Implementación de los Tableaux Semánticos de la Lógica Proposicional

Semana 10

Edgar Andrade, PhD

Última revisión: Marzo de 2022

Departmento de Matemáticas Aplicadas y Ciencias de la Computación

Presentación

En esta sesión estudiaremos:

1. Elementos de la implementación
2. Algoritmo primero en anchura
3. Algoritmo primero en profundidad
4. Algoritmo backtracking

Presentación

Diagrama de flujo de los tableaux

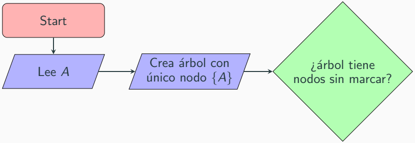


Diagrama de flujo de los tableaux

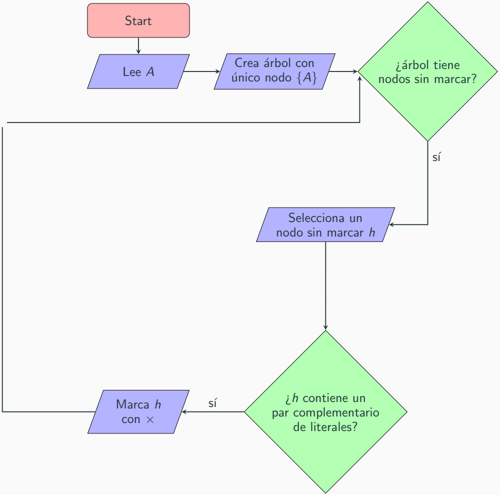


Diagrama de flujo de los tableaux

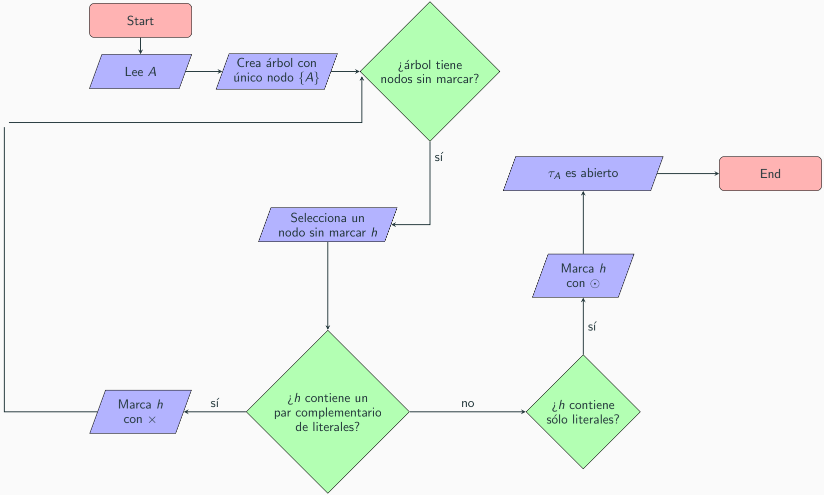


Diagrama de flujo de los tableaux

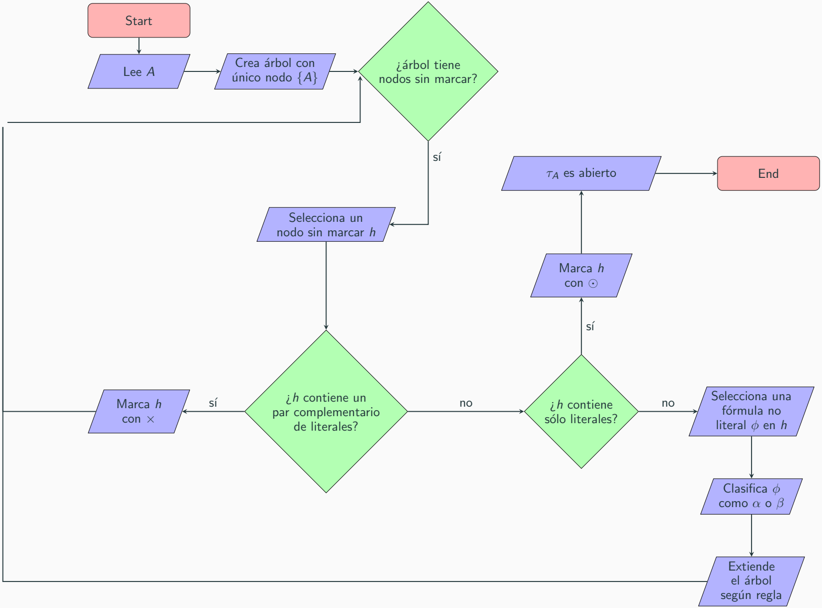
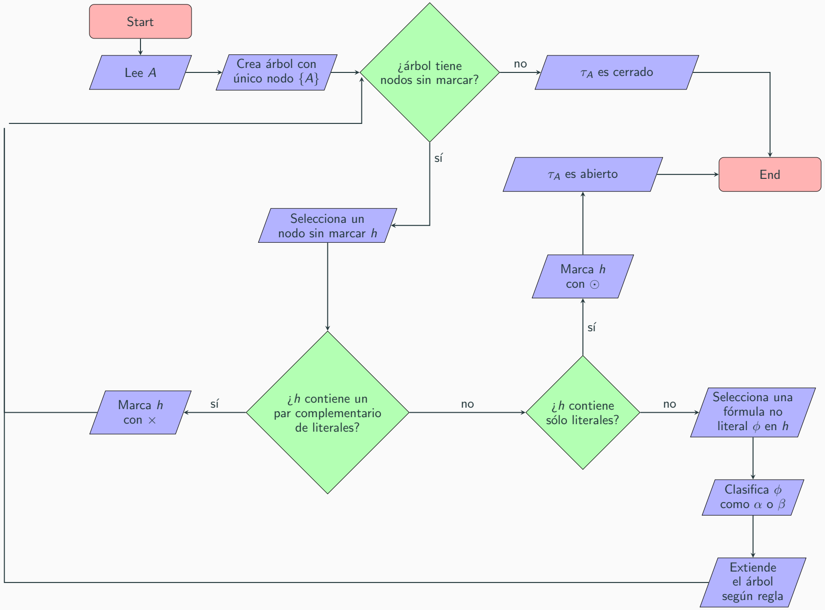


Diagrama de flujo de los tableaux



nodo_tableaux — atributos

Alfas

`self`.alfas es una lista con las fórmulas tipo α, su representación como cadena, el número de regla y la cadena 'alfa'.

Ejemplo:

$$n = \text{nodo\_tableaux}(['-(p > q)', '(pOq)', '-p'])$$

$$n.\text{alfas} = [(Formula, '-(p > q)', 4, 'alfa')]$$

nodo_tableaux — atributos

Betas

`self`.betas es una lista con las fórmulas tipo β, su representación como cadena, el número de regla y la cadena 'beta'.

