

Inteligencia Artificial 1

Proyecto 1	7 septiembre 2021
201504051	Asunción Mariana Sic Sor

Tabla de Contenido

- [Problema 1](#)
 - [Descripción del Problema](#)
 - [Solución](#)
 - [Hechos](#)
 - [Reglas](#)
 - [Culpable](#)
- [Problema 3](#)
 - [Reverso de lista](#)
 - [Palíndromo](#)
 - [Duplicar Lista](#)
 - [División en dos listas](#)
 - [Insertar en lista](#)
- [Problema 4](#)
 - [Descripción](#)
 - [Solución de Sudoku](#)
 - [Sudoku](#)
- [Referencias](#)

Problema 1

Descripción del problema

La policía de Gotham está investigando el asesinato a mano armada de Marta. Una madre que tenía 4 hijos, todos ellos con hijos sumando 9 nietos en total.

La hermana de el/la culpable tenía sospechas de dicha persona por lo que procedió a comentarlo con otros tres parientes que se encontraban junto a ella. Los parientes con los que comentó sus sospechas eran: el abuelo que se llama Bruce, el primo Clark y el tío Barry, los cuales también sospechaban de la misma persona por lo que le dijeron que llamara a la policía.

Finalmente, se detuvo a la persona sospechosa el mismo día del funeral de Marta donde solo estaban presentes los familiares directos de ellos y sus respectivos cónyuges.

De la familia se sabe lo siguiente:

- Ningún nieto tiene pareja.
- Nadie tiene el nombre repetido en la familia.
- Cada una de las 4 parejas en la familia tienen 2 hijos o más.

De lo acontecido durante el funeral de Marta se sabe lo siguiente:

- Barry se encontraba charlando con su cuñado llamado Enrique.

- Los 3 hermanos, Ezio, Lorenzo y Sergio, estaban sentados junto al ataúd de su abuela.
- Una de las hijas de Marta, Diana, estaba junto a su esposo Peter y sus 2 hijos, Mary y Harry.
- El yerno de Bruce, Ben, se encontraba consolando a su esposa May y, a su vez, vigilaba que sus hijos se comportaran bien.
- Clark se encontraba llorando solo mientras su hermana Lois hablaba con su prima Lara.
- Pepper estaba junto a su hijo primogénito Tony que se encontraba llorando por su pobre abuela.
- Bruce se encontraba junto a su hija Rachel.

¿Quién fue la persona culpable del asesinato?

Solución

Hechos

Los hechos más importantes fueron:

- Definir los hijos y las hijas

Con el hecho:

```
% si es hijo
hijo(hijo,progenitor_hijo) .
% si es hija
hija(hija,progenitor_hija) .
```

Los hijos se detallan a continuación:

Tabla I

Hijo/a	Padre/Madre
Diana,Rachel,May,Barry	Marta,Bruce
Mary,Harry	Diana,Peter
Clark,Lois	Rachel,Enrique
Ezio,Lorenzo,Sergio	May,Ben
Lara,Tony	Pepper,Barry

Se creo un hecho por cada hijo en la tabla anterior.

- Definir los padres

Se ha definido una regla para cada padre (incluyendo madre)

```
padre(padre_o_madre, [lista_hijos]) :- ! .
```

Las reglas se definieron según la [Tabla I](#).

- Definir las parejas

Se han definido las parejas con una regla de la siguiente manera:

```
% la persona 'p1' es pareja de la persona 'p2'
pareja(p1,p2) :- ! .
```

Las reglas para las parejas se crearon según la siguiente tabla:

Tabla II

Parejas
Marta y Bruce
Diana y Peter
Rachel y Enrique
May y Ben
Pepper y Barry

- Definir los hermanos

Se crearon los hechos:

```
% si es hermano
hermano(el_hermano, [lista_de_hermanos]) .
% si es hermana
hermana(la_hermana, [lista_de_hermanos]) .
```

Los hermanos se definieron para cada hijo descrito en la [Tabla I](#)

Reglas

- Para saber si es hermano

Para determinar si dos personas son hermanos, se crea la siguiente regla

```
esHermano(X,Y) :- hermano(X, Hermanos) , member(Y,Hermanos) ;
                  hermana(X, Hermanos), member(Y, Hermanos).
```

Es verdadero si la persona `X` está en los hechos de hermanos y su hermano `Y` pertenece a la lista de hermanos mediante el predicado `member` [3].

