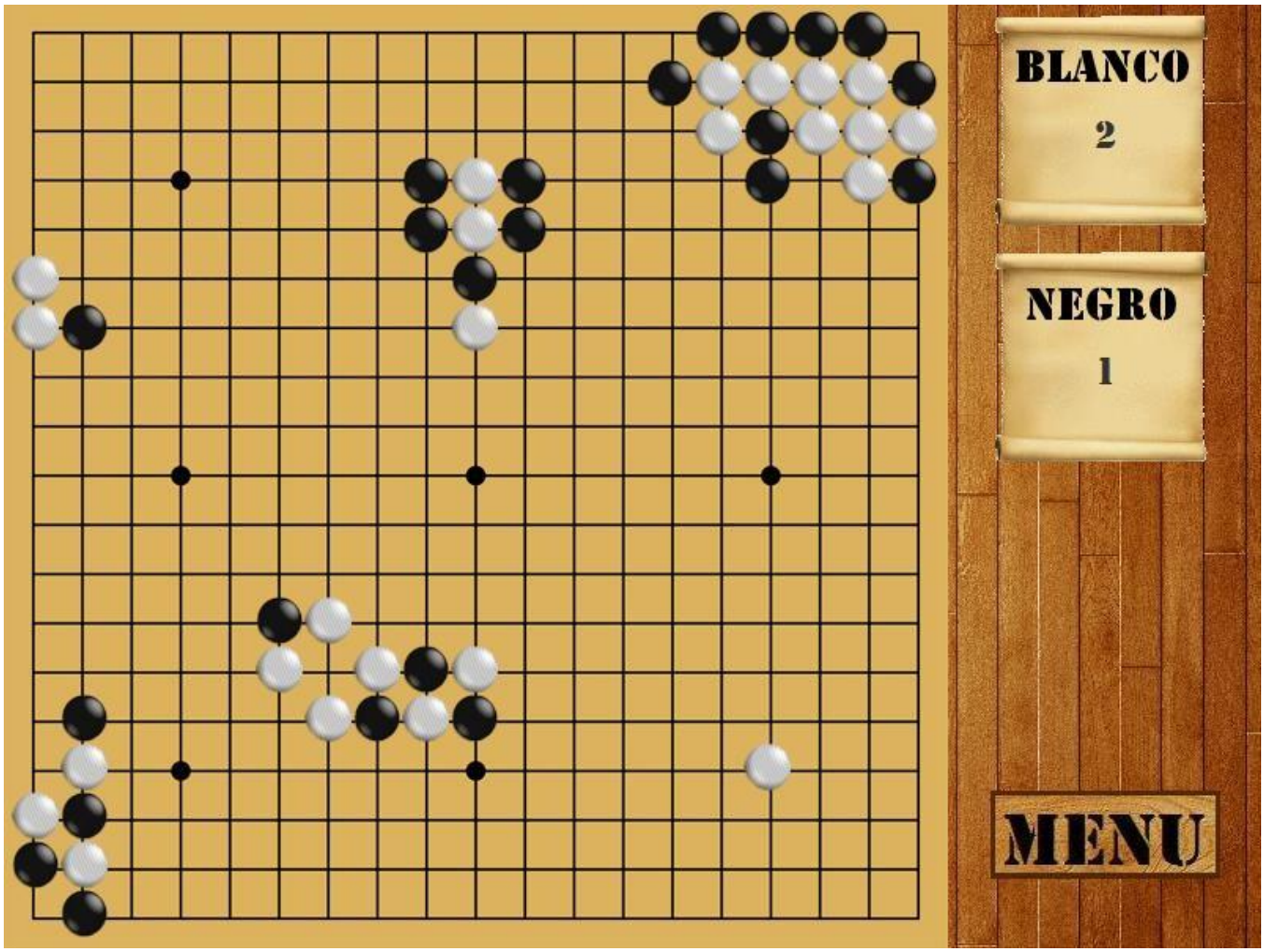
Juego GO! - LCC 2019

Mauro Montano – Rafael Aloisio



[DOCUMENTACIÓN]

A continuación se detallarán las decisiones de diseño y técnicas para la resolución del problema planteado, tanto a nivel del lenguaje que lleva a cabo la lógica del juego (Prolog), como también a las modificaciones hechas en la interfaz gráfica Web ya dada, y cómo ambos lenguajes interaccionan para lograr el objetivo propuesto. Además se incluyen imágenes para ayudar a la comprensión de las explicaciones.

