Resolución de problemas usando lógica proposicional

Sesión 7

Edgar Andrade, PhD

Marzo de 2019

Departmento de Matemáticas Aplicadas y Ciencias de la Computación

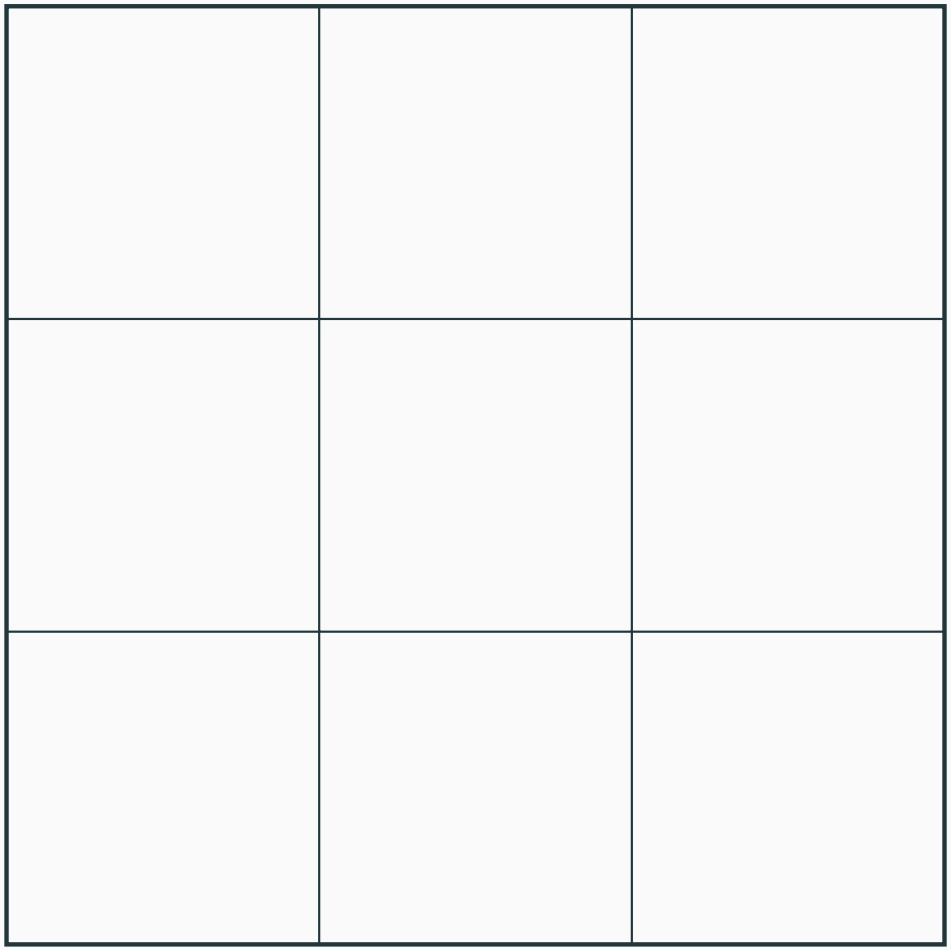
Presentación

En esta sesión estudiaremos:

1. Representación de situaciones sin condiciones iniciales
2. Representación de situaciones con condiciones iniciales
3. Búsqueda de soluciones
4. Interpretación de soluciones

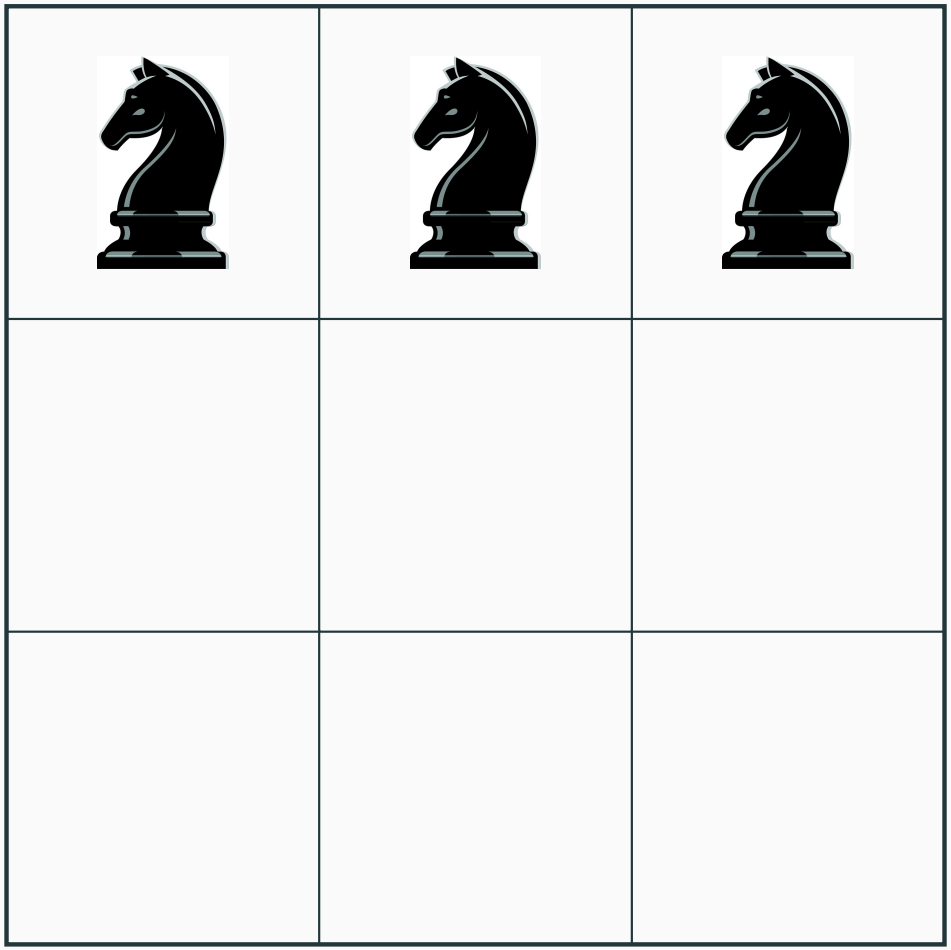
Contenido

Problema —sin condiciones iniciales—



Considere un tablero de ajedrez de 3×3. El problema consiste en ubicar tres caballos en el tablero de tal manera que ningún caballo ataque a otro.