Ejemplo:

$$n = \text{nodo\_tableaux}([{'}{-}(p > q){'},{'}(pOq){'},{'}{-}p{'}])$$

$$n.\text{betas} = [(Formula,{'}(pOq){'}, 2,{'}beta{'})]$$

nodo_tableaux — atributos

Literales

`self`.literales es una lista con los literales, su representación como cadena, `None` y la cadena 'literal'.

Ejemplo:

$$n = \text{nodo\_tableaux}([' {-}(p > q)', '(pOq)', ' {-}p'])$$

$$n.\text{literales} = [(Formula, ' {-}p', \texttt{None}, ' literal')]$$

nodo_tableaux — métodos

tiene_lit_comp

`self`.tiene_lit_comp() retorna `True` sii `self`.literales tiene un par complementario de literales.

Ejemplo:

$$n = \text{nodo\_tableaux}([\text{'}-(p > q)\text{'}, \text{'}(pOq)\text{'}, \text{'}-p\text{'}])$$

$$n.\text{tiene\_lit\_comp}() = \texttt{False}$$

nodo_tableaux — métodos

es_hoja

`self`.es_hoja() retorna:

- Cerrada si tiene literales complementarios,
- Abierta si no tiene literales complementarios y no tiene reglas ni alfa ni beta,
- `None` en otro caso.

Ejemplo:

$$n = \text{nodo\_tableaux}([{}'{-}(p > q)', '(pOq)', '{-}p'])$$

$$n.\text{es\_hoja}() = \texttt{None}$$

nodo_tableaux — métodos

interp

`self`.interp() retorna un diccionario que hace verdaderos a sus literales.