- Para saber si es hijo

Determina que dos personas tengan relación de padre/madre e hijo/hija con ayuda del hecho `hija` e `hijo` .

```
esHijo(Padre, Hijo) :- hijo(Hijo, Padre) ; hija(Hijo, Padre).
```

- Para saber si dos personas son primos/primas

Determina que `primo1` sea primo con `primo2` , para esto debe cumplirse que alguno de los papás de los primos deben ser hermanos:

```
primo(Primo1,Primo2) :-
    % busca el padre de 'Primo1' y lo almacena en 'Padre1'
    esHijo(Padre1,Primo1),
    % busca el padre de 'Primo2' y lo almacena en 'Padre2'
    esHijo(Padre2,Primo2),
```

```
% verifica que 'Padre1' y 'Padre2' sean hermanos
esHermano(Padre1, Padre2) .
```

- Para saber si es un sobrino

Determina que `Posible_tio` tenga como sobrino a `Posible_sobrino` verificando que el hijo del `tio` sea primo con el `sobrino` .

```
sobrino(Posible_tio, Posible_sobrino) :-
    % busca algun hijo del 'Posible_tio' y lo almacena en 'Hijo'
    esHijo(Posible_tio, Hijo),
    % verifica que el 'Hijo' encontrado sea primo con 'Posible_sobrino'
    primo(Hijo, Posible_sobrino).
```

- Para saber si es un tío

Verifica que la `Persona` tenga como tío a `Posible_tio` mediante la regla anterior de `sobrino`

```
tio(Persona, Posible_tio) :- sobrino(Posible_tio, Persona) .
```

- Para saber si es abuelo o abuela

Determina si `Abuelo` tiene como nieto a `Nieto`

```
esAbuelo(Abuelo, Nieto) :-
    % busca el padre de 'Nieto' y lo almacena en 'Padre'
    esHijo(Padre, Nieto),
    % busca que el padre de 'Padre' sea 'Abuelo' para que sea verdadera la
    relación
    esHijo(Abuelo, Padre).
```

- Imprimir árbol

Imprime el árbol de familia a partir de la `Persona` dada con ayuda del predicado `forall` [4] para que imprima todos los hijos de la `Persona` si es que los tiene.

```
arbol(Persona) :- pareja(Persona, Esposo), tab(10), write(Persona), write('--'),
write(Esposo), nl,
    forall(esHijo(Persona, Hijo), (tab(5), write(' | '))), nl,
    forall(esHijo(Persona, Hijo), (tab(5), write(Hijo))), nl,
    forall(esHijo(Persona, Hijo), imprimirHijos(Hijo)).
```

```
Terminal - swipl -s problema1.pl
File Edit View Terminal Tabs Help
> swipl -s problema1.pl
Welcome to SWI-Prolog (threaded, 64 bits, version 8.2.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.
Para saber si es abuelo o abuela
For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- arbol(peter).
Abuelo(Abuelo, peter--diana
    % busca el padre de 'Nieto' y lo almacena en 'Padre'
    harry(Padre, mary(Nieto)),
    % busca que el padre de 'Padre' sea 'Abuelo' para que sea verdadera la relación
    esHijo(Abuelo, Padre).

true.
?- arbol
?- el arbol de familia a partir de la Persona dada con ayuda del predicado forall
(1/41) para que imprima todos los hijos de la Persona si es que los tiene
```

