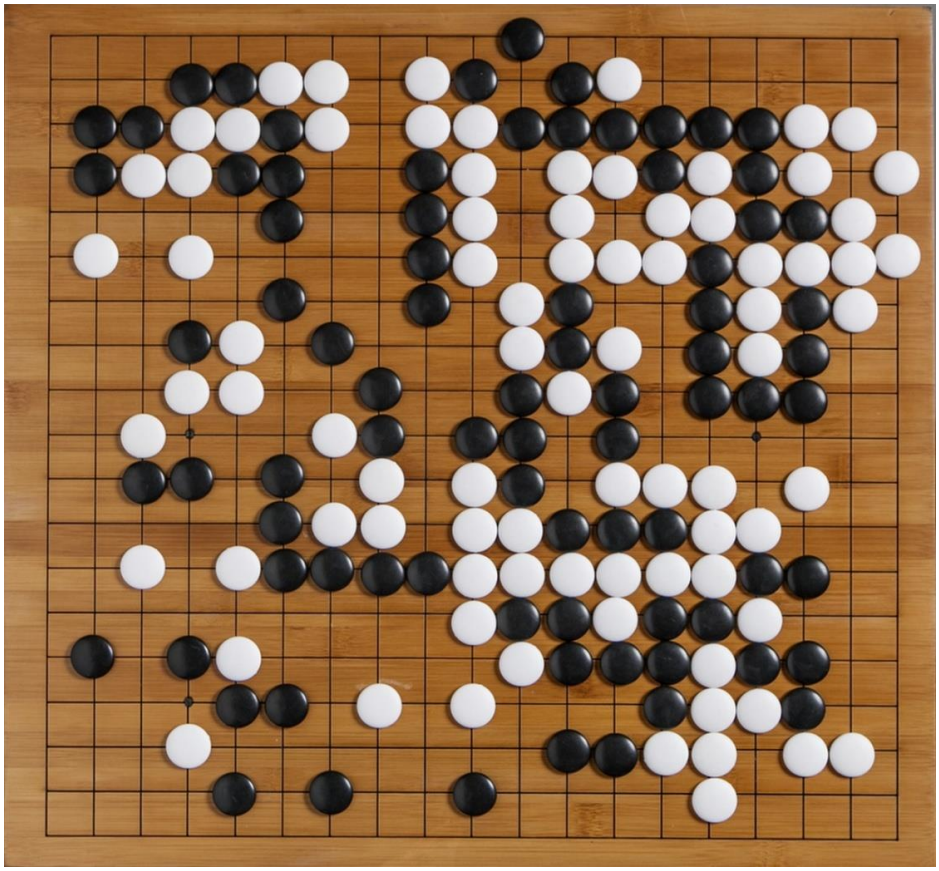
Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur

Informe Proyecto Nº1: "GO"

Lógica Para Ciencias de la Computación

Primer Cuatrimestre 2019

Docente Designada: Dra. Andrea Cohen



Integrantes de Comisión:

- **Fabio Campetti - L.U.: 114.761**
- **Francisco Larrasolo - L.U.: 94.195**

Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur

Introducción

En el siguiente informe se plasma la documentación del Proyecto 1 de la cátedra de Lógica para Ciencias de la Computación 2019, el cual consta en, a partir de un código base entregado por la cátedra, llevar a cabo una implementación para lograr la jugabilidad a partir de una interfaz web, también dado por la cátedra , del juego chino conocido como GO.
El Proyecto se divide en dos partes, una interfaz web escrita en JavaScript en el archivo "go.js", y la lógica del juego escrita en Prolog en el archivo "proylcc.pl".

Descripción del Juego y conceptos sobre el mismo

GO es un juego de mesa para dos jugadores que consiste de un tablero de 19 líneas horizontales (filas) por 19 líneas verticales (columnas) y piezas llamadas piedras (fichas) de color blanco y negro.
El Tablero se encuentra vacío al comienzo del juego, y los Jugadores, llamados Blanco y Negro, deben colocar piedras del color correspondiente, por turnos, en las intersecciones de las líneas verticales y horizontales. El jugador negro realiza la primer movida, luego de lo cual blanco y negro alternan. Gana el juego quien consigue adquirir mayor Territorio[1].
Existe la posibilidad en el transcurso del Juego, de eliminar piedras del rival al encerrar una o un grupo de las mismas en las intersecciones adyacente a la piedra o grupos de piedras pertenecientes al oponente.