Ejemplo



Por ejemplo, si ubicamos los caballos como en la figura, ninguno ataca a otro.

Claves de representación (1/2)

Primero enumeramos las casillas del tablero de la siguiente manera:

1	2	3
4	5	6
7	8	9

Claves de representación (2/2)

Una letra proposicional c_i para cada casilla i.

c_i es verdadera sii hay un caballo ocupando la casilla i.

c_1	c_2	c_3
c_4	c_5	c_6
c_7	c_8	c_9

Ejemplo

c_1: hay un caballo en 1

$\neg c_2$: no hay un caballo en 2

$\neg c_3$: no hay un caballo en 3

$\neg c_4$: no hay un caballo en 4

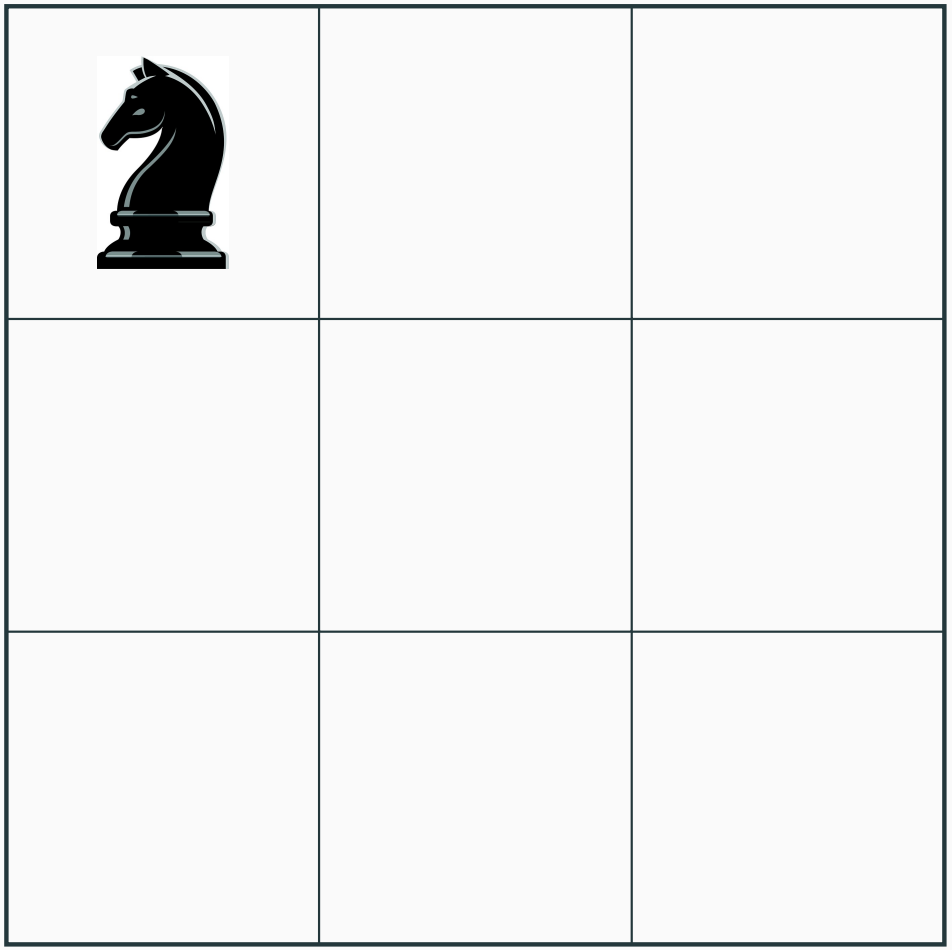
$\neg c_5$: no hay un caballo en 5

$\neg c_6$: no hay un caballo en 6

$\neg c_7$: no hay un caballo en 7