Culpable

Para determinar al culpable se realizó la siguiente regla

```
esCulpable(Culpable) :-
    % tiene como abuelo a Bruce
    esAbuelo(bruce, Culpable),
    % tiene como primo a Clark
    primo(clark, Culpable),
    % tiene como tío a Barry
    tio(Culpable, barry),
    % tiene una hermana (debe pertenecer a la lista de hermanos de quien sea su
hermana)
    % los tres antecedentes se saben por su hermana
    hermana(_, Hermanos), member(Culpable, Hermanos).
```

```
Terminal - swipl -s problema1.pl
File Edit View Terminal Tabs Help
> swipl -s problema1.pl
Welcome to SWI-Prolog (threaded, 64 bits, version 8.2.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.
Para determinar al culpable se realizó la siguiente regla
For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).
Culpable(Culpable) :-
?- esCulpable(ElCulpableEs).
ElCulpableEs = harry
    % tiene como primo a Clark
```

Problema 3

Para este problema se requiere manejo de listas en prolog y la solución se encuentra en el archivo [problema 3](#) de prolog.

Reverso de Lista

Se crea la regla **darVuelta** , que recibe la lista a voltear y devuelve la lista volteada

```
darVuelta(L_ingreso, L_volteada) :- reverse(L_ingreso, L_volteada).
```

Este se realiza mediante el predicado `reverse` de prolog [1]

Por lo tanto, se ingresa la lista `[6,7,87,[a,b,c],17]` :

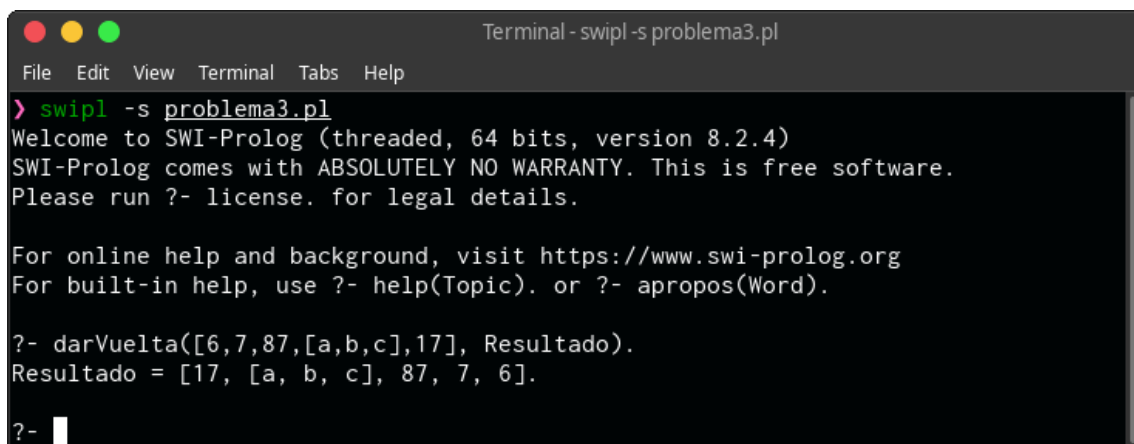


```
Terminal - swipl -s problema3.pl
File Edit View Terminal Tabs Help
> swipl -s problema3.pl
Welcome to SWI-Prolog (threaded, 64 bits, version 8.2.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- darVuelta([6,7,87,[a,b,c],17], Resultado).
```

Y el resultado se muestra en la variable `Resultado` :



```
Terminal - swipl -s problema3.pl
File Edit View Terminal Tabs Help
> swipl -s problema3.pl
Welcome to SWI-Prolog (threaded, 64 bits, version 8.2.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- darVuelta([6,7,87,[a,b,c],17], Resultado).
Resultado = [17, [a, b, c], 87, 7, 6].
?- 
```