Ejemplo:

$$n = \text{nodo\_tableaux}([{}'{-}(p > q)', '(pOq)', '{-}p'])$$

$$n.\text{interp}() = \{'p' : \texttt{False}\}$$

expandir

`self`.expandir() retorna un nodo_tableaux que es el resultado de aplicar la primera regla α. Si no hay reglas α, retorna los dos nodo_tableaux que son el resultado de aplicar la primera regla β. Si no hay reglas β, retorna `None`, `None`.

Ejemplo:

$$n = \text{nodo\_tableaux}([' {-}(p > q)',' (pOq)',' {-}p'])$$

$$n.\text{expandir}() = \text{nodo\_tableaux}([' (pOq)',' p',' {-}q',' {-}p'])$$

Tipos de listas

First In First Out

FIFO =

Tipos de listas

First In First Out

$$\text{FIFO} = \underset{\substack{\uparrow \\ 1}}{\boxed{\text{A}}}$$

ADD(FIFO, A)

Tipos de listas

First In First Out

$$\text{FIFO} = \boxed{\text{A}}\boxed{\text{B}}$$
$$\quad\quad\quad\quad\;\underset{1}{\uparrow}\;\underset{2}{\uparrow}$$

ADD(FIFO, B)

Tipos de listas

First In First Out

$$\text{FIFO} = \underset{\substack{\uparrow\\1}}{\boxed{\text{A}}}\underset{\substack{\uparrow\\2}}{\boxed{\text{B}}}\boxed{\cdots}\underset{\substack{\uparrow\\n-1}}{\boxed{\text{Y}}}$$

ADD(FIFO, Y)

Tipos de listas

First In First Out

$$\text{FIFO} = \boxed{\underset{\substack{\uparrow\\1}}{\text{A}}}\boxed{\underset{\substack{\uparrow\\2}}{\text{B}}}\boxed{\cdots}\boxed{\underset{\substack{\uparrow\\n-1}}{\text{Y}}}\boxed{\underset{\substack{\uparrow\\n}}{\text{Z}}}$$

ADD(FIFO, Z)

Tipos de listas

First In First Out

$$\text{FIFO} = \underset{1}{\underset{\uparrow}{\boxed{\text{A}}}}\underset{2}{\underset{\uparrow}{\boxed{\text{B}}}}\boxed{\cdots}\underset{n-1}{\underset{\uparrow}{\boxed{\text{Y}}}}\underset{n}{\underset{\uparrow}{\boxed{\text{Z}}}}$$

$s \leftarrow \text{POP(FIFO)}$

Tipos de listas

First In First Out

$$\text{FIFO} = \underset{\substack{\uparrow\\ 1}}{\boxed{\text{B}}}\boxed{\cdots}\underset{\substack{\uparrow\\ n-2}}{\boxed{\text{Y}}}\underset{\substack{\uparrow\\ n-1}}{\boxed{\text{Z}}}$$

$s \leftarrow\ \text{A}$

Tipos de listas

First In First Out

$$\text{FIFO} = \boxed{\text{B}}\boxed{\cdots}\boxed{\text{Y}}\boxed{\text{Z}}$$

$$\underset{1}{\uparrow} \quad \underset{n-2}{\uparrow} \; \underset{n-1}{\uparrow}$$

$s \leftarrow \text{A}$

Last In First Out

$\text{LIFO} =$

Tipos de listas

First In First Out

$$\text{FIFO} = \underset{\substack{\uparrow \\ 1}}{\boxed{\text{B}}}\boxed{\cdots}\underset{\substack{\uparrow \\ n-2}}{\boxed{\text{Y}}}\underset{\substack{\uparrow \\ n-1}}{\boxed{\text{Z}}}$$

$s \leftarrow$ A

Last In First Out

$$\text{LIFO} = \underset{\substack{\uparrow \\ 1}}{\boxed{\text{A}}}$$

ADD(FIFO, A)

Tipos de listas

First In First Out

$$\text{FIFO} = \begin{array}{|c|c|c|c|}\hline \text{B} & \cdots & \text{Y} & \text{Z} \\ \hline \end{array}$$

$$\begin{array}{cccc} \uparrow & & \uparrow & \uparrow \\ 1 & & {\scriptstyle n-2} & {\scriptstyle n-1} \end{array}$$