Conceptos Útiles para comprender la descripción del código

Tablero: Tablero de juego representado como una lista de Filas, donde cada Fila es una lista de Columnas, donde a su vez cada columna contiene el elemento almacenado en la iésima Fila y la iésima Columna.

Movida: Una movida consiste en ubicar una piedra del color propio en una intersección vacía del tablero.

Ficha Capturada: Una vez que una piedra es ubicada ya no puede moverse, pero puede ser capturada como resultado de una jugada del jugador contrario, al tener una piedra del color del contrario en cada una de sus posiciones adyacentes (sin diagonales).

Consideraciones:
-Dos posiciones del tablero son **adyacentes** si coinciden en la fila y se encuentran en columnas contiguas (adyacentes horizontales), o coinciden en la columna y se encuentran en _las contiguas (adyacentes verticales).

-luego de colocar una piedra, podría existir más de un conjunto de piedras del oponente rodeadas, y todas deberían ser capturadas.

Fichas Rodeadas (o encerradas): Al colocar una piedra, todas aquellas piedras del oponente que queden rodeadas (o encerradas) por piedras propias serán capturadas, esto es, removidas del tablero.

Suicidio: Diremos que colocar una piedra de color C en una intersección dada constituiría suicidio si como resultado de colocarla en dicha posición se generará un conjunto de piedras rodeadas de color C (por supuesto, entre las cuales estará la piedra colocada), salvo que piedras contrarias también queden atrapadas, en cuyo caso no estaríamos en presencia de suicidio. Un jugador podrá hacer una movida válida al colocar una piedra en cualquier intersección vacía, con la restricción de que al colocarla en dicho lugar no cometa suicidio.

Territorio[1]: número de intersecciones que quedaron ocupadas por piedras propias, además de aquellas intersecciones vacías que quedaron encerradas o rodeadas por piedras propias.

Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur
De PROLOG

La parte lógica del desarrollo del Juego se divide en dos partes que sirven a dos propósitos distintos, la primera remite al desarrollo del Juego y una segunda parte que resuelve la conclusión del mismo a partir de la situación de "Fin del Juego".
La primera parte consta de la definición de predicados que comprenden los conceptos de: tablero vacío, jugada válida, definición de adyacentes y colores opuestos; obtención, reemplazo y eliminación de piedras en una posición determinada del tablero y el concepto de una o más piedras "capturadas o encerradas".
La segunda parte, por otro lado, consiste en el recuento del concepto de "territorio" (explicado con anterioridad), a partir de la consideración primero de las fichas negras, y luego lo mismo para las blancas.

Explicación y Detalle de Predicados

Predicado "emptyBoard/1"

Propósito: Retornar un Tablero de juego vacío

emptyBoard(...).

Parámetros de Entrada: Sin Parámetros

Parámetros de Retorno: Un único parámetro que retorna un Tablero vacío

Hechos "opuesto/2"

% **opuesto(+Color1, -ColorOp)**

Propósito: Definen el color opuesto de un primero pasado por parámetro.

Hechos "adyacente/2"

% **adyacente(+Pos, -PosAdy)**

Propósito: Definen cada una de las cuatro posibles posiciones adyacentes a una primera pasada por parámetro, controlando que no se caiga de los límites del Tablero**.**

Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur

Predicado "goMove/4"

Propósito: Resolver un intento de jugada por parte de un Jugador

% **goMove(+Board, +Player, +Pos, -NewBoard)**

Parámetros de Entrada:

-**Board**: Representación del Tablero al momento del intento de colocar una ficha (piedra)

-**Player**: Color del Jugador que intenta la jugada

-**Pos**: Posición como lista de dos elementos [R,C], donde R (row) es la fila y C (column) la columna como coordenadas en el Tablero en las que Player intenta la jugada.

Parámetros de Retorno:

-**NewBoard**: es la configuración resultante de reflejar la movida del jugador Player en la posición Pos a partir de la configuración Board y eventualmente eliminar fichas del oponente encerradas por esta jugada.