Palíndromo

Se crea la regla **esPalindromo** la cual recibe la lista, verifica que la lista al darle vuelta sea exactamente lo mismo a la lista original y devuelve el estado `true` o `false` , según sea el caso.

```
esPalindromo(L_ingreso) :- reverse(L_ingreso, L_ingreso) .
```

Esta regla también hace uso del predicado `reverse` de prolog [1]

```
Terminal - swipl-s problema3.pl
File Edit View Terminal Tabs Help
> swipl -s problema3.pl
Welcome to SWI-Prolog (threaded, 64 bits, version 8.2.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- esPalindromo([a,b,c,d,c,b,a]).
true.

?- esPalindromo([q,w,e,r,t,y]).
false.

?-
```

Duplicar Lista

Se crea la regla `duplicarLista`, la cuál recibe la lista a duplicar y la variable donde se almacenará la lista ya duplicada. El cuerpo de la regla consiste en duplicar cada elemento, llamarse a sí misma para seguir con los demás elementos e ir adjuntando los elementos duplicados en la otra lista.

```
duplicarLista([Cabeza|Cola], L) :- X is Cabeza * 2, duplicarLista(Cola, L1),
append([X], L1, L) .
```

Esta regla es recursiva por la izquierda y su recursividad tendrá corta cuando:

```
duplicarLista([], []) :- ! .
```

Se corta cuando la lista a duplicar ya esté vacía, es decir, ya no tiene elementos por duplicar y la lista donde se almacenará aún esté vacía.

```
Terminal - swipl-s problema3.pl
File Edit View Terminal Tabs Help
> swipl -s problema3.pl
Welcome to SWI-Prolog (threaded, 64 bits, version 8.2.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- duplicarLista([6,81,7,17], ListaDuplicada).
ListaDuplicada = [12, 162, 14, 34].

?-
```

División en dos listas

Para la división de dos listas dada una lista completa, se utilizaron cuatro reglas:

1. Determinar el tamaño de la lista de entrada

Mediante la regla `tamanoLista` que es recursivo y corta la recursividad cuando haya recorrido toda la lista y el contador del resultado aún siga siendo `0`.

```
tamanoLista([],0) :- ! .
tamanoLista([_|Cola], L) :- tamanoLista(Cola, L1), L is L1 + 1.
```

2. Determinar mitad de la lista

Mediante la regla `mitadLista` que recibe la lista a determinar y la variable `Mitad` que almacena la mitad del tamaño de la lista. Esta regla llama a la [Regla 1](#) de esta [sección](#) para determinar la longitud de la lista y luego redondearla mediante el predicado `round` [2].

```
mitadLista(Lista, Mitad) :- tamanoLista(Lista, Long), Mitad is round(Long / 2) .
```

3. Dividir lista hasta cierta posición

Mediante la regla `dividir` se divide la lista hasta la posición que se le indique en `Contador` y la lista regresa en la variable `Lista`, la regla corta con su recursividad cuando el `Contador` es `0` y la lista de retorno todavía esté vacía sin importar lo que tenga la lista ingresada.

```
dividir(_, 0, []) :- ! .
dividir([Cabeza|Cola], Contador, Lista) :-
    % disminuir el contador una unidad
    C1 is Contador - 1,
    % invocarse a sí mismo para seguir dividiendo la lista
    dividir(Cola, C1, L1),
    % cuando corte con la recursividad, adjuntar en 'Lista' la lista dividida
    append([Cabeza], L1, Lista) .
```