$\neg c_8$: no hay un caballo en 8

$\neg c_9$: no hay un caballo en 9



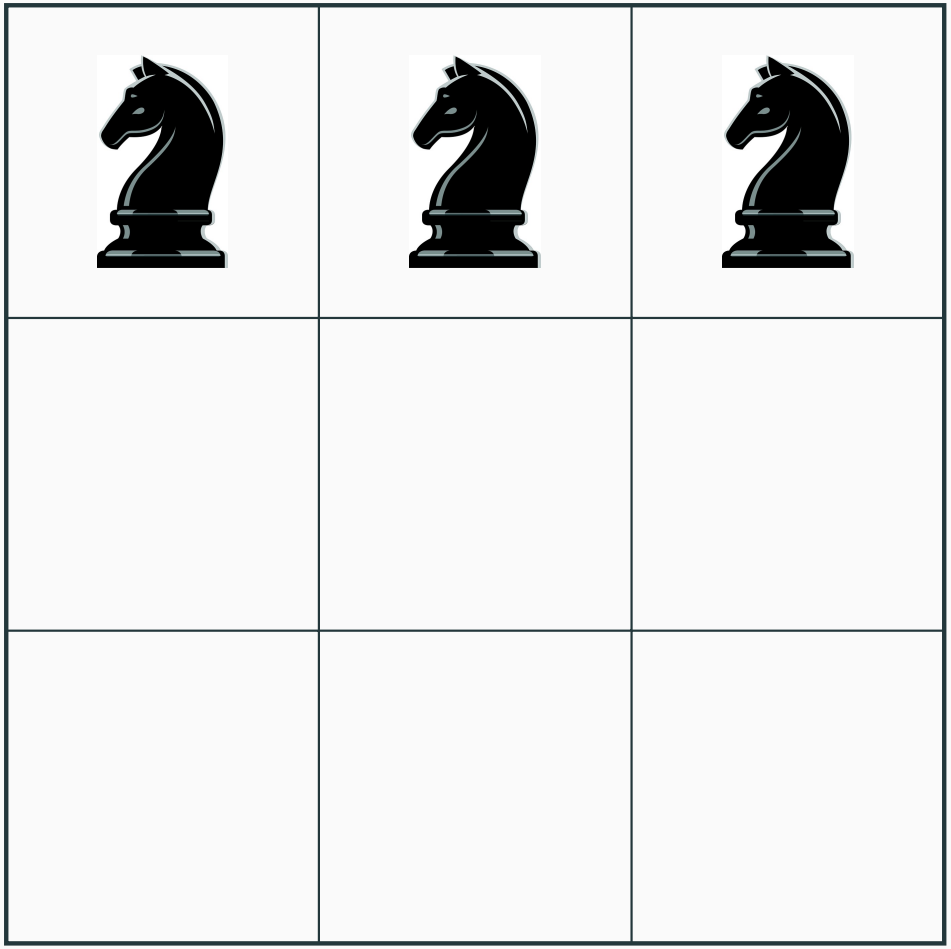
Tipos de reglas

Regla 1: Debe haber exactamente tres caballos en el tablero.

Regla 2: Ningún caballo debe poder atacar a otro.

Regla 1 (incompleta...) (1/2)

$$
\begin{aligned}
&(\\
&\quad c_1 \wedge c_2 \wedge c_3 \\
&\quad \wedge \neg c_4 \wedge \neg c_5 \wedge \neg c_6 \\
&\quad \wedge \neg c_7 \wedge \neg c_8 \wedge \neg c_9 \\
&) \vee \ldots
\end{aligned}
$$

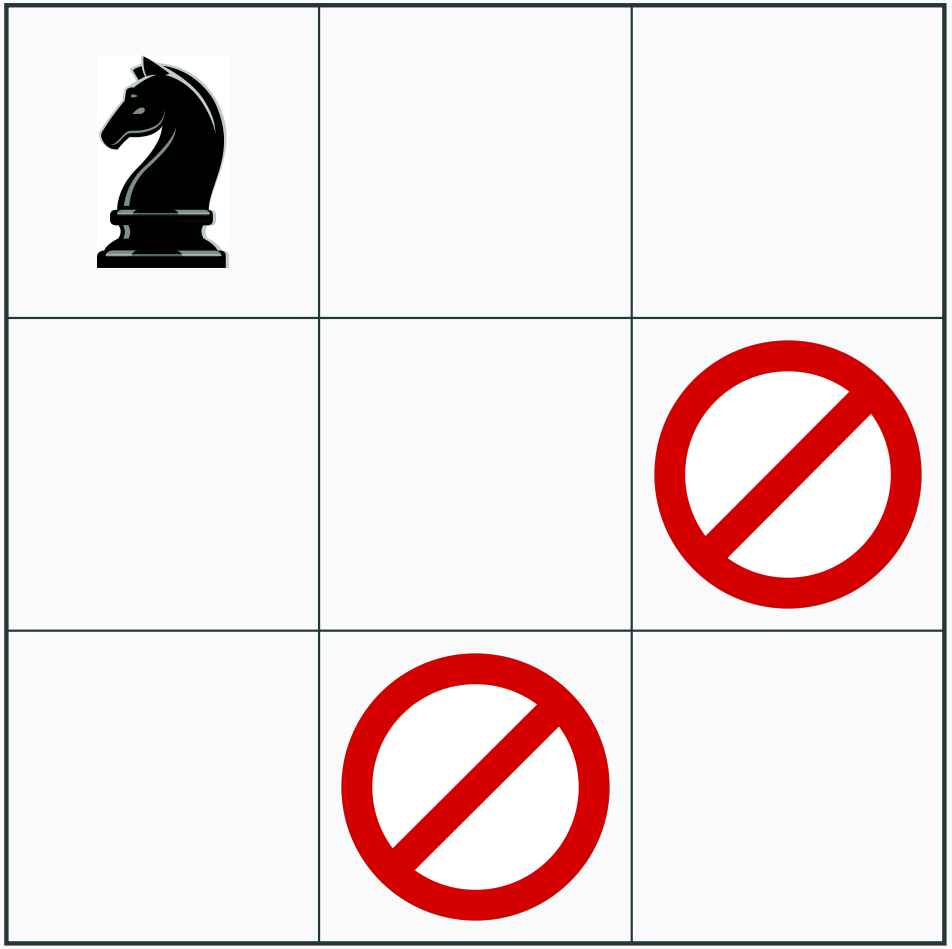


Regla 1 (incompleta...) (2/2)

Son exactamente $9 \times 8 \times 7$ cláusulas, una por cada configuración posible de exactamente tres caballos en el tablero 3×3.

