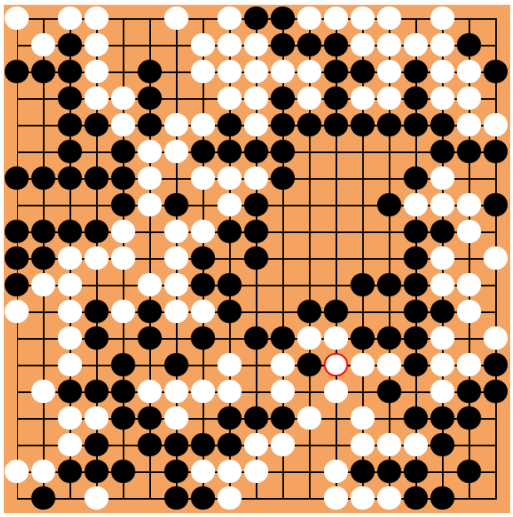
LÓGICA PARA
CIENCIAS DE LA COMPUTACIÓN
2019

Proyecto N1: GO



Profesor: Marcelo Falappa

Asistente: Mauro Gómez Lucero

INTEGRANTES

Zarate Tomas – LU 111365

Andrade Sergio – LU 114059

UNIVERSIDAD NACIONAL DEL SUR

Tabla de contenido

DESCRIPCION GENERAL DEL SOFTWARE **3**

Introducción 3

Método de juego 3

Atari y suicidio 3

DESCRIPCION DE LA IMPLEMENTACION **5**

DOCUMENTACION PARA USUARIOS TECNICOS **6**

DOCUMENTACION PARA USUARIOS FINALES **8**

DESCRIPCION GENERAL DEL SOFTWARE

Introducción:

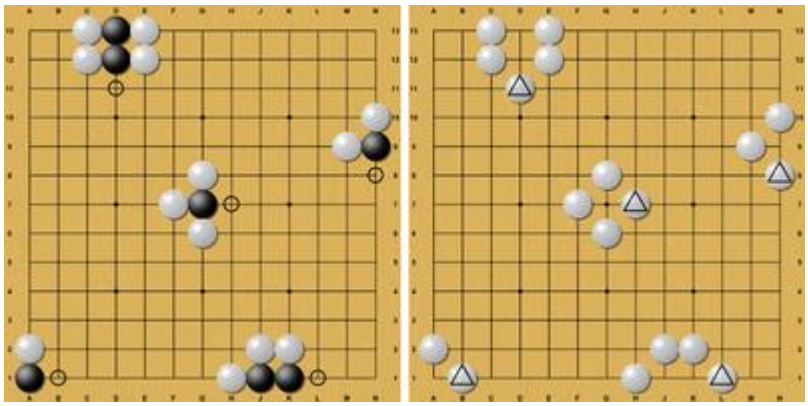
El Go es conocido por ser el juego de mesa más antiguo que aún perdura. Su orígen se remonta a más de 2000 años en China y, a pesar de haber pasado por los tamices de diferentes culturas y de miles de años, aún hoy día mantiene las reglas básicas que hacen al significado detrás de su nombre “juego de encerrar”. El juego se basa en un tablero de 19 filas por 19 columnas donde, turno a turno, enfrenta a dos jugadores, uno en poder de fichas blancas y el otro en poder de fichas negras, para conseguir la mayor cantidad de “territorio”.

Método de Juego:

El Go es un juego muy simple en su esencia pero muy complejo de dominar. El objetivo del juego es poder encerrar la mayor cantidad de “territorio” encerrado al finalizar la partida mediante jugadas de encierro sobre terreno vacío (es decir, posiciones del tablero) o bien sobre las fichas del oponente, al momento de finalizada la partida. Una posición del tablero se considera encerrada si en sus adyacentes (Norte, Sur, Este y Oeste) se encuentran 4 piezas de un mismo jugador o bien otra posición a su vez encerrada. Cada jugador puede colocar una ficha por turno en el tablero y con esto encerrar territorio o bien encerrar una o más fichas del jugador contrario; con dicha maniobra el jugador “captura” la(s) ficha(s) del contrario y la(s) retira del tablero, apoderándose del terreno donde esta(s) se encontraba(n) y el juego concluye cuando ambos jugadores saltean su turno de forma consecutiva. En caso de empate, se da la victoria al jugador blanco por haber tenido un turno menos de ventaja.