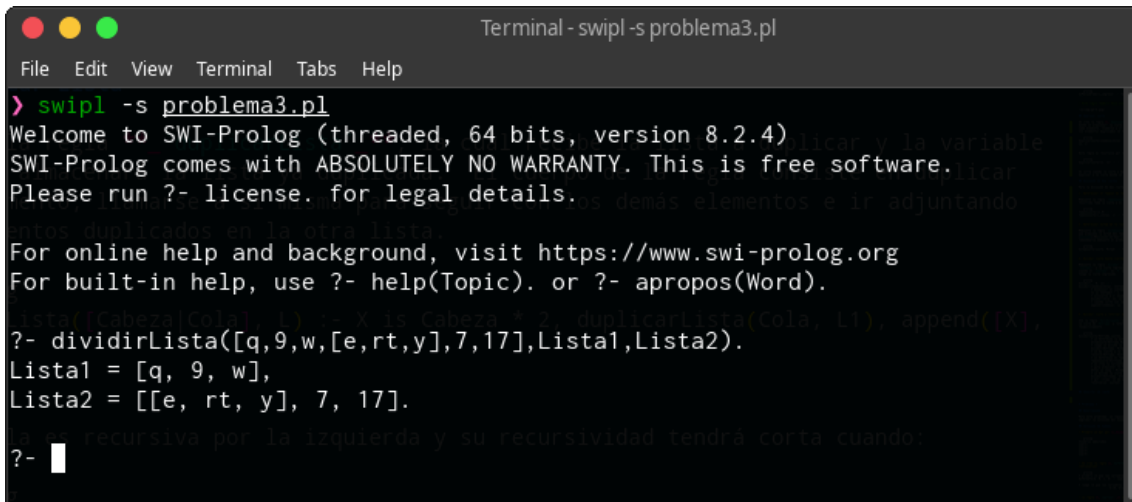
4. Dividir lista y retornar en dos listas

Con la regla `dividirLista` se llaman las anteriores reglas de esta sección para dividir la lista de `Entrada` y retornar en dos listas diferentes `Lista1` y `Lista2` respectivamente.

```
dividirLista(Entrada, Lista1, Lista2) :-
    % se determina la mitad de la lista
    mitadLista(Entrada, Mitad),
    % luego se divide la lista hasta la mitad de la lista, retorna en 'Lista1'
    dividir(Entrada, Mitad, Lista1),
    % se da vuelta a la lista de 'Entrada' y regresa en 'Volteada'
    reverse(Entrada, Volteada),
    % se determina el tamaño de la lista 'Volteada'
    tamanoLista(Volteada, Long),
    % se crea la variable 'Atras' que contiene la longitud total menos la mitad
    % para almacenar lo restante de la lista de 'Entrada'
    Atras is Long - Mitad,
    % se divide la lista original 'Volteada' y se almacena en 'L2'
    dividir(Volteada, Atras, L2),
```



```
% por último, se da vuelta a la lista 'L2' y se almacena en 'Lista2'  
reverse(L2, Lista2).
```



```
Terminal - swipl -s problema3.pl  
File Edit View Terminal Tabs Help  
> swipl -s problema3.pl  
Welcome to SWI-Prolog (threaded, 64 bits, version 8.2.4)  
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.  
Please run ?- license. for legal details.  
For online help and background, visit https://www.swi-prolog.org  
For built-in help, use ?- help(Topic). or ?- apropos(Word).  
?- dividirLista([q,9,w,[e,rt,y],7,17],Lista1,Lista2).  
Lista1 = [q, 9, w],  
Lista2 = [[e, rt, y], 7, 17].  
?-
```

Insertar en lista

Para insertar lista se crea una sola regla que recibe:

- Elemento: el elemento a insertar en la lista
- Entrada: la lista en donde se va a inserta el 'Elemento'
- Pos: la posición en donde se va a insertar en la lista 'Entrada' . (La posición comienza en 0).
- Lista: la variable donde se almacenará la lista con el 'Elemento' insertado en la lista 'Entrada' .