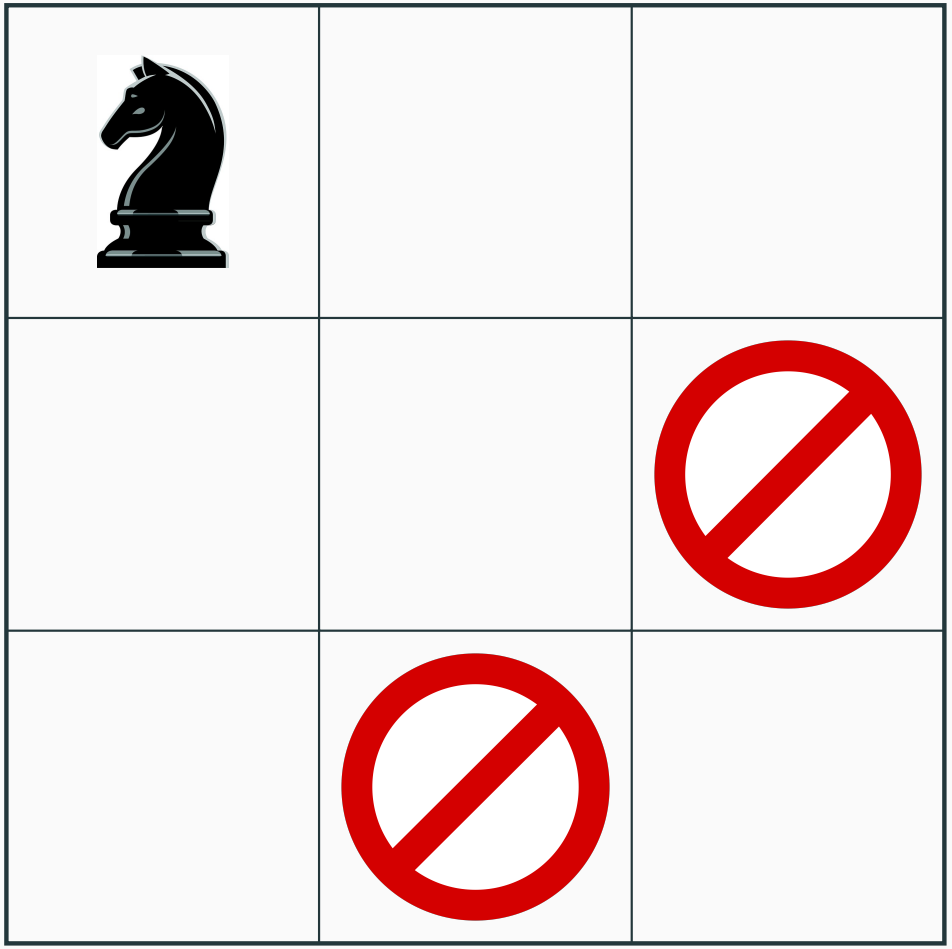
Ejemplo de reglas tipo 2 —informal— (1/4)

Si hay un caballo en 1, no debe haber un caballo ni en 6 ni en 8, puesto que se estarían atacando mútuamente.



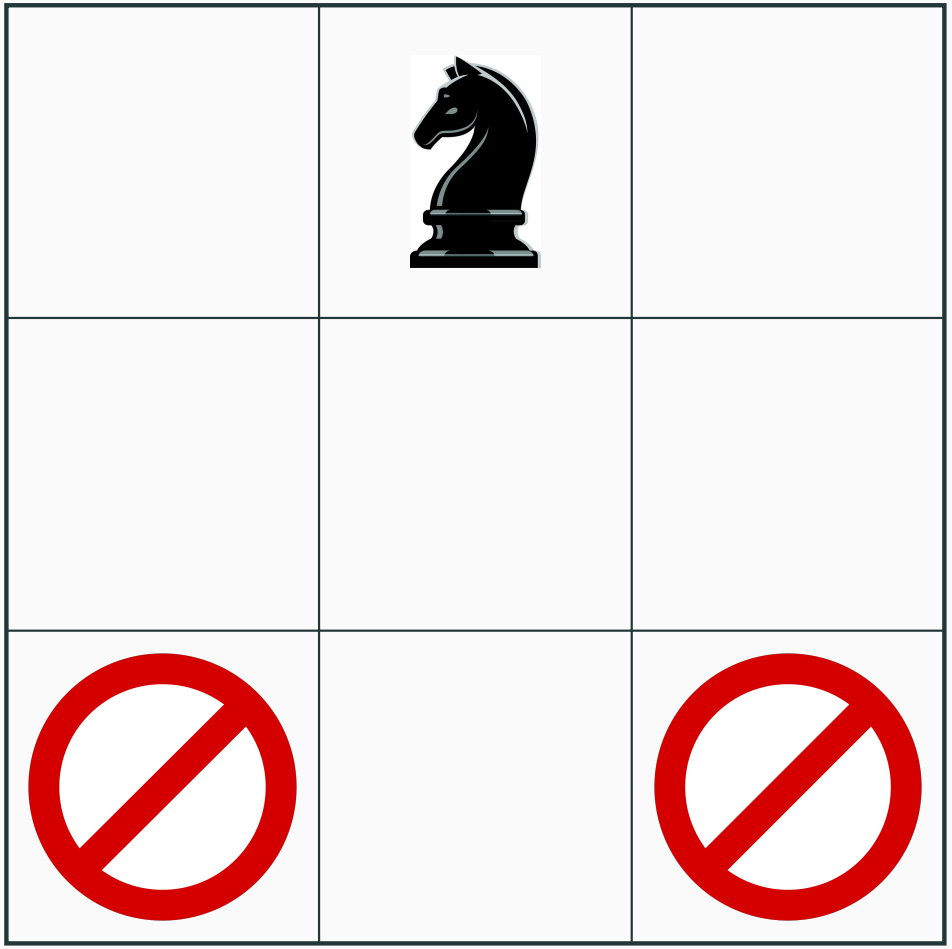
Ejemplo de reglas tipo 2 —formal— (2/4)

$c_1 \rightarrow (\neg c_6 \wedge \neg c_8)$



Ejemplos de reglas tipo 2 (3/4)

