución de la lógica aplicada. Para lograr este cometido se decidió contar con hechos que representen cada unidad de este conocimiento con la ayuda de distintas herramientas de Prolog. Cada hecho almacenado

representa un casillero lleno del tablero que ha sido deducido por el programa. El conjunto de todos ellos conforma una gran base de datos a la que el programa puede acceder y consultar en cualquier parte del código que lo requiera.

En conclusión, la ventaja de esta representación es que no requiere el manejo de una estructura compleja (como es una lista), sino que los hechos están accesibles de forma instantánea y pueden ser consultados como si se trabajase con una base de datos, sin contar con la necesidad de estar acarreando parámetros y estructuras innecesarias.

Ciclo de resolución del tablero



Dificultades en el desarrollo

Para la aplicación de las reglas 1 y 2 se partió de la primera celda del tablero y se recorrió hasta la última, mirando siempre hacia abajo y a la derecha de cada casillero. Realizando estas comprobaciones se pudieron ahorrar muchas verificaciones que las celdas anteriores ya habían realizado.

Luego, a partir de esta estrategia surgieron distintos casos especiales como son las dos últimas columnas del lado derecho y las dos últimas filas del lado inferior. En estos casos se tenía que tener cuidado hacia dónde mirar ya que dependiendo de la situación en que se encontraba se debía analizar hacia abajo o hacia la derecha, para evitar caerse de la matriz.

A raíz de esto último surgieron también otros casos especiales como fueron las 4 celdas del borde inferior derecho. En ellas no se debió analizar nada gracias al trabajo previo que habían efectuado las demás celdas.

Por último, se tuvo en cuenta que en el caso que una celda vacía sea llenada aplicando la regla 1, luego se reapliquen las reglas 1 y 2 ahora con la celda ocupada aprovechando la misma pasada sobre la matriz.

En la regla 4 se presentó el inconveniente de cómo determinar cuántos espacios vacíos consecutivos teníamos entre dos celdas llenas, ya que la detección de dichos lugares era una necesidad recurrente en las sucesivas subreglas que surgen a partir de la regla 4. Para esto se decidió armar una lista de posiciones de celdas llenas y, a través de ellas, conocer las distancias entre celdas ocupadas. A partir de esta solución y buscando hacer un recorrido eficiente sobre la fila o columna sin hacer verificaciones innecesarias se decidió mirar solamente a la derecha de la celda llena en la que se estaba haciendo foco. A raíz de esto surgieron dos casos especiales los cuales eran cuando existían uno o más espacios vacíos antes del primer elemento o después del último.

Por último, el recorrido sobre el ciclo de resolución y las sucesivas decisiones sobre si continuar o no entraron en conflicto con el backtracking que nos provee Prolog. Para solucionar este problema y evitar inconsistencias en la lógica del programa se debió determinar dónde era necesario cortar el backtracking. Por lo tanto, se hizo uso del operador cut (!) en los lugares donde estábamos seguros que el programa no tenía que reanalizar el tablero. Con esto se logró mantener la lógica del ciclo mostrado anteriormente de la forma que desde un principio se pensó.

Conclusión

Al momento de encarar este proyecto se nos vinieron a la cabeza muchas ideas interesantes que la IA podía resolver. Sin embargo, pronto nos dimos cuenta que buscar un juego iba a proponernos un escenario donde íbamos a poder utilizar los conocimientos adquiridos en la materia. Investigando distintos desafíos de lógica nos encontramos con Binairo, un rompecabezas de lógica divertido y simple en un principio pero que, al jugarlo repetidamente, se fue encontrando una lógica más compleja para su resolución. Prolog fue la

herramienta indicada para representar esa lógica y luego, pedirle que la utilice para encontrar la solución del juego.

La tarea de llevar el conocimiento adquirido sobre la lógica de Binairo a un código eficiente en Prolog fue mucho más compleja de lo esperado. Durante todo este trabajo estuvimos constantemente enfrentándonos a la tarea de tomar decisiones importantes que iban a determinar tanto la eficiencia del programa como su correcto funcionamiento. La prueba de que estas decisiones fueron correctas está demostrada en el breve tiempo que demora la resolución de una partida compleja.

En conclusión, podemos decir que el trabajo representó un desafío para demostrar nuestras habilidades y que, gracias a la aplicación sistemática y cuidadosa de las mismas que se tuvo en el análisis del juego y su posterior codificación, permitió desarrollar un software a la altura de lo que requería la cátedra.