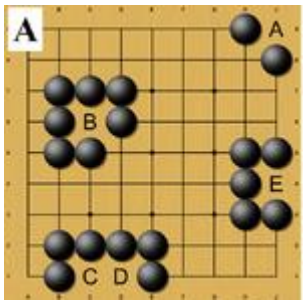
Atari y suicidio:

El “Atari” se define como una situación en la que el jugador favorecido podría capturar, o bien territorio, o bien fichas enemigas, con su siguiente movimiento. Sería recomendable prevenir las situaciones de “atari” del contrario.



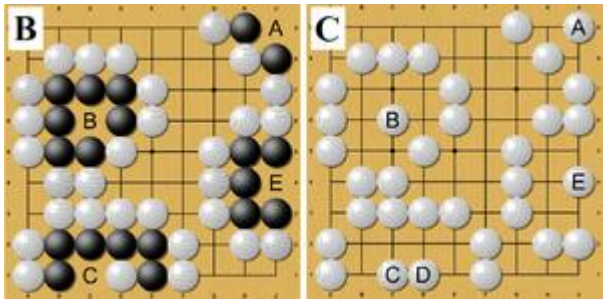
Todas las piedras negras están en atari. Las blancas pueden capturar las piedras negras.

El suicidio, como podría intuirse, significa sacrificar de manera deliberada una pieza al momento de jugarla, esto se considera una jugada inválida y el jugador que la realizó deberá de reposicionar dicha ficha evitando el suicidio.



(A) Blanco no puede mover a *A*, *B* o *E* pues sería suicidio. Sin embargo, *C* y *D* serían malos movimientos, aunque legales.

Sin embargo, hay una situación en la cual el "suicidio" se encuentra permitido, esto se da si, al momento inmediato de colocar la ficha, se estuviera capturando algunas de las fichas del contrario que encierran a la propia, evitando así el suicidio y convirtiendo la jugado en una válida.



(B) Blanco puede mover a *A*, *B*, *C* o *E* porque ha rodeado completamente al enemigo. (C) Blanco captura las piedras enemigas.

Reglas utilizadas: (Japonesas)

https://es.wikipedia.org/wiki/Go#Reglas_del_go

DESCRIPCIÓN DE LA IMPLEMENTACIÓN

Para implementar la parte lógica del software se utilizó el lenguaje de programación Prolog, asociado a una interfaz gráfica web empleando los tres lenguajes básicos de desarrollo web del lado del cliente: HTML, CSS y JavaScript.

El juego comienza con un tablero vacío de 19x19 celdas implementado en Prolog mediante el predicado emptyBoard/1 y almacenado en un objeto en JavasCript. Siempre inicia el jugador con fichas negras.

A lo largo del programa, para corroborar condiciones y realizar movimientos de eliminación, se optó por representar las fichas mediante ternas de [Ficha, Fila, Columna].

Todas las funciones referidas a condiciones para los movimientos de los jugadores se encuentran implementadas mediante predicados en Prolog, mientras que las funciones en JavaScript actualizan la interfaz visual, y envían consultas al servidor de Prolog para realizar un movimiento, o marcar el final del juego y mostrar el resultado.

Para realizar un movimiento se llama al predicado goMove/4

Para verificar que una jugada sea válida, el programa consulta si se está encerrando fichas del jugador contrario mediante encerradosContrario/7, o en su defecto si solo no se está cometiendo suicidio mediante encerrados/7.

Una ficha se considera encerrada si todas sus adyacentes son del color opuesto, o si sus fichas adyacente del color propio se encuentran encerradas.

Si una ficha resulta encerrada durante una jugada, debe ser eliminada mediante el predicado eliminarEncerrados/3 tomando como entrada una lista de fichas encerradas y el tablero actual, recorriendo el tablero para cada ficha encerrada y reemplazandola por un espacio vacío.

La partida finaliza cuando los dos jugadores pasan el turno de manera consecutiva, y se procede a calcular el puntaje de cada jugador.

Mediante el predicado terminarJuego/3 que retorna el puntaje de cada jugador, se procede a separar el tablero en tres listas, una con los espacios vacíos, otra con las fichas blancas, y otra con las fichas negras, a través de sucesivas llamadas al predicado aplanarTablero/4, que retorna una lista con todas las fichas en el tablero que coincidan con la entrada del predicado. Una vez obtenidas las listas, se reparten los espacios vacíos entre los dos jugadores mediante dos llamadas al predicado contarPuntos/5, que recibe la lista de espacios vacíos y un jugador, y añade el espacio vacío a una lista si este se encuentra encerrado por el jugador ingresado. Luego se suman los espacios vacios y la cantidad de fichas en el tablero de cada jugador para obtener el puntaje final.