$c_2 \rightarrow (\neg c_7 \wedge \neg c_9)$



Reglas tipo 2 (4/4)

$$c_1 \rightarrow (\neg c_6 \wedge \neg c_8)$$

$$c_2 \rightarrow (\neg c_7 \wedge \neg c_9)$$

$$c_3 \rightarrow (\neg c_4 \wedge \neg c_8)$$

$$c_4 \rightarrow (\neg c_3 \wedge \neg c_9)$$

$$c_6 \rightarrow (\neg c_1 \wedge \neg c_7)$$

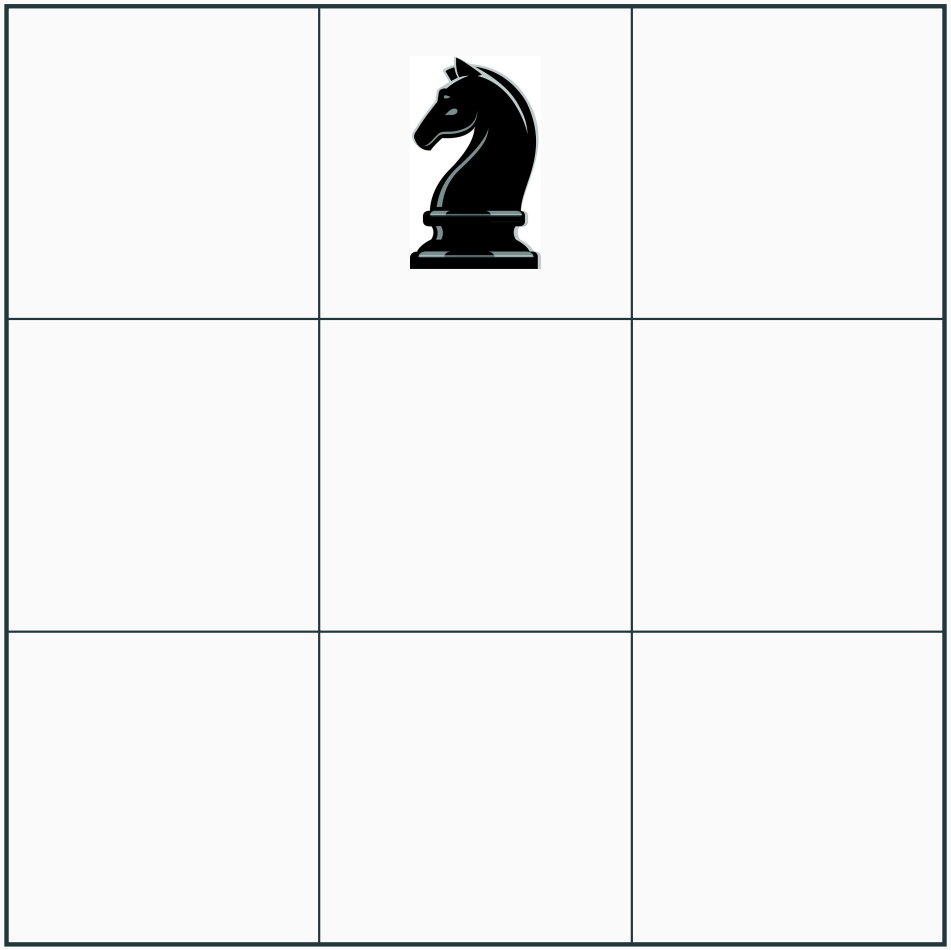
$$c_7 \rightarrow (\neg c_2 \wedge \neg c_6)$$

$$c_8 \rightarrow (\neg c_1 \wedge \neg c_3)$$

$$c_9 \rightarrow (\neg c_2 \wedge \neg c_4)$$

Contenido

Problema —con condiciones iniciales—



Dado un caballo en un tablero de ajedrez de 3×3, el problema consiste en ubicar otros dos caballos de tal manera que ningún caballo ataque a otro.

Tipos de reglas

Regla 1: Debe haber exactamente tres caballos en el tablero.

Regla 2: Ningún caballo debe poder atacar a otro.

Regla 3: Debe haber un caballo en la casilla c_i.

Contenido

Idea del procedimiento

1. Representar las reglas mediante fórmulas de la lógica proposicional, $\varphi_1, \ldots, \varphi_n$.

Idea del procedimiento

1. Representar las reglas mediante fórmulas de la lógica proposicional, $\varphi_1, \ldots, \varphi_n$.
2. Encontrar las interpretaciones I que hacen verdadera a la fórmula $\varphi_1 \wedge \ldots \wedge \varphi_n$.

Idea del procedimiento

1. Representar las reglas mediante fórmulas de la lógica proposicional, $\varphi_1, \ldots, \varphi_n$.
2. Encontrar las interpretaciones I que hacen verdadera a la fórmula $\varphi_1 \wedge \ldots \wedge \varphi_n$.

☞ Este procedimiento se puede realizar, aunque de manera muy ineficiente, mediante tablas de verdad.

Idea del procedimiento

1. Representar las reglas mediante fórmulas de la lógica proposicional, $\varphi_1, \ldots, \varphi_n$.
2. Encontrar las interpretaciones I que hacen verdadera a la fórmula $\varphi_1 \wedge \ldots \wedge \varphi_n$.

☞ Este procedimiento se puede realizar, aunque de manera muy ineficiente, mediante tablas de verdad.

3. Finalmente, las interpretaciones I encontradas se interpretan como soluciones del problema.

Idea del procedimiento —1—

1. Representar las reglas mediante fórmulas de la lógica proposicional

   1a. Representar las reglas como cadenas en notación polaca inversa.

Idea del procedimiento —1—

1. Representar las reglas mediante fórmulas de la lógica proposicional

   1a. Representar las reglas como cadenas en notación polaca inversa.

   1b. Transformar las cadenas en árboles.