INDICE

PAG.

IMPLEMENTACIÓN DE LA LÓGICA DEL JUEGO

PROLOG

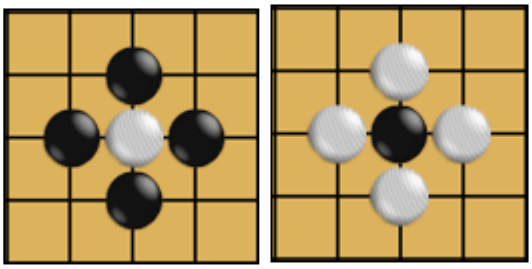
Características generales del juego.

A continuación se detallarán las características generales del juego, tenidas en cuenta en la implementación.

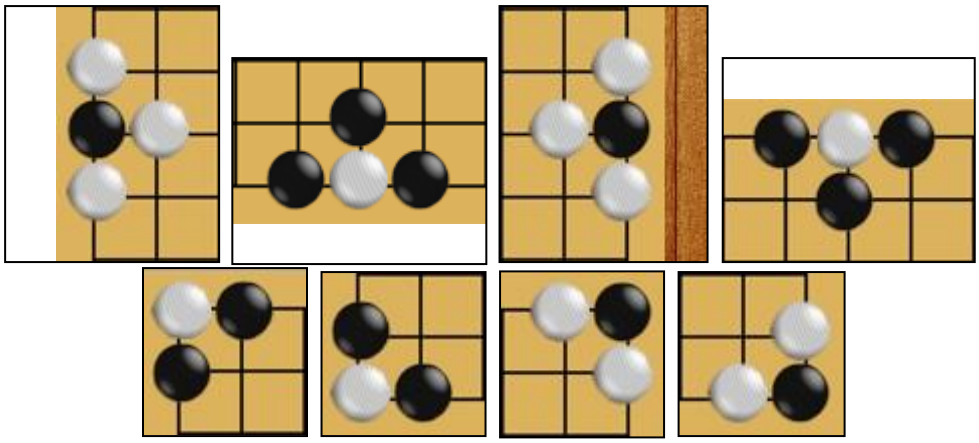
El juego se desarrolla en un tablero constituido por 19 líneas horizontales y 19 líneas verticales, en las cuales se podrán ubicar fichas negras y blancas de a una por turno. Una vez ubicada una ficha no podrá moverse salvo que sea capturada por una ficha del jugador (color) oponente.

Se considera que *una ficha está atrapada* cuando en las posiciones adyacentes (vertical y horizontal) se encuentra una ficha del color contrario o se encuentra un borde del tablero. Esta descripción abarca las siguientes situaciones:

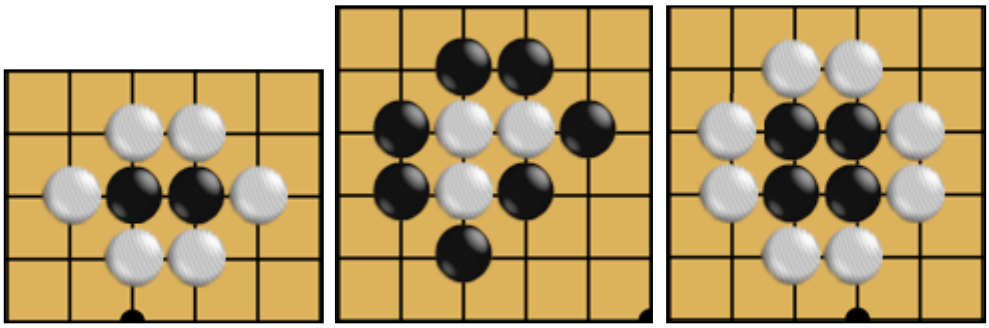
- Rodeada por fichas del color oponente.



- Rodeada por fichas del color oponente y por los bordes del tablero.