$s \leftarrow$ A

Last In First Out

$$\text{LIFO} = \begin{array}{|c|c|}\hline \text{A} & \text{B} \\ \hline \end{array}$$

$$\begin{array}{cc} \uparrow & \uparrow \\ 1 & 2 \end{array}$$

ADD(FIFO, B)

Tipos de listas

First In First Out

$$\text{FIFO} = \underset{\substack{\uparrow\\1}}{\boxed{\text{B}}}\boxed{\cdots}\underset{\substack{\uparrow\\n-2}}{\boxed{\text{Y}}}\underset{\substack{\uparrow\\n-1}}{\boxed{\text{Z}}}$$

$s \leftarrow$ A

Last In First Out

$$\text{LIFO} = \underset{\substack{\uparrow\\1}}{\boxed{\text{A}}}\underset{\substack{\uparrow\\2}}{\boxed{\text{B}}}\boxed{\cdots}\underset{\substack{\uparrow\\n-1}}{\boxed{\text{Y}}}$$

ADD(FIFO, Y)

Tipos de listas

First In First Out

$$\text{FIFO} = \boxed{\text{B}}\boxed{\cdots}\boxed{\text{Y}}\boxed{\text{Z}}$$

$$\underset{1}{\uparrow} \qquad \underset{n-2}{\uparrow} \; \underset{n-1}{\uparrow}$$

$s \leftarrow$ A

Last In First Out

$$\text{LIFO} = \boxed{\text{A}}\boxed{\text{B}}\boxed{\cdots}\boxed{\text{Y}}\boxed{\text{Z}}$$

$$\underset{1}{\uparrow} \; \underset{2}{\uparrow} \qquad \underset{n-1}{\uparrow} \; \underset{n}{\uparrow}$$

ADD(FIFO, Z)

Tipos de listas

First In First Out

$$\mathrm{FIFO} = \begin{array}{|c|c|c|c|}\hline \mathrm{B} & \cdots & \mathrm{Y} & \mathrm{Z} \\ \hline \end{array}$$
$$\begin{array}{cccc} \uparrow & & \uparrow & \uparrow \\ 1 & & {\scriptstyle n-2} & {\scriptstyle n-1} \end{array}$$

$s \leftarrow\ \mathrm{A}$

Last In First Out

$$\mathrm{LIFO} = \begin{array}{|c|c|c|c|c|}\hline \mathrm{A} & \mathrm{B} & \cdots & \mathrm{Y} & \mathrm{Z} \\ \hline \end{array}$$
$$\begin{array}{ccccc} \uparrow & \uparrow & & \uparrow & \uparrow \\ 1 & 2 & & n-1 & n \end{array}$$

$s \leftarrow \mathrm{POP(LIFO)}$

Tipos de listas

First In First Out

$$\text{FIFO} = \boxed{\text{B}}\boxed{\cdots}\boxed{\text{Y}}\boxed{\text{Z}}$$

$$\underset{1}{\uparrow} \quad \underset{n-2}{\uparrow} \quad \underset{n-1}{\uparrow}$$

$s \leftarrow$ A

Last In First Out

$$\text{LIFO} = \boxed{\text{A}}\boxed{\text{B}}\boxed{\cdots}\boxed{\text{Y}}$$

$$\underset{1}{\uparrow} \quad \underset{2}{\uparrow} \quad \underset{n-1}{\uparrow}$$

$s \leftarrow$ Z

Presentación

Pseudocódigo

```
función primero_anchura(nodo):
    estado ← nodo
    Si estado es hoja cerrada
            retornar None
    Si no, si estado es hoja abierta
            retornar estado.interp()
    frontera ← lista FIFO [estado]
    Mientras frontera no vacía
            estado ← POP(frontera)
            Para cada hijo en estado.expandir()
                    Si hijo es hoja abierta
                        retornar hijo.interp()
                    Si no, si hijo no es hoja
                        ADD(frontera, hijo)
    retornar None
```