Intermezzo: Notación Polaca y Notación Polaca Inversa

La notación polaca pone los conectivos lógicos primero, seguidos de sus argumentos (no requiere paréntesis):

$p \to q$ se denota como $\to pq$

Intermezzo: Notación Polaca y Notación Polaca Inversa

La notación polaca pone los conectivos lógicos primero, seguidos de sus argumentos (no requiere paréntesis):

$p \to q$ se denota como $\to pq$

$p \land \neg(q \lor r)$ se denota como $\land p \neg \lor qr$

Intermezzo: Notación Polaca y Notación Polaca Inversa

La notación polaca pone los conectivos lógicos primero, seguidos de sus argumentos (no requiere paréntesis):

$p \to q$ se denota como $\to pq$ Inversa: $qp \to$

$p \wedge \neg(q \vee r)$ se denota como $\wedge p \neg \vee qr$

Intermezzo: Notación Polaca y Notación Polaca Inversa

La notación polaca pone los conectivos lógicos primero, seguidos de sus argumentos (no requiere paréntesis):

$p \to q$ se denota como $\to pq$ Inversa: $qp \to$

$p \wedge \neg(q \vee r)$ se denota como $\wedge p \neg \vee qr$ Inversa: $rq \vee \neg p \wedge$

1a

1a. Representar las reglas en notación polaca inversa:

Hay exactamente tres caballos en la primera fila.

1a

1a. Representar las reglas en notación polaca inversa:

Hay exactamente tres caballos en la primera fila.

$(((((((c_1 \wedge c_2) \wedge c_3) \wedge \neg c_4) \wedge \neg c_5) \wedge \neg c_6) \wedge \neg c_7) \wedge \neg c_8) \wedge \neg c_9$

1a

1a. Representar las reglas en notación polaca inversa:

Hay exactamente tres caballos en la primera fila.

$(((((((c_1 \wedge c_2) \wedge c_3) \wedge \neg c_4) \wedge \neg c_5) \wedge \neg c_6) \wedge \neg c_7) \wedge \neg c_8) \wedge \neg c_9$

Notación Polaca: $\wedge \wedge \wedge \wedge \wedge \wedge \wedge \wedge c_1 c_2 c_3 \neg c_4 \neg c_5 \neg c_6 \neg c_7 \neg c_8 \neg c_9$

1a

1a. Representar las reglas en notación polaca inversa:

Hay exactamente tres caballos en la primera fila.

$(((((((c_1 \land c_2) \land c_3) \land \neg c_4) \land \neg c_5) \land \neg c_6) \land \neg c_7) \land \neg c_8) \land \neg c_9$

Notación Polaca: $\land \land \land \land \land \land \land \land c_1 c_2 c_3 \neg c_4 \neg c_5 \neg c_6 \neg c_7 \neg c_8 \neg c_9$

NP Inversa: $c_9 \neg c_8 \neg c_7 \neg c_6 \neg c_5 \neg c_4 \neg c_3 c_2 c_1 \land \land \land \land \land \land \land \land$

1a

1a. Representar las reglas en notación polaca inversa:

Hay exactamente tres caballos en la primera fila.

$(((((((c_1 \wedge c_2) \wedge c_3) \wedge \neg c_4) \wedge \neg c_5) \wedge \neg c_6) \wedge \neg c_7) \wedge \neg c_8) \wedge \neg c_9$

Notación Polaca: $\wedge\wedge\wedge\wedge\wedge\wedge\wedge\wedge c_1 c_2 c_3 \neg c_4 \neg c_5 \neg c_6 \neg c_7 \neg c_8 \neg c_9$

NP Inversa: $c_9 \neg c_8 \neg c_7 \neg c_6 \neg c_5 \neg c_4 \neg c_3 c_2 c_1 \wedge\wedge\wedge\wedge\wedge\wedge\wedge\wedge$

☞ Faltan las otras $9 \times 8 \times 7 - 1$ reglas.

1a —Python—

```
# Regla 1: Debe haber exactamente tres caballos
letrasProposicionales = [str(i) for i in range(1, 10)] # Creo las letras proposicionales
conjunciones = ' ' # Para ir guardando las conjunciones de trios de disyunciones de literales
inicial = True # Para inicializar la primera conjuncion

for p in letrasProposicionales:
    aux1 = [x for x in letrasProposicionales if x != p] # Todas las letras excepto p
    for q in aux1:
        aux2 = [x for x in aux1 if x != q] # Todas las letras excepto p y q
        for r in aux2:
            literal = r + q + p + 'Y' + 'Y'
            aux3 = [x + '-' for x in aux2 if x != r]
            for k in aux3:
                literal = k + literal + 'Y'
            if inicial: # Inicializar la primera conjuncion
                conjunciones = literal
                inicial = False
            else:
                conjunciones = literal + conjunciones + 'O'
```

1b —Python—

Transformar las cadenas en árboles.

```
def string2Tree(A, letrasProposicionales):
    # Crea una formula como tree dada una formula como cadena escrita en notacion polaca inversa
    # Input: A, lista de caracteres con una formula escrita en notacion polaca inversa
    #        letrasProposicionales, lista de letras proposicionales
    # Output: formula como tree
    conectivos = ['O', 'Y', '>']
    pila = []
    for c in A:
            if c in letrasProposicionales:
                    pila.append(Tree(c, None, None))
            elif c == '-':
                    formulaAux = Tree(c, None, pila[-1])
                    del pila[-1]
                    pila.append(formulaAux)
            elif c in conectivos:
                    formulaAux = Tree(c, pila[-1], pila[-2])
                    del pila[-1]
                    del pila[-1]
                    pila.append(formulaAux)
    return pila[-1]
```

Idea del procedimiento —2—

2. Encontrar las interpretaciones I que hacen verdadera a la fórmula $\varphi_1 \wedge \ldots \wedge \varphi_n$.

2a. Construir todas las interpretaciones.