DOCUMENTACIÓN PARA USUARIOS TECNICOS

A continuación, se describen detalles acerca de funciones implementadas, relaciones entre funciones y relaciones entre interfaces.

Implementación en Prolog,

Funciones y subrutinas:

- emptyBoard(Board): este predicado se encarga de devolver en la variable pasada por argumento un tablero vacío que será la base para el juego.
- goMove(Board, Player, Pos, RBoard): este predicado es llamado cada vez que un jugador coloca una ficha y es el encargado de, mediante los predicados auxiliares, actualizar la estructura del tablero que recibe como parámetro en Board y retorna el resultado en RBoard.
- replace(X, XIndex, Y, Xs, XsY): mediante el uso del contador XIndex este método reemplaza en una posición dada el elemento X por el elemento Y.
- getElem(X,XIndex,XsY): mediante el uso del contador XIndex busca un elemento X y lo vincula a la variable Y.
- adyacentes(Board, Fila, Columna,Adyacentes): mediante una posición Fila, Columna devuelve una lista de adyacentes en Adyacentes respecto de esa posición en el Board.
- obtenerAdyacente(Board, Fila, Columna,Adyacente): hace uso del predicado getElem para obtener el adyacente.
- encerrados(Board, Player, R, C,Adyacentes,Vistos,Encerrados): dado un color de ficha Player y una posición R (fila) y C (columna) permite obtener los elementos encerrados de ese color en formato lista en Encerrados.
- encerradosContrario(Board, Player, R, C,Adyacentes,Vistos,EncerradosContrarios): dado un color Player y una posición R (fila) y C (columna) permite obtener en EncerradosContrarios los elementos encerrado del otro color.
- encerradosVacios(Board, Vacio, Player, R, C, Adyacentes, Vistos, EncerradosTerreno): en base a una posicion R (fila) y C (columna) obtiene los vacios, encerrados por Player en Board, que se devuelven en EnecerradosTerreno
- eliminarEncerrados(RBoard, ListaEncerradosContrarios, NBoard): mediante un lista de elementos ListaEncerradosContrario toma RBoad y devuelve en NBoare el resultado de eliminar de Board el contenido de ListaEncerradosContrario.
- aplanarTablero(Board, Ficha,18, ListaFichas): devuelve el tablero Board como una lista plana de ternas Ficha, Fila y Columna en ListaFichas.
- obtenerTernas(Player,Fila, F, IndexAux, Lista): en base a una Fila y un incide F busca las ternas Player, Fila y Columna que luego devuelve en Lista.
- terminarJuego(Board, PuntosW, PuntosB): este predicado es invocado únicamente al finalizar el juego y, en base al Estado actual del Board, calcular los puntos del jugador blanco (PuntosW) y del jugador negro (PuntosB).

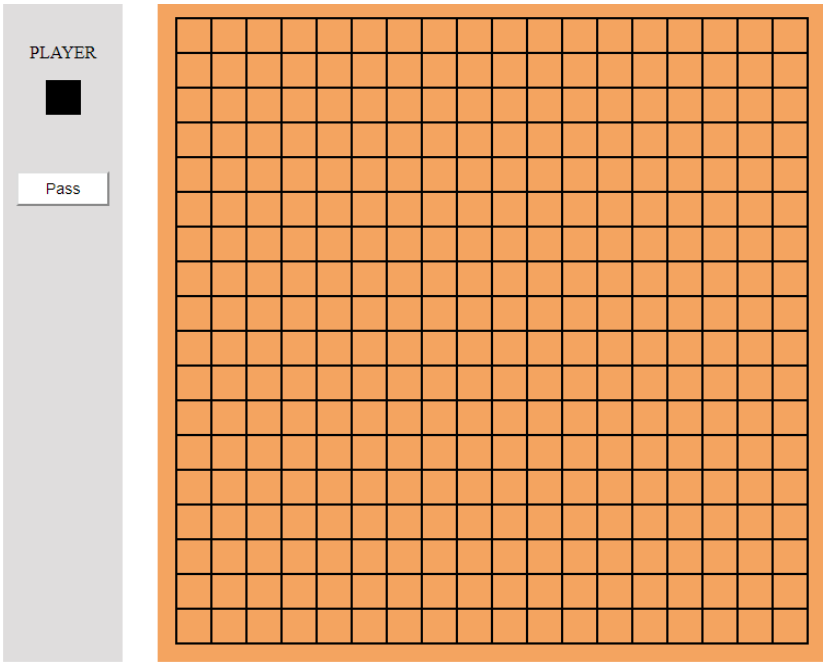
- contarPuntos(Board, Player, Vacios, Vistos, EncerradosPlayer): mediante un Board, un Player y una lista de Vacíos, devuelve en EncerradosPlayer los elementos que dicho jugador encierra estando vacíos.