Y la regla tiene como cuerpo lo siguiente con sus respectivas explicaciones:

```
insertar(Elemento,Entrada,Pos,Lista) :-  
    % verificar que 'Pos' sea mayor a 0  
    Pos >= 0,  
    % se almacena el tamaño de la lista en 'Long'  
    tamanoLista(Entrada, Long),  
    % verificar que 'Pos' sea menor que la máxima posición en la lista 'Entrada'  
    Pos < Long + 1,  
    % se divide la lista de entrada hasta la posición indicada y se guarda en  
    'Lista1'  
    dividir(Entrada,Pos,Lista1),  
    % se adjunta el 'Elemento' a insertar a la lista anterior 'Lista1'  
    append(Lista1, [Elemento], Nueva),  
    % se voltea la lista de entrada y se almacena en 'Volteada'  
    reverse(Entrada, Volteada),  
    % en la variable 'Atras' se guarda la cantidad de elementos que hacen falta  
    Atras is Long - Pos,  
    % en 'L2' se almacena el resto de la lista  
    dividir(Volteada, Atras, L2),  
    % se da vuelta a 'L2' y se almacena en 'Lista2'  
    reverse(L2,Lista2),  
    % se adjunta 'Lista2' a la lista 'Nueva' y regresa en 'Lista'  
    append(Nueva, Lista2, Lista) .
```

```
Terminal - swipl -s problema3.pl
File Edit View Terminal Tabs Help
> swipl -s problema3.pl
Welcome to SWI-Prolog (threaded, 64 bits, version 8.2.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.
For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).
?- insertar(CabeceraCola, L1, X, Y, Cabecera, Cola, L2, duplicarLista(Cola, L1), append([X],
|
Lista = [4, 5, 7, 8, [a, b, c], h, d, r, 4].
?- 
```

Problema 4

Descripción

Para este problema, se requiere ingresar un tablero de sudoku de 4x4 a resolver para imprimir la solución del mismo.

Solución de Sudoku

- Primero se dar por **hecho** que los números admitidos únicamente son `1, 2, 3, 4`

```
%numeros admitidos
num(1).
num(2).
num(3).
num(4).
```

- Se crea la regla **diferente** la cuál realiza la validación como se detalla a continuación