También es posible que existan un *conjunto de fichas atrapadas* (no vacío), y esto sucede cuando dicho conjunto verifica que cada piedra del mismo está rodeada por piedras del color contrario y/o bordes del tablero, o por piedras del mismo color que también pertenezcan al conjunto. Notar que el caso mencionado anteriormente (una piedra atrapada) representa un caso especial de un conjunto de fichas atrapadas donde el tamaño del conjunto es 1. Nuevamente se presentan imágenes representativas de esta posible situación de juego:

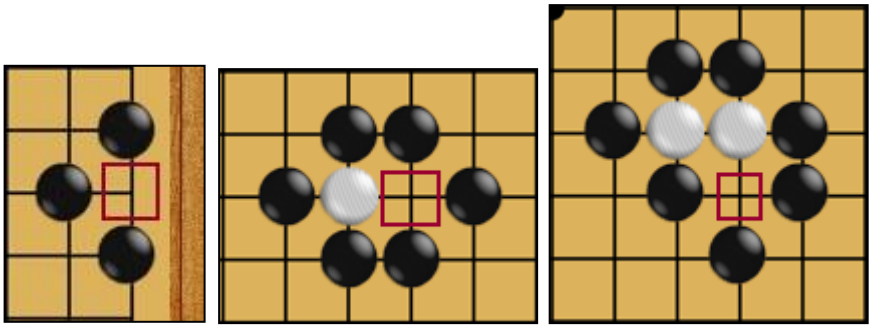


En la primer y última imagen, están atrapadas el grupo de fichas negras, y en la segunda imagen, el grupo de fichas blancas.

Jugadas inválidas

Se considerarán jugadas inválidas aquellas en las cuales el resultado de ubicar una ficha constituye suicidio o eternidad (eternity o ko, en el juego original).

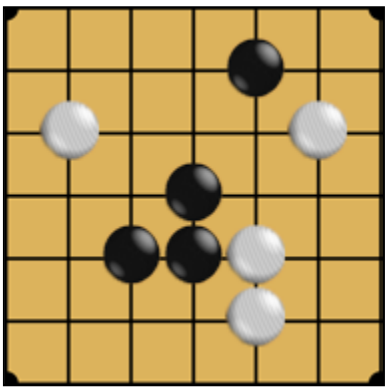
- Una posición constituye suicidio si al ubicar una ficha en dicha posición se genera un conjunto de piedras atrapadas del mismo color, sin haber piedras contrarias atrapadas. Por ejemplo, las imágenes mostradas previamente nunca serán configuraciones de tablero. Para ilustrar mejor esta regla, se muestran posibles situaciones en las cuales no se podrán agregar fichas blancas en la posición apuntada por el marco rojo:



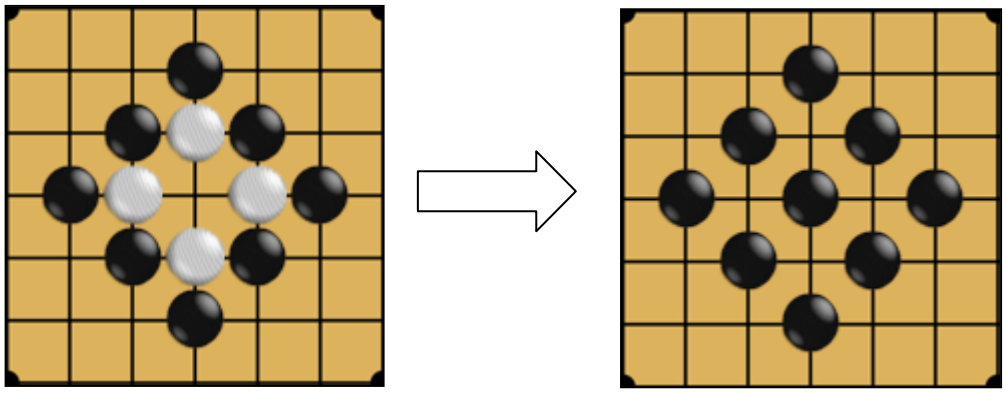
Jugadas válidas

Una vez descriptas las jugadas inválidas, la descripción de las jugadas válidas es trivial. Una jugada es válida si:

- Se coloca una piedra de tal forma que ésta no queda rodeada.



- Se coloca una piedra de tal forma que se genera un conjunto de piedras rodeadas del mismo color pero, a su vez, se captura al menos una piedra del color contrario.