Idea del procedimiento —2—

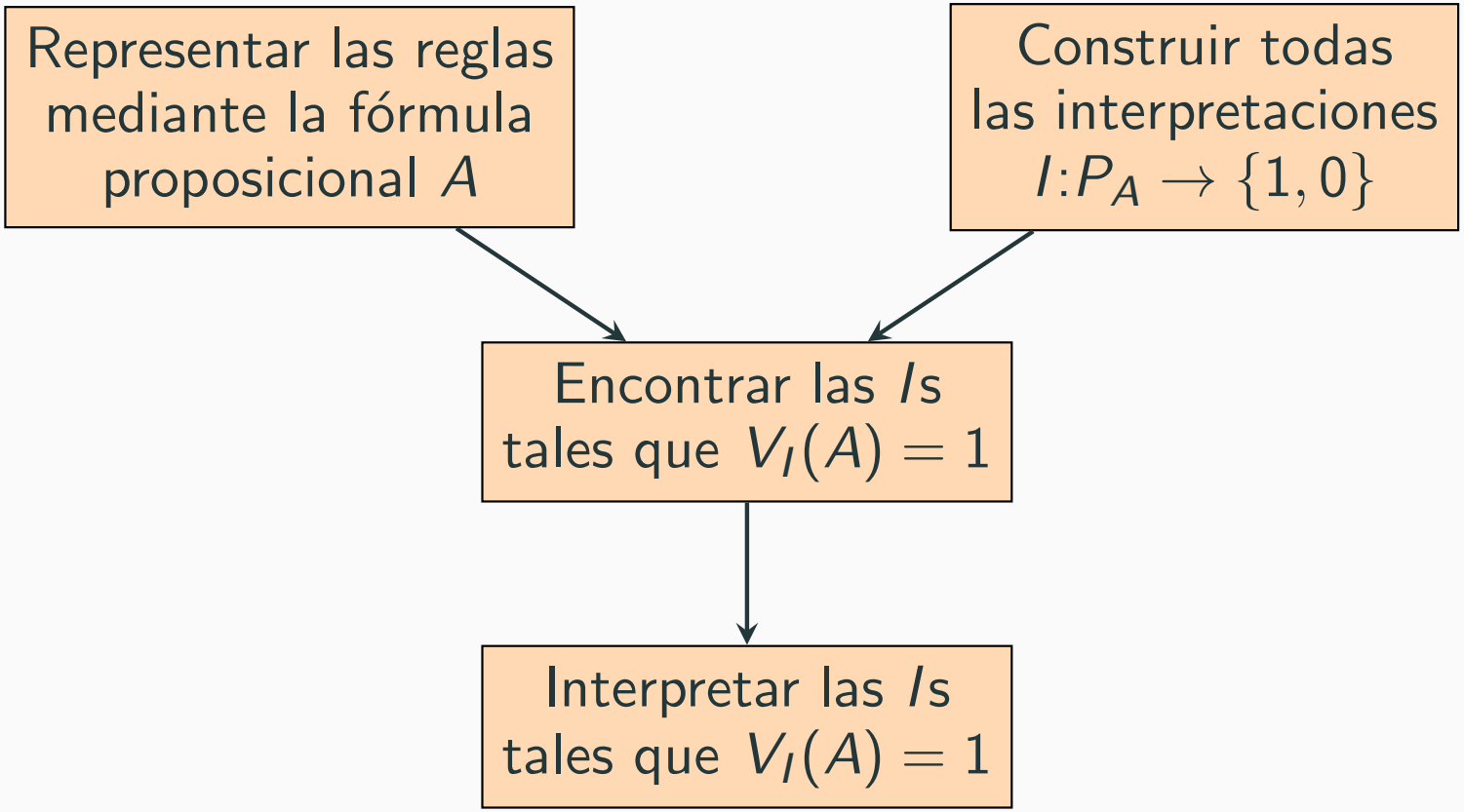
2. Encontrar las interpretaciones I que hacen verdadera a la fórmula $\varphi_1 \wedge \ldots \wedge \varphi_n$.

2a. Construir todas las interpretaciones.

2b. Para cada I, determinar si $V_I(\varphi_1 \wedge \ldots \wedge \varphi_n) = 1$.

Contenido

Esquema del procedimiento



Interpretación de soluciones

Las soluciones son asignaciones de valores 0s y 1s a letras proposicionales, es decir interpretaciones Is, tales que $V_I(\varphi_1 \wedge \ldots \wedge \varphi_n) = 1$.

Interpretación de soluciones

Las soluciones son asignaciones de valores 0s y 1s a letras proposicionales, es decir interpretaciones Is, tales que $V_I(\varphi_1 \wedge \ldots \wedge \varphi_n) = 1$.

Para resolver el problema, es indispensable interpretar esos valores de las letras proposicionales en la situación representada.

Interpretación de soluciones

Las soluciones son asignaciones de valores 0s y 1s a letras proposicionales, es decir interpretaciones Is, tales que $V_I(\varphi_1 \wedge \ldots \wedge \varphi_n) = 1$.

Para resolver el problema, es indispensable interpretar esos valores de las letras proposicionales en la situación representada.

Por ejemplo, en el problema de los caballos, las letras proposicionales con valor 1 son las casillas donde van los caballos.