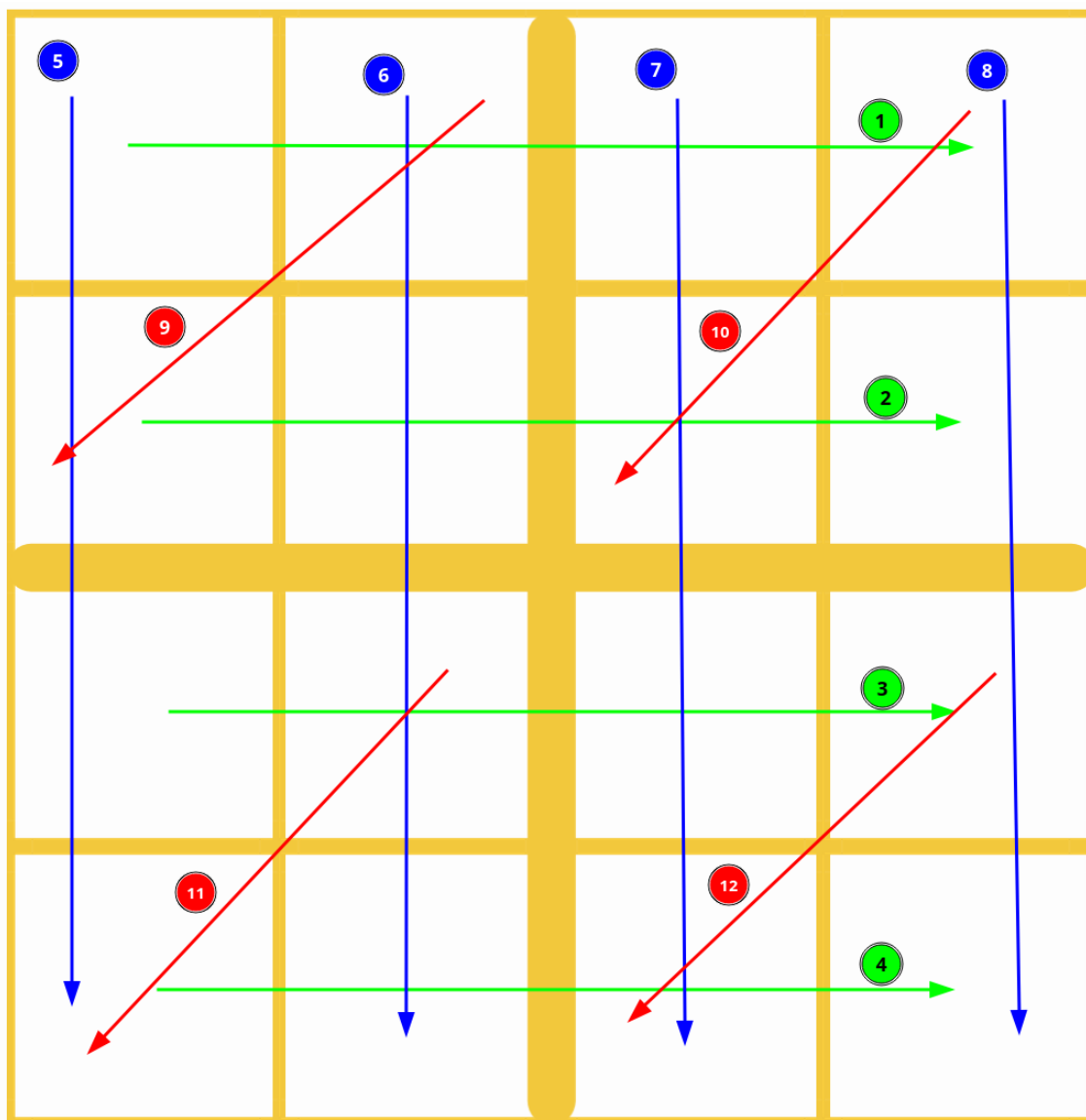
```
% primero ve que las variables A,B,C,D sean del 1-4
diferente(A, B, C, D) :- num(A), num(B), num(C), num(D),

% luego ve que las variables A,B,C,D sean distintas entre ellas
A\=B, A\=C, A\=D, B\=C, B\=D, C\=D .
```

- La regla **imprimir** sirve para imprimir los valores de cada columna por fila.

```
imprimir(A, B, C, D) :-
    write(' '), write(A), write(' | '), write(B), write(' || '), write(C), write(' | '),
    write(D), nl.
```

- La regla **resuelto** recibe el tablero completo y hace las siguientes verificaciones con la regla **diferente** :



% recibe el tablero completo de sudoku

```
resuelto(F1C1, F1C2, F1C3, F1C4,
        F2C1, F2C2, F2C3, F2C4,
        F3C1, F3C2, F3C3, F3C4,
        F4C1, F4C2, F4C3, F4C4) :-
```

% 1. Fila 1

```
diferente(F1C1, F1C2, F1C3, F1C4),
```

% 2. Fila 2

```
diferente(F2C1, F2C2, F2C3, F2C4),
```

% 3. Fila 3

```
diferente(F3C1, F3C2, F3C3, F3C4),
```

% 4. Fila 4

```
diferente(F4C1, F4C2, F4C3, F4C4),
```

% 5. Columna 1

```
diferente(F1C1, F2C1, F3C1, F4C1),
```

```

% 6. Columna 2
diferente(F1C2, F2C2, F3C2, F4C2),
% 7. Columna 3
diferente(F1C3, F2C3, F3C3, F4C3),
% 8. Columna 4
diferente(F1C4, F2C4, F3C4, F4C4),
% 9. Cuadrante 1
diferente(F1C1, F1C2, F2C1, F2C2),
% 10. Cuadrante 2
diferente(F1C3, F1C4, F2C3, F2C4),
% 11. Cuadrante 3
diferente(F3C1, F3C2, F4C1, F4C2),
% 12. Cuadrante 4
diferente(F3C3, F3C4, F4C3, F4C4).

