Práctica 1: Senku

El *senku* es un solitario en el que, partiendo de un tablero en el que todas las posiciones del tablero menos una están ocupadas con fichas, se van eliminado fichas hasta quedarnos con una única ficha. Para ello, las piezas saltan unas sobre otras. Cuando una pieza salta sobre otra, esta se "come" aquella sobre la que salta, retirándose del tablero. Por tanto, el único movimiento válido consiste en saltar sobre una ficha adyacente hasta una posición vacía al otro lado de esta *en línea recta*, eliminado la ficha sobre la que se saltó. Si alcanzamos una configuración en la que nos quedan dos o más fichas que no pueden comerse entre sí habremos perdido.

Podéis encontrar más información sobre el juego en su entrada en la Wikipedia. El objetivo de esta práctica es resolver el juego para distintos tipos de tablero usando Maude. Además de módulos Maude con las especificaciones desarrolladas, se entregará un informe (en formato pdf) con las respuestas a cada una de las preguntas que se plantean en el ejercicio. Todas las respuestas deberán estar adecuadamente justificadas.

El solitario puede jugarse en diferentes tipos de tableros.

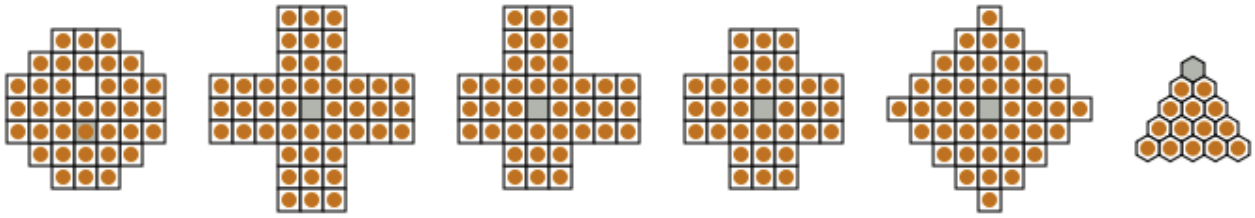

Figura 1: Tableros

Dada la complejidad del problema a resolver, empezaremos resolviendo tableros 4x4 como el de la figura 2, pero la estructura diseñada debe permitir tableros de los formatos de la figura 2.

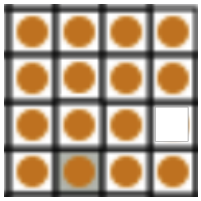

Figura 2: Tableros

1. Como siempre, se nos presentan varias opciones para representar el tablero. Plantea dos alternativas: una fija, modelada como una lista de listas, y otra basada en objetos, modelada como una colección de fichas con una posición. Especifica dos módulos de sistema de Maude, uno `PEG-SOLITAIRE-FIJA` y otro `PEG-SOLITAIRE-OBJETOS`, con
   - Un tipo de datos `Board` para representar un tablero de Senku, con tantas operaciones como se considere necesario, pero al menos tendremos una operación
     `op count : Board -> Nat .`
     que calcule el número de fichas en un tablero dado.
   - Para cada una de las dos opciones anteriores, proporciona la reglas necesarias para modelar los distintos movimientos que pueden realizarse en un tablero.
   - Discute las ventajas e inconvenientes de cada una de las dos opciones.
2. ¿Tienen sentido las signaturas de tus especificaciones? (¿Son *sensibles*?) Razona tu respuesta.
3. ¿Son tus especificaciones pre-regulares? Razona tu respuesta.
4. Razona sobre la confluencia, terminación y coherencia de las especificaciones desarrolladas.
5. Para las dos especificaciones alternativas desarrolladas, utiliza el comando `search` para encontrar una posible solución al problema a partir de un tablero 4x4 en la posición inicial mostrada en la imagen de la figura 3 (posición libre en la fila 3, columna 4). A partir de dicha configuración

de partida pueden alcanzarse dos posibles configuraciones ganadoras. Indica el o los comandos utilizados y la salida obtenida. Compara las salidas obtenidas para las dos implementaciones y justifica su diferencia en tiempo de ejecución.

Para el resto del ejercicio, selecciona una de las especificaciones.