Estrategia:
Para resolver el intento de jugada por parte del Jugador, la estrategia tomada es primero recuperar la lista de adyacentes de la posición Pos en la que se intenta la jugada, luego el color opuesto al Jugador, para reemplazar (únicamente) el espacio vacío de Pos por una piedra del color Player. Esto se ejecutará correctamente, siempre y cuando no quede encerrada (jugada suicida), ó quede encerrada pero a su vez esté encerrando alguna/s ficha/s del jugador oponente. Por último eliminará todas aquellas fichas del jugador oponente que hayan sido encerradas por este último movimiento.

Predicados Auxiliares: ***adyacentes, opuesto, replace, encerrada, eliminarAds***

Predicado " adyacentes/2 "

Propósito: Predicado que obtiene la lista de todas las posiciones adyacentes a una Posición en el Tablero pasada por parámetro

% **adyacentes (+P, -L)**

Parámetros de Entrada:

- P: Posición de la cual se pretende obtener la lista de posiciones adyacentes.

Parámetros de Retorno:

- L: Lista que contiene todas las posiciones adyacentes a la pasada por parámetro.

Estrategia:

Al declarar como hechos los cuatro casos posibles de posiciones adyacentes a una posición genérica en una matriz, y agregándole controles para mantener dichas posiciones dentro de los límites del tablero definido (19x19), utilizamos el predicado predefinido *findall* para encontrar todas aquellas soluciones que no fallen para la posición-parámetro.

Predicados Auxiliares: ***adyacente***

Predicado "eliminarAds/5"

Propósito: En caso de que la jugada provoque que una o un grupo de piedras del rival son encerradas, las elimina.

% **eliminarAds(+Board, +Opuesto, +Player, +Adyacentes, -NewBoard)**

Parámetros de Entrada:

-**Board**: Representación del Tablero al momento del intento de eliminar una ficha (piedra).

-**Opuesto**: Color opuesto al del Jugador que intenta la jugada

-**Player**: Color del Jugador que intenta la jugada

-**Adyacentes:** Lista de piedras adyacentes

Parámetros de Retorno:

-**NewBoard**: es la configuración resultante de reflejar la movida del jugador Player en la posición Pos a partir de la configuración Board.

Estrategia:

Determina las adyacentes encerradas y elimina los conjuntos también encerrados al que estas pertenecen.

Predicados Auxiliares: ***encerrada, sort, eliminarVarias***

Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur

Predicado " eliminarVarias/4"

Propósito:

% **eliminarVarias(+Tablero, +Color, +ListaEliminables, -NuevoTablero)**

Parámetros de Entrada:

- Tablero: Representación del Tablero al momento del intento de eliminar un conjunto de fichas (piedras).

- Color: Color de las fichas a eliminar.

- ListaEliminables: Lista de fichas que deben ser eliminadas

Parámetros de Retorno:

- NuevoTablero: Nueva configuración del Tablero de Juego una vez que han sido eliminadas todas las fichas que se deseaban eliminar presentes en la lista pasada por parámetro.

Estrategia:

Primero se recorre la lista de eliminables recursivamente hasta el final, y al finalizar la recursión, se eliminan una a una las fichas que se desean eliminar invocando el predicado *eliminar/4*.

***Predicados Auxiliares:* eliminar**

Predicado "eliminar/4"

Propósito: Reemplazar una ficha de un Jugador determinado en el Tablero, en una posición determinada, por un espacio vacío.

% eliminar(+Board, +Player, +[R,C], -RBoard)

Parámetros de Entrada:

- **Board**: Tablero al momento de eliminar la ficha solicitada

- **Player**: Tipo/Color del Jugador de la ficha a eliminar

- **[R,C]**: Posición en el tablero (fila *Row* y columna *Column*), de la ficha a eliminar

Parámetros de Retorno:

- **RBoard**: Nuevo Tablero con la ficha a eliminar, eliminada.

Estrategia:

Doble invocación al predicado *replace*, donde a partir del primero se obtiene la fila del elemento a eliminar, y en la segunda invocación, por unificación de Prolog, se reemplaza el elemento en la posición exacta (fila , columna) por un "-" (intersección vacía).

***Predicados Auxiliares:* replace**

Predicado "replace/5"

Propósito: Reemplazar un elemento X en una lista por otro Y.