La interfaz gráfica implementada en HTML y modelada en CSS cuenta un botón de pasar turno el cual actualiza el turno del jugador actual y cuenta cuantas veces consecutivas se pasó el turno, y un tablero visual seleccionable el cual al ser presionado consulta al archivo en JavaScript el cual deriva la consulta en la librería pengine de Prolog.
Los predicados Prolog consultados por JavaScript son emptyBoard/1, goMove/4 y contarPuntos/3.

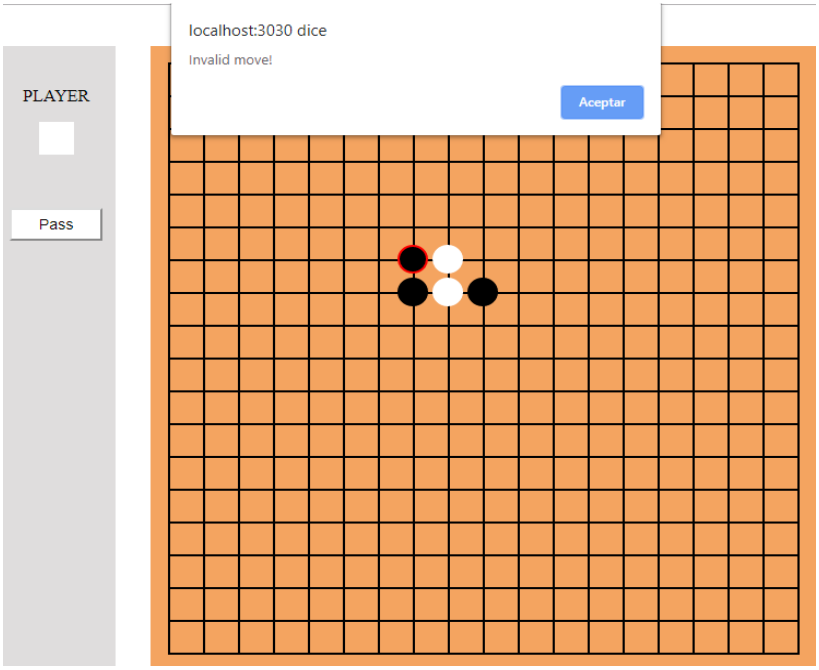
La librería emite una respuesta que es procesada por JavaScript modificando la interfaz gráfica.

DOCUMENTACION PARA USUARIOS FINALES

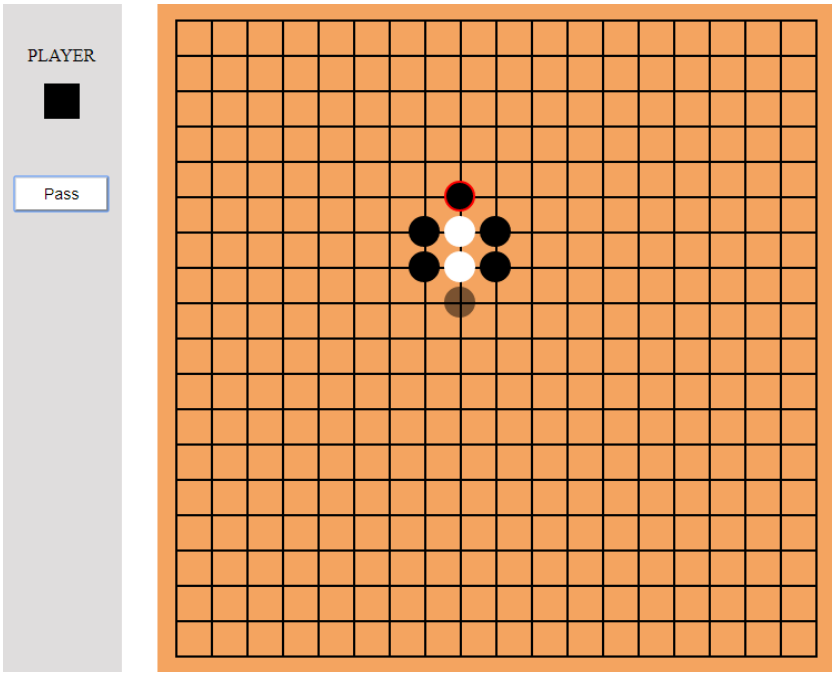
El juego consta de un tablero seleccionable y un botón que permite pasar de turno