Interpretación de soluciones —ejemplo caballos—

{1:0, 2:1, 3:0,
4:0, 5:0, 6:1,
7:0, 8:1, 9:0}

{1:0, 2:0, 3:0,
4:0, 5:1, 6:1,
7:0, 8:0, 9:1}

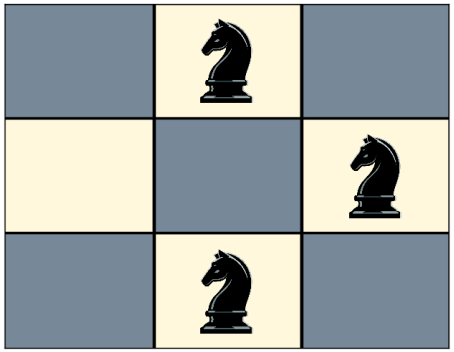
{1:0, 2:0, 3:0,
4:0, 5:1, 6:0,
7:1, 8:1, 9:0}

Interpretación de soluciones —ejemplo caballos—

{1:0, 2:1, 3:0,
4:0, 5:0, 6:1,
7:0, 8:1, 9:0}

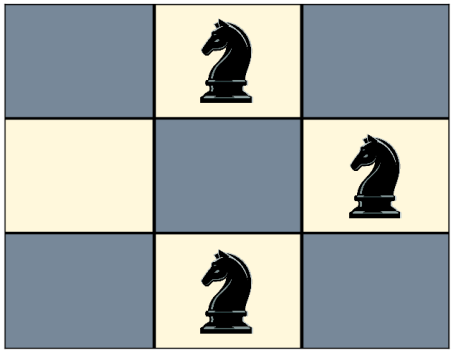
{1:0, 2:0, 3:0,
4:0, 5:1, 6:1,
7:0, 8:0, 9:1}

{1:0, 2:0, 3:0,
4:0, 5:1, 6:0,
7:1, 8:1, 9:0}

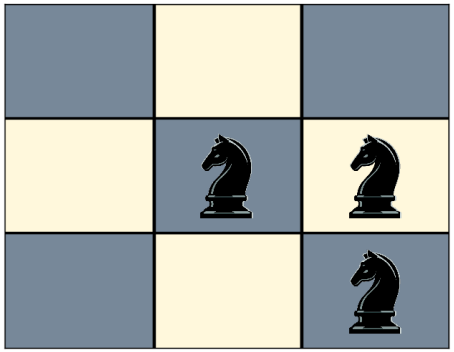


Interpretación de soluciones —ejemplo caballos—

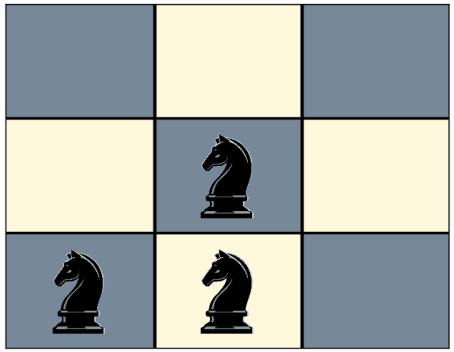
{1:0, 2:1, 3:0,
4:0, 5:0, 6:1,
7:0, 8:1, 9:0}



{1:0, 2:0, 3:0,
4:0, 5:1, 6:1,
7:0, 8:0, 9:1}

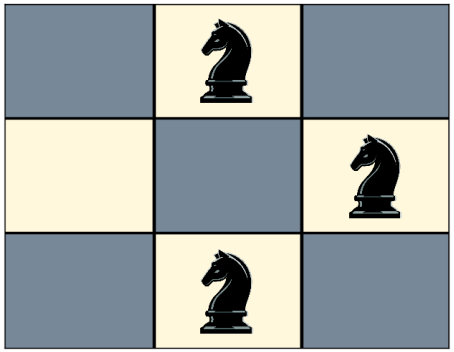


{1:0, 2:0, 3:0,
4:0, 5:1, 6:0,
7:1, 8:1, 9:0}

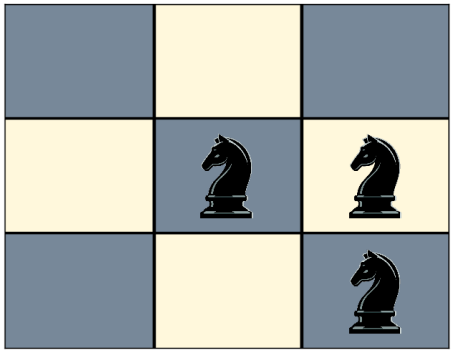


Interpretación de soluciones —ejemplo caballos—

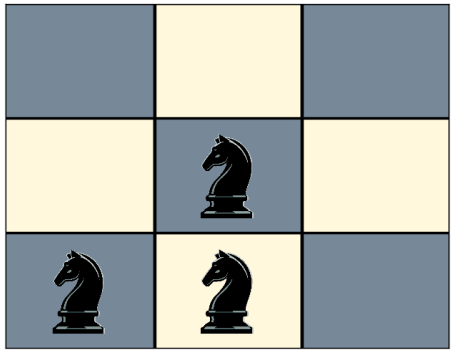
{1:0, 2:1, 3:0,
4:0, 5:0, 6:1,
7:0, 8:1, 9:0}



{1:0, 2:0, 3:0,
4:0, 5:1, 6:1,
7:0, 8:0, 9:1}



{1:0, 2:0, 3:0,
4:0, 5:1, 6:0,
7:1, 8:1, 9:0}



☞ Consultar archivo ‘visualizacion_tablero.py’.

Ejercicio

Escribir un código python que dibuje todas las soluciones posibles al problema de poner tres caballos en un tablero de ajedrez de tamaño 3×3, sin que se ataquen mutuamente, dados dos caballos iniciales: uno en la casilla 2 y otro en la casilla 6. [¡Son cuatro soluciones posibles!]

Fin de la sesión 7

En esta sesión usted ha aprendido a:

1. Representar situaciones mediante la lógica proposicional, con y sin condiciones iniciales.
2. Buscar soluciones mediante tablas de verdad.
3. Interpretar las soluciones.