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

Parámetros de Entrada:

- XIndex: posición del elemento a reemplazar en la lista Xs

-Y: elemento que reemplazará al X-ésimo elemento de estar en la posición XIndex de la lista Xs.

-Xs: lista a recorrer para reemplazar uno de sus elementos en la posición XIndex, de estar presente, por el elemento Y.

Parámetros de Retorno:

-XsY: lista de retorno, cuyo contenido es la lista Xs pasada por parámetro con Y como Xésimo elemento, de haber estado X en la posición XIndex de la misma.

Estrategia:

Caso Base

En el caso de encontrar el X-ésimo (XIndex) elemento X de una lista, como cabeza de la lista con cola Xs, retorno la lista recorrida reemplazando el X-ésimo elemento por Y.

Caso Recursivo

En el caso de no estar en el X-ésimo elemento de una lista, salteo el elemento actual, reduciendo el caso recursivo restando 1 a XIndex, y realizando la llamada recursiva.

Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur

Predicado " iesimo/3"

Propósito:

% **iesimo (+Lista, +IesimaPos, -Elemento)**

Parámetros de Entrada:

- Lista: Lista en la cual se pretende buscar el iésimo elemento.

- IesimaPos: Iésima posición en la cual está el elemento buscado en la Lista

Parámetros de Retorno:

- Elemento: Elemento de la Lista pasada por parámetro, ubicado en la IésimaPos de la misma.

Estrategia:

Se llama recursivamente, IésimaPos cantidad de veces hasta llegar a la posición deseada en la Lista y se retorna el Elemento en dicha ubicación.

Predicado " recuperar /3"

Propósito: Dada una posición, recorrer el Tablero para recuperar el color del elemento de la x-ésima fila y la y-ésima columna.

% **(+Tablero, +Pos, -Color)**

Parámetros de Entrada:

- Tablero: Configuración del Tablero al momento de realizar una jugada.

- Pos: Posición del elemento buscado en el Tablero, representada como una lista de dos elementos (x,y), donde x corresponde al número de fila y el valor de y representa el número de columna.

Parámetros de Retorno:

- Color: Color de la ficha de la posición Pos en el Tablero, una vez recorrido y encontrada dicha ubicación en el mismo.

Estrategia:

Utilizando el predicado auxiliar *iesimo*, para recuperar el iésimo elemento de una lista, y el concepto de representación del tablero donde el mismo es una lista de filas, donde a su vez cada una de estas es una lista de columnas. En primer lugar se utiliza la coordenada de la posición X para encontrar la X-ésima fila, para luego recuperando dicha fila buscar el Y-ésimo elemento de dicha fila y devolver el elemento que corresponde a la posición Pos.

Predicados Auxiliares: ***iesimo***

Predicado "encerrada/6"

Propósito:

% **encerrada(+Tablero, +Posición, +ColorActual,+ColorEncierra, +Visitados, -ListaEncerrados)**

Parámetros de Entrada:

- Tablero: Representación del Tablero al momento chequear si la Posición pasada por parámetro está encerrada.

- Posición: Posición compuesta por una lista con coordenadas del tipo (fila,columna), de la cual se desea corroborar si está encerrada o pertenece a un conjunto de fichas encerradas.

- ColorActual: Color de la ficha o grupo de fichas que se desea chequear si están encerradas por ColorEncierra

- ColorEncierra: Color de las fichas que se pueden encerrar o encierran a la ficha de la Posición pasada por parámetro.

- Visitados: Lista de fichas del ColorActual que ya han sido visitadas en caso de ser varias encerradas, recorriendo las mismas a partir de Posición y sus adyacentes.

Parámetros de Retorno:

- ListaEncerrados: Lista de Posiciones en el Tablero de las fichas de ColorActual encerradas por ColorEncierra, controlando a partir de la Posición dada.

Estrategia:

En primer lugar se chequea que las posiciones adyacentes a la Posición actual sean de ColorEncierra . Caso contrario, se chequea que todas las adyacentes a la Posición actual sean

fichas de ColorActual o de ColorEncierra, en cuyo caso se buscan todas aquellas fichas potencialmente eliminables (conjunto de fichas del mismo color encerradas), tales que pertenezcan a la lista de adyacentes de la Posición actual, no hayan sido visitadas y sean del ColorActual. Chequeando que dicha lista de potencialmente fichas eliminables estén encerradas, marcando como visitada la Posición actual.