6. Utiliza los comandos `show path labels` y `show path` para obtener los pasos a seguir hasta completar el juego a cada una de las configuraciones ganadoras. Indica la diferencia entre las secuencias de movimientos seguidos en cada una de estas soluciones.
7. Interpreta la salida del comando `show search graph` para representar gráficamente el subgrafo con los 20 primeros estados del grafo de búsqueda. (El gráfico puedes hacerlo utiliando la herramienta que quieras, pero un dibujo a mano es suficiente. Puedes escanear dicho dibujo para incluirlo en el informe de la práctica.)
8. Utiliza el comando `search` para encontrar la primera situación de bloqueo (ninguna ficha puede comerse a otra). ¿Cuántos movimientos es necesario realizar para alcanzarla? ¿Cuántas situaciones de bloqueo distintas son alcanzables a partir de esta configuración inicial?
9. La posición que queda libre inicialmente es importante a la hora de empezar el juego. Prueba con diferentes configuraciones para ver si tienen solución o no y cuántas soluciones tiene cada una. Observa que al ser cuadrado el tablero, y con los movimientos que podemos hacer, no es necesario probar 16 configuraciones iniciales distintas. ¿Cuál es el número mínimo de configuraciones que tenemos que probar para comprobar todas las posibilidades? ¿Cuántas configuraciones de partida de estas 16 posibles tienen solución? Razona tu respuesta.
10. Se dice que tenemos una solución *perfecta* si la única ficha que queda al final de una partida se encuentra en la posición donde inicialmente estaba el hueco. ¿Es perfecta alguna de las soluciones para alguna de las posibles configuraciones de partida?
11. Prueba ahora con un tablero 5x5, posición libre en la fila 3, columna 4. (El tiempo de ejecución puede ser largo, en torno a los 10-15 minutos.)
12. Analiza el número de estados visitados para los problemas con tablero 4x4 y 5x5. ¿Puedes establecer algún patrón que permita estimar cuánto tardaría la ejecución del problema para un tablero 6x6 o 7x7?
13. En todos los problemas anteriores, la solución siempre se encuentra en el último nivel del grafo de búsqueda. ¿Por qué? ¿Crees que ocurrirá lo mismo en cualquier problema de búsqueda en un módulo de sistema? ¿Por qué?
14. El tablero más a la derecha de los mostrados en la figura 1 es lo que se conoce como un tablero triangular. Podemos representar tableros triangulares utilizando nuestra representación de tableros rectangulares simplemente considerándolos rotados como se muestra en la figura 3. Los movimientos en el tablero se corresponden con los movimientos definidos en el rectangular como se muestra en la figura. Adicionalmente, podemos considerar un tercer tipo de movimiento correspondiente a mover en horizontal en el tablero triangular, que podríamos representar como un nuevo movimiento en diagonal sobre el tablero rectangular. Véase figura 4. Define un módulo de sistema `TRIANGULAR-PEG-SOLITAIRE` que extienda el módulo anterior `PEG-SOLITAIRE-*` (el que hayas elegido). El tablero de un juego triangular será del mismo tipo `Board` ya definido, pero ahora serán permitidos movimientos adicionales.

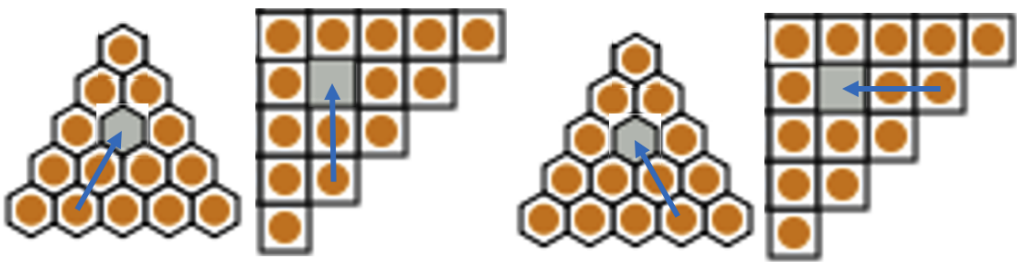

Figura 3: Tableros

15. Utiliza el comando `search` para encontrar una posible solución al problema a partir de un tablero

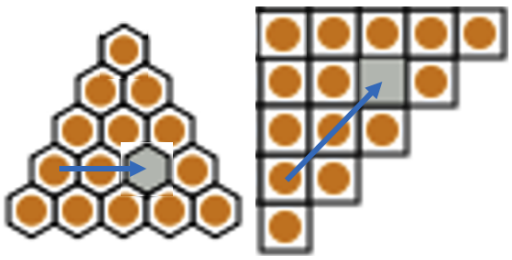

Figura 4: Tableros

triangular de lado 5 como el que se muestra en la imagen.

16. ¿Cuál es el número mínimo de configuraciones iniciales distintas que debemos contemplar para garantizar el haberlas contemplado todas? Razona tu respuesta.
17. Estima el número mínimo de pasos para alcanzar una solución para problemas con tablero triangular con lados 4, 6 y 8.
18. ¿Cómo podríamos mejorar el planteamiento del problema (especificación, análisis, etc.) para poder considerar tableros más grandes?