```

- Finalmente, la regla **sudoku** recibe el tablero que ingresa el usuario para luego resolver mediante la regla **resuelto** y por último imprimir el tablero con la regla **imprimir**

```

sudoku(F1C1, F1C2, F1C3, F1C4,
       F2C1, F2C2, F2C3, F2C4,
       F3C1, F3C2, F3C3, F3C4,
       F4C1, F4C2, F4C3, F4C4) :-

    resuelto(F1C1, F1C2, F1C3, F1C4,
             F2C1, F2C2, F2C3, F2C4,
             F3C1, F3C2, F3C3, F3C4,
             F4C1, F4C2, F4C3, F4C4),nl,

    nl,write('Solución para sudoku:'),nl,nl,
    imprimir(F1C1, F1C2, F1C3, F1C4),
    write('---|---|---|---'),nl,
    imprimir(F2C1, F2C2, F2C3, F2C4),
    write('===|===|===|==='),nl,
    imprimir(F3C1, F3C2, F3C3, F3C4),
    write('---|---|---|---'),nl,
    imprimir(F4C1, F4C2, F4C3, F4C4),nl.

```

Sudoku

Se ejemplifica con el siguiente tablero

			3
	2		
		3	
4			

Los espacios en blanco se ingresan con la variable `_` de prolog con la regla **sudoku** que recibe le tablero de la siguiente manera:

```

Terminal - swipl-s problema4.pl
File Edit View Terminal Tabs Help
> cd Documents/IA/proyecto1-IA/
> swipl -s problema4.pl
Welcome to SWI-Prolog (threaded, 64 bits, version 8.2.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).
?- sudoku(F3C1, F3C2, F3C3, F3C4,
|      _,_,_,3,
|n1,w_,2,_,_,ucion para sudoku:').n1,n1,
|mpr_,_,3,_,_,F1C2, F1C3, F1C4),
|rit(4,_,_,_,_).|---|---').n1,
|mpr(F2C1, F2C2, F2C3, F2C4),

```

Luego de ingresar, se procede a dar `enter` y el resultado se muestra en la pantalla tal y como se muestra a continuación

```
Terminal - swipl -s problema4.pl
File Edit View Terminal Tabs Help
> swipl -s problema4.pl
Welcome to SWI-Prolog (threaded, 64 bits, version 8.2.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.
?- F3C1, F3C2, F3C3, F3C4).
For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- sudoku(
|   _,-,-,3,
|   _,-,2,-,
| on _,-,3,-,nte tablero
|   4,-,-,-).
true)

Solución para sudoku: la variable _ de prolog con la regla
recibe la tablero de la siguiente manera:
 1 | 4 | | 2 | 3
---|---|---|---
 3 | 2 | | 4 | 1
===|===|===|===
 2 | 1 | | 3 | 4
---|---|---|---
 4 | 3 | | 1 | 2

true
```

Referencias

1

SWI-Prolog. (2020). Predicate reverse/2. septiembre 4, 2021, de SWI-Prolog Sitio web: <https://www.swi-prolog.org/pldoc/man?predicate=reverse/2>

2

SWI-Prolog. (2020). Predicate round/1. septiembre 6, 2021, de SWI-Prolog Sitio web: <https://eu.swi-prolog.org/pldoc/man?function=round/1>

3

SWI-Prolog. (2020). Predicate member/2. septiembre 6, 2021, de SWI-Prolog Sitio web: https://eu.swi-prolog.org/pldoc/doc_for?object=member/2

4

SWI-Prolog. (2020). Predicate forall/2. septiembre 6, 2021, de SWI-Prolog Sitio web: https://www.swi-prolog.org/pldoc/doc_for?object=forall/2