Predicados Auxiliares: ***adyacentes, forall, findall, member, recuperar, checkEncerradas***

Predicado "checkEncerradas/6"

Propósito: Determinar de una lista de posiciones cuáles de estas se encuentran encerradas y devolver el territorio encerrado al cual estas pertenecen

% **checkEncerradas(+Tablero, +Ads, +ColorAct, +ColorEncierra,+Vis,-Encer)**

Parámetros de Entrada:

-Tablero: Representación del Tablero al momento chequear si la Posición pasada por parámetro está encerrada.

-Ads: Lista de posiciones a chequear si están encerradas

-ColorAct: Color de las fichas en las posiciones de la lista “Ads”

-ColorEncierra: Color de las fichas que pueden encerrar o encierran a las fichas de las posiciones de la lista

-Vis: Lista de las posiciones ya visitadas del posible territorio encerrado.

Parámetros de Retorno:

-Encer: Lista de todas las posiciones pertenecientes al territorio encerrado.

Estrategia:

Recorre la lista de posiciones pasada, corroborando si cada una de las posiciones se encuentra encerrada por el ColorEncierra y siendo el territorio final a retornar el conjunto unión de cada uno de los territorios encerrados a los que pertenecen las posiciones.

Predicados Auxiliares: ***encerrada***

Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur

----------SEGUNDA PARTE (Fin del Juego)----------

Predicado "finDelJuego /3"

Propósito: Calcula los puntajes de ambos colores de fichas teniendo en cuenta la noción de Territorio

% **finDelJuego(+Tablero, -PuntajeNegras, -PuntajeBlancas)**

Parámetros de Entrada:

- Tablero: Configuración del Tablero de juego al finalizar la partida.

Parámetros de Retorno:

- Puntaje Negras: Puntaje de las fichas/piedras Negras.

- Puntaje Blancas: Puntaje de las fichas/piedras Blancas.

Estrategia: Calcula en primer lugar la cantidad de fichas y el Territorio de las fichas Negras, para luego teniendo en cuenta el Territorio ganado por las fichas negras (lugares vacíos *encerrados* conservados en una lista que contiene las Posiciones correspondientes a dicho Territorio), efectuar el mismo procedimiento para las fichas Blancas.
La estrategia descripta adoptada para el predicado contempla el caso de fin del Juego con un tablero vacío (sin jugadas hechas).

Predicados Auxiliares: ***contar***

Predicado "contar/6"

Propósito: Procesar las filas del tablero una a una calculando el puntaje del Color correspondiente.

% **contar(+Tablero, +Fila, +Color, +Visitados, +AVisitar, -Cant)**

Parámetros de Entrada:

- Tablero: Configuración del tablero correspondiente al final de la partida.

- Fila: Número de fila actual a procesar.

- Color: Color para el cual se debe calcular el puntaje.

- Visitados: Lista de Posiciones ya visitadas.

- AVisitar: Lista de Posiciones a visitar.

Parámetros de Retorno:

- Cant: Sumatoria total del puntaje asignado a las fichas de Color correspondiente.

Estrategia:

Recorre todas las filas del Tablero llamando recursivamente y comienza desde la última fila donde en todo momento el puntaje de las fichas del Color correspondiente, es igual a el puntaje recopilado de la fila actual, más el de las filas hasta llegar a la última del Tablero.

Predicados Auxiliares: ***contarEnFila, is***

Predicado "contarEnFila/7"

Propósito: Procesar una fila del tablero moviéndose columna por columna calculando el puntaje del Color correspondiente.

% **contar(+Tablero, +Fila, +Columna,+Color, +Visitados, +AVisitar, -Cant)**

Parámetros de Entrada:

- Tablero: Configuración del tablero correspondiente al final de la partida.

- Fila: Número de fila que se recorrerá.

-Columna: Numero de columna a procesar

- Color: Color para el cual se debe calcular el puntaje.

- Visitados: Lista de Posiciones ya visitadas.

- AVisitar: Lista de Posiciones a visitar.