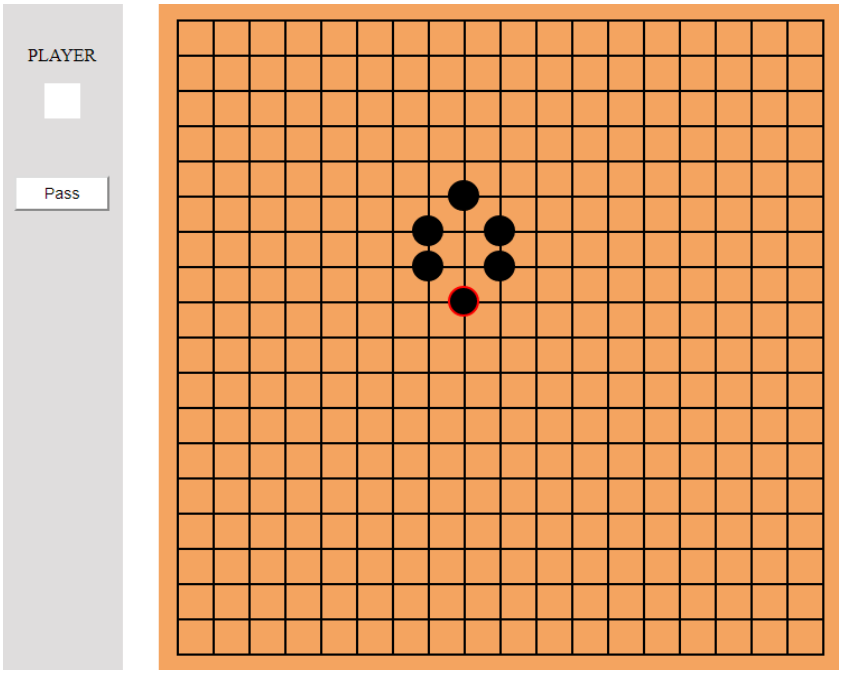


Comienza jugando el jugador con fichas negras, una vez hecho el movimiento pasa el turno automáticamente al jugador blanco. En caso de realizarse un movimiento inválido se informará mediante un cartel de alerta.

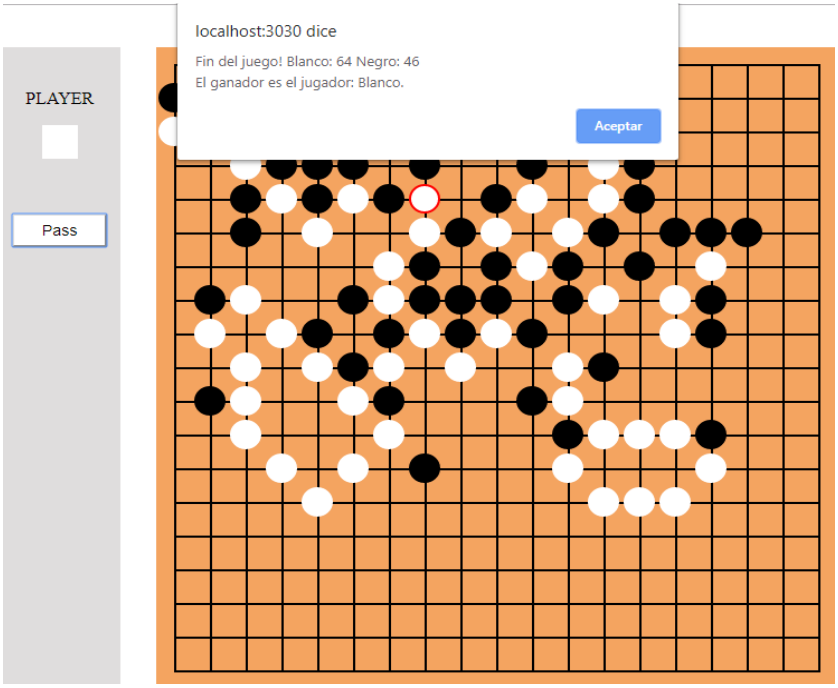


Cuando un jugador se encuentre en posición favorable de atari y efectúe la captura, la lógica automáticamente eliminará las fichas capturadas





El juego termina cuando ambos jugadores deciden que no pueden mejorar su posición en el tablero respecto del otro jugador y pasan de forma consecutiva, en este punto se procede a realizar el cálculo del puntaje, deberá esperar unos segundos hasta que el resultado sea computado.


PLAYER
Pass
localhost:3030 dice
Fin del juego! Blanco: 64 Negro: 46
El ganador es el jugador: Blanco.
Aceptar