- Se coloca una piedra en una posición en la cual no se comete suicidio, no esté fuera del tablero, o no sea inválida por generar situación de eternidad.

Decisiones generales de implementación.

- Un aspecto importante de mencionar es el grado de modularidad desarrollado que, además de brindar prolijidad y legibilidad al código, permitió documentar adecuadamente la funcionalidad y el propósito de cada predicado.
  Se ha evitado lo más posible el uso de predicados predefinidos como el **not** y el **fail**, excepto en ciertos casos en los que nos vimos sin otra alternativa que emplearlos. Por otro lado, se ha aprovechado el uso del operador de corte para hacer más eficiente el programa.

Decisiones Específicas de implementación.

En el análisis del juego hemos detectado que éste simplemente se lleva a cabo mediante una sola acción: inserción de fichas, con las restricciones ya establecidas en la primera sección del informe. Además, esta acción puede tener una sola consecuencia: capturar fichas del oponente.

La cátedra brinda 3 métodos que se deben usar para la implementación del juego:

emptyBoard: Brinda un tablero compuesto por una lista de listas como elementos.

goMove: Dado un tablero Board, el color Color y una posición [Fila, Columna], se retorna el tablero RRBoard con una nueva ficha colocada en la posición especificada.

Replace: Este predicado se aplica en dos etapas, en primer lugar para obtener la fila buscada luego con el segundo replace se modifica esta fila en la posición de la columna deseada y se inserta esta fila modificada en el tablero.

A partir de esto se ha modificado en Prolog el predicado *goMove* ya dado por la cátedra, que se encarga de llevar a cabo la acción de insertar una ficha, verificando que esto realmente pueda ser posible. Esto se hace llamando al predicado capturadas que dado un tablero Board y una posicion [X,Y] verifica si la ficha de color Color encierra a sus adyacentes. Si ocurre esto devuelve un tablero sin esas fichas encerradas, caso contrario, devuelve el tablero recibido originalmente.

Luego se llama al predicado haySuicidio que identifica que efectivamente no se haya roto esta regla. El procedimiento llevado a cabo en su implementación recibe como argumentos el tablero y los datos de la ficha insertada y controla si la ficha recibida está atrapada por fichas de color opuesto, esto se hace utilizando el predicado estaAtrapada que consta de 4 casos:

- Se encuentre una ficha del color opuesto,
- Sea una posición no válida (es decir, que se encuentre algún borde del tablero),
- Se encuentre una ficha del mismo color que haya sido visitada, es decir, que pertenece a la lista de visitadas.
- Se encuentre una ficha del mismo color que no haya sido visitada, y entonces, se verifica si la ficha en cuestión está también atrapada.

Predicados implementados.

Una vez introducidos los principales predicados que controlan la lógica del juego, presentamos los predicados implementados en Prolog: los que ya hemos presentados y otros auxiliares. Para cada uno de ellos listaremos sus argumentos, si se utilizan para la entrada o salida de información, y qué predicados utilizan para realizar su función.

Predicados principales

goMove(+Board, +Player, +Pos, -RBoard)

Descripción: RBoard es la configuración resultante de reflejar la movida del jugador Player en la posición Pos a partir de la configuración Board.

Predicados que usa:

- replaceBoard/6
- capturadas/5
- haySuicidio/4

replaceBoard(+Actual ,+Board, +X, +Y, +Color, -RBoard)

Predicados que usa:

- replace/5

Descripción:
Utiliza el replace que provee la cátedra para reemplazar una ficha color Actual por una ficha color Color en la posición [X,Y] del tablero Board y devuelve el tablero modificado en RBoard.

replace(?X, +XIndex, +Y, +Xs, -XsY)

Descripcion:

Este predicado se aplica en primer lugar para obtener la fila buscada luego con el segundo replace se modifica esta fila en la posición de la columna deseada y se inserta esta fila modificada en el tablero.

haySuicidio (+Board, +X, +Y, Color)

Predicados que usa:

- estaAtrapada/7
- colorOpuesto/2

Descripción: Dado un tablero Board, verifica si en la posición X e Y, la ficha de color Color queda en situación de suicidio.

capturadas(+Board,+ X,+ Y, +Color,- RBoard)

Predicados que usa:

- rodeada/6
- colorOpuesto/2

Descripcion: Dado un tablero Board y una posicion [X,Y] verifica si la ficha de color Color encierra a sus adyacentes. Si ocurre esto, devuelve un tablero sin esas fichas encerradas, caso contrario, devuelve el tablero recibido originalmente.