Parámetros de Retorno:

- Cant: Sumatoria total del puntaje asignado a las fichas de Color correspondiente.

Estrategia:

Recorre una Fila del Tablero moviéndose por las columnas (es decir posición por posición) llamando recursivamente, comienza desde la última columna hasta llegar a la primera, donde en todo momento el puntaje de las fichas del Color correspondiente será una de la siguientes:

*0: la posición ya fue contada.

*1: la posición tiene una ficha del Color correspondiente.

*Terr: la posición se encuentra encerrada por el color correspondiente, siendo Terr la cantidad de posiciones también encerradas junto a la posición.

*0: la posición tiene una ficha de color opuesto al correspondiente.

Sumado a el puntaje recopilado de las otras posiciones recorridas a lo largo de la Fila.

Predicados Auxiliares: ***member, recuperar, encerrada, sort, append, length.***

Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur
De JAVASCRIPT

Para la parte de interfaz del Juego con el Usuario, la misma se genera a partir de código Javascript, en un principio dado por la cátedra, donde la misma consta de la creación del Tablero y el vínculo con el servidor Pengines (Prolog Engines), así como también el manejo de una jugada por parte del Usuario y el cambio de Turno del mismo una vez que se ha realizado una jugada o simplemente el jugador desea pasar el Turno y que juegue el rival. Para completar la funcionalidad requerida de la interfaz, fueron introducidas ciertas modificaciones al código con el propósito de implementar el Fin del Juego, y el resultado del mismo, dichas modificaciones serán explicadas a continuación.

Interacción JavaScript – Prolog

El funcionamiento de la Interfaz Gráfica del Juego, utiliza código definido en JavaScript y CSS para definir aspectos gráficos y de funcionalidad, que disparan llamados a la lógica a partir de la interacción con Prolog.
La forma en la que trabaja dicha interacción, es a partir de la definición de una variable que establece la conexión con el servidor (Pengines), y luego a partir de la asignación de oyentes a los elementos gráficos plasmados en la interfaz. Esto es, a partir de la variable de servidor se definen funciones asignadas a distintos eventos producidos por solicitudes al servidor (creación, éxito, falla), y por otro lado la asignación de oyentes a los elementos clickeables de la Interfaz.
La parte principal de la jugabilidad que dispara todo lo que tiene que ver con la lógica, está en la construcción inicial y actualización del tablero a medida que transcurren los turnos. Esto se produce porque en la construcción del tablero, a cada intersección se le asigna un oyente que ante el click de alguno de los Jugadores, invoca la función *handleClick* con sus coordenadas de fila y columna, a su vez dentro de dicha función, se le realiza una solicitud al servidor para ejecutar el predicado principal de la lógica en Prolog ***goMove*** que dispara todos los controles y posibles situaciones explicadas en la documentación de la lógica y mostrada en algunos casos de prueba.
En caso de producirse eliminaciones o colocaciones de nuevas fichas en el Tablero, se captura la nueva configuración del Tablero al finalizar ***goMove*** con éxito, dentro del evento ***onSuccess*** que produce una llamada a la función ***handleSuccess***.

Código Agregado al archivo "go.js"

FLAG AGREGADO COMO VARIABLE GLOBAL

```
var passedTurn = false;
```

Propósito: El propósito de utilizar un flag 'passedTurn' como variable global, es la identificación del primer click en el botón de Pasar Turno, para que luego del cambio de turno, de volver a ser clickeado dicho botón, se identifique la situación de dos turnos pasados en forma consecutiva, que representa la situación de Final del Juego.

Los valores que toma en distintas situaciones son verdadero luego de ser clickeado el botón anteriormente mencionado, y vuelve a falso luego de una jugada válida o al reiniciar el Juego luego de haber terminado una partida.

Su valor representativo es consultado únicamente dentro del oyente del botón de Pase de Turno.

MODIFICACIONES EN CUERPO DE FUNCIONES

- **EN FUNCION init()**

```
document.getElementById("passBtn").addEventListener('click', () => passedTurn? finDelJuego(): passTurn() );
```

Modificación en la reacción del oyente según el valor del 'flag' passedTurn, en caso de ser verdadero ejecuta un llamado a finDelJuego() para calcular los puntajes finales de la partida, sino a passTurn() para cambiar de turno y 'bajar' el flag.

- **EN FUNCION handleSuccess()**

Modificación cuyo propósito entra en acción al ejecutarse la consulta Prolog finDelJuego(...) al servidor, ya que se intenta recuperar los parámetros de retorno del mismo y de otra manera ante cualquier otra situación los mismos quedan indefinidos por no coincidir los nombres de las variables de retorno. En caso de resultar exitosa la situación de consulta y extracción de los valores de puntaje de cada Jugador, se invoca a la resolución final de la partida como consecuencia de mostrar el resultado a los Usuarios y la limpieza del Tablero para el comienzo de una nueva partida, caso contrario, el éxito de consulta que se pretende manejar es el de una movida por parte de un Jugador.

```
function handleSuccess(response) {

    gridData = response.data[0].Board;

    //Get data de fin del juego
    var black = response.data[0].Negro;
    var white = response.data[0].Blanco;

    if (black!=undefined && white!=undefined)
        mostrarYLimpiar(black,white);
    else{   ..Manejo goMove..
```

Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur

FUNCIONES AGREGADAS PARA COMPLETAR REQUISITOS DE FUNCIONALIDAD DE LA INTERFAZ Y SU VINCULO CON PROLOG

Función *passTurn()*

```
function passTurn(){
        passedTurn = true;
        switchTurn();
}
```

Propósito: El propósito de esta función es un paso intermedio antes de cambiar de Turno por un click en el botón de pasar turno (bypassear la función switchTurn), para 'levantar' el flag declarado al principio con el propósito de identificar la situación de Final del Juego.

Función *finDelJuego()*

Propósito: El único propósito de esta función es realizar la llamada al servidor para que evalúe el predicado Prolog 'finDelJuego' definido en la lógica del Juego, que al tener éxito, captura los parámetros de retorno de Prolog en el llamado a la función de JavaScript *handleSuccess* definido en el evento *onsuccess* del servidor Pengines.

```
function finDelJuego(){

    const s = "finDelJuego("+Pengine.stringify(gridData)+", Negro , Blanco)";

    pengine.ask(s);

}
```

Función *mostrarYLimpiar(puntosNegro,puntosBlanco)*

Parámetros de Entrada:

- **puntosNegro: puntaje de las piedras/fichas del color Negro**
- **puntosBlanco: puntaje de las piedras/fichas del color Blanco**

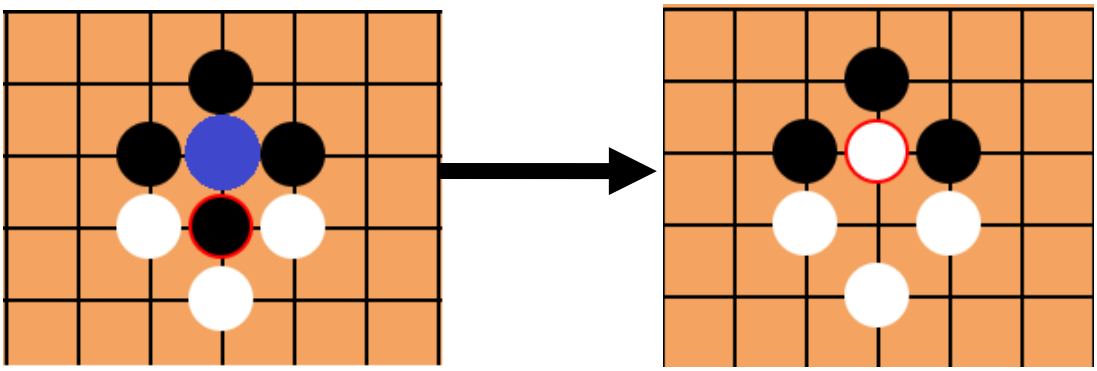
Propósito: El propósito de esta función es mostrar con un mensaje por pantalla el resultado de la partida jugada, para luego 'limpiar' tanto las variables globales utilizadas como 'flags', como así también el tablero de Juego para permitir jugar una nueva partida, donde comienza jugando el color Negro (producto de settear como falsa la variable global (flag) *turnBlack*).

```
function mostrarYLimpiar(puntosNegro,puntosBlanco){

    if (puntosNegro == puntosBlanco)
        alert("EMPATE\nNegro = "+puntosNegro+"\nBlanco = "+puntosBlanco);
    else{
        var ganador = (puntosNegro>puntosBlanco)? "Negro" : "Blanco";

        alert("GANADOR "+ganador+"\nNegro = "+puntosNegro+"\nBlanco = "+puntosBlanco);
    }

    //Limpio el tablero y comienza un juego nuevo
    turnBlack = false;
    passedTurn = false;
    pengine.ask('emptyBoard(Board)');
}
```