Predicados auxiliares

rodeado(+Board, +X, +Y, +ColorAtrap, +ColorAdy, -RBoard)

Predicados que usa:

- estaAtrapada/7

Descripción: Método cascara de capturadas que devuelve un tablero modificado si hubo encierro de la ficha de ColorAtrap por fichas de color ColorAdy, en caso contrario devuelve el mismo tablero recibido originalmente.

estaAtrapada (+**Board, +X, +Y, +ColorAdy,+ ColorAtrap, +ColorRemp, -RBoard)**

Predicados que usa:

- posNoValida/2
- replaceBoard/6

Descripción: Dado un tablero, una posición [X,Y] ,verifica si la ficha de color ColorAtrap está encerrada por fichas de color opuesto recibido como ColorAdy, y la reemplza por la ficha de color ColorRemp. Se considera como límite de encierro las fichas de color opuesto al recibido y también los bordes del tablero.

posNoValida(+X,+Y)

Descripción:

- Controla si los valores de X y de Y están en el rango (0, 19) (sin incluir los extremos).

colorOpuesto(+Color, -ColorOp)

Descripción:

- Devuelve el color contrario al recibido, es decir, si se recibe “blanco” devuelve “negro”, y viceversa.

Score(+Board,-White,-Black)

Predicado que usa:

- countC/3
- llenarY/5

Descripción:

Recibe un tablero Board como argumento y se encarga de contar las fichas de color Blanco y de color Negro de ese tablero recibido, por ultimo devuelve el puntaje final en White y Black.

countC(+Board,+color,-puntaje)

Predicado que usa:

- countR/3

Descripción:

Dado un tablero Board y la ficha de color Color recorre las columnascontando las fichas de ese Color dado para devolver el puntaje determinado.

countR(+Board,+color,-puntaje)

Descripción:

Dado un tablero Board y la ficha de color Color recorre las filas contando las fichas de ese Color dado para devolver el puntaje determinado.

llenarY(+Board, +X, +Y, +Color, -Board)

Predicado que usa:

- llenarX/5

Descripción: Dado un tablero y una posición [X,Y] recorro las columnas y coloco fichas del Color recibido en las zonas vacias encerradas por ese mismo Color.

llenarX(+Board, +X, +Y, +Color, -Board)

Predicado que usa:

- llenarAux/5

Descripción: Dado un tablero y una posición [X,Y] recorro las filas y coloco fichas del Color recibido en las zonas vacías encerradas por ese mismo Color.

llenarAux(+Board, +X, +Y, +Color, -Board)

Predicado que usa:

- estaAtrapada/7

Descripción: Metodo auxiliar de llenarX que se encarga de colocar fichas del Color recibido en las zonas vacías encerradas por ese mismo Color.

INTERACCION PROLOG-WEB

La parte grafica del juego, se realizo en 3 partes: CSS, HTML Y JAVASCRIPT

Esto ya lo dio implementado la cátedra, solo se debió modificar la parte de javascript para que al no haber más movimientos disponibles por parte de ambos jugadores se termine el juego, se devuelva un puntaje y se pueda volver a jugar nuevamente.

La partida se da por finalizada cuando se detecta que el botón Pasar se presiona dos veces seguidas por parte de los jugadores, por lo tanto al ocurrir esto la variable PassTurn se hace True y se llama a la funcion que creamos llamada gameOver() que activa el flag en “gameover” (anteriormente en “ficha”) y se llama a la funcion de Prolog Score que es la encargada de calcular los puntajes de los jugadores. Luego de esto, al llamarse satisfactoriamente al metodo HandleSuccess() , como se detecta que esta activado el flag “gameover” se calculan los puntajes de cada jugador y se asigna el ganador de la partida, en caso de haber un empate se avisa que se empató y no hay ningún ganador. Por último, se reinicia la partida llamando nuevamente al método HandleCreate().