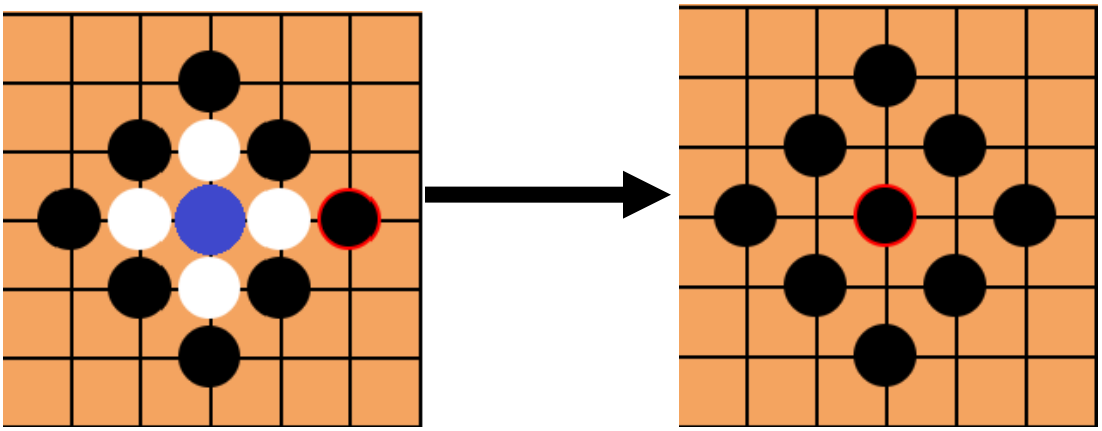
Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur

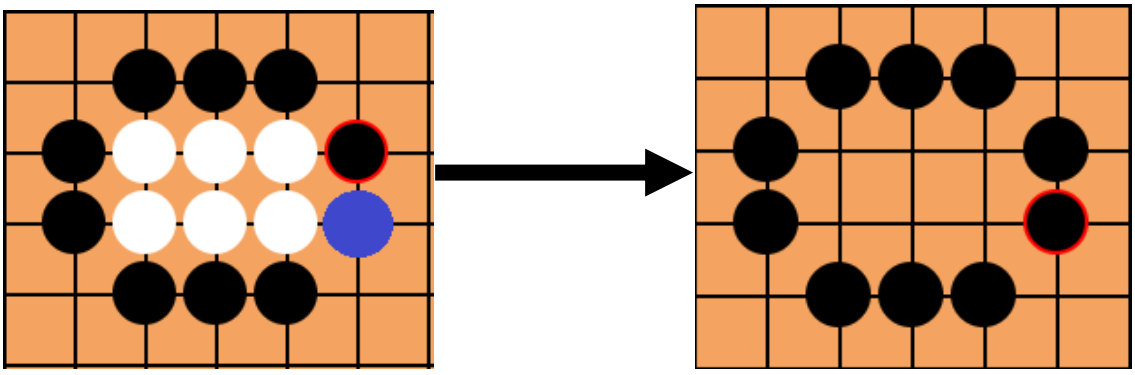
POSIBLES JUGADAS REALIZABLES

- En la siguiente situación si se encuentra en el escenario de la izquierda y se coloca una ficha de color Blanca en el lugar Azul se producirá el escenario de la derecha.



- En las siguientes situaciones si se encuentra en el escenario de la izquierda y se coloca una ficha de color Negra en el lugar Azul se producirá el escenario de la derecha.





- En la siguiente situación si se encuentra en el escenario de la izquierda y se coloca una ficha de color Blanca en el lugar Azul se mostrará un cartel de jugada invalida (“jugada sucia” o “suicidio”) en la parte superior de la pantalla como el del escenario de la derecha (siendo la jugada no permitida).