Algoritmo primero en anchura

A: $\{(p \wedge q) \vee (r \vee q), \neg q \vee \neg p\}$

Frontera

FIFO $=$

Algoritmo primero en anchura

A: $\{(p \wedge q) \vee (r \vee q), \neg q \vee \neg p\}$

Frontera

$\text{FIFO} = \boxed{\text{A}}$

Algoritmo primero en anchura

A: $\{(p \land q) \lor (r \lor q), \neg q \lor \neg p\}$

Frontera

$\text{FIFO} = \boxed{\text{A}}$

$estado \leftarrow \text{POP(Frontera)}$

Algoritmo primero en anchura

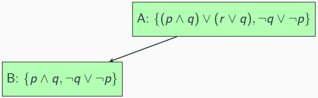
A: $\{(p \wedge q) \vee (r \vee q), \neg q \vee \neg p\}$

Frontera

FIFO $=$

estado $\leftarrow$ A

Algoritmo primero en anchura



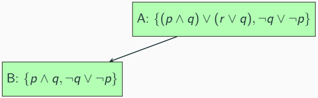
Frontera

FIFO $=$

estado $\leftarrow$ A

hijo $\leftarrow$ B (hoja abierta? No)

Algoritmo primero en anchura



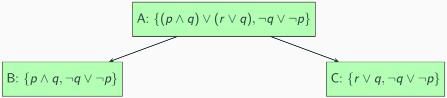
Frontera

$\text{FIFO} = \boxed{\text{B}}$

$\textit{estado} \leftarrow \text{A}$

ADD(*frontera*, B)

Algoritmo primero en anchura



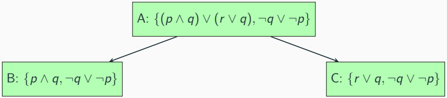
Frontera

$\text{FIFO} = \boxed{\text{B}}$

estado $\leftarrow$ A

hijo $\leftarrow$ C (hoja abierta? No)

Algoritmo primero en anchura



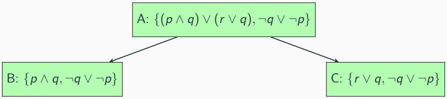
Frontera

$\text{FIFO} = \boxed{\text{B} \mid \text{C}}$

estado $\leftarrow$ A

ADD(*frontera*, C)

Algoritmo primero en anchura

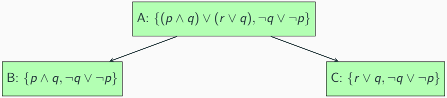


Frontera

$\text{FIFO} = \boxed{\text{B} \mid \text{C}}$

estado $\leftarrow$ POP(Frontera)

Algoritmo primero en anchura

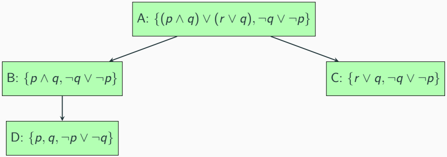


Frontera

$\text{FIFO} = \boxed{\text{C}}$

$estado \leftarrow \text{B}$

Algoritmo primero en anchura



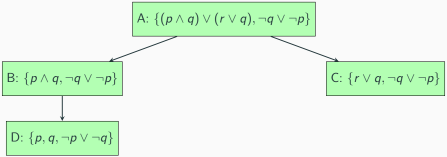
Frontera

$\text{FIFO} = \boxed{\text{C}}$

estado $\leftarrow$ B

hijo $\leftarrow$ D (hoja abierta? No)

Algoritmo primero en anchura



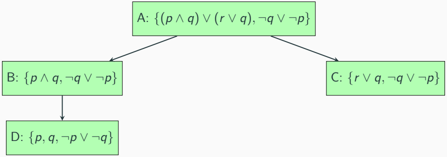
Frontera

$\text{FIFO} = \boxed{\text{C}}\boxed{\text{D}}$

estado $\leftarrow$ B

ADD(*frontera*, D)

Algoritmo primero en anchura

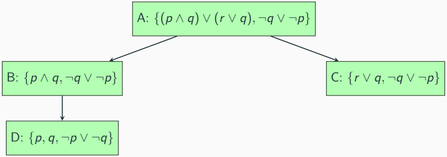


Frontera

$$\text{FIFO} = \boxed{\text{C}}\boxed{\text{D}}$$

$estado \leftarrow \text{POP(Frontera)}$

Algoritmo primero en anchura

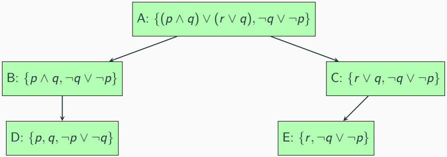


Frontera

$\text{FIFO} = \boxed{\text{D}}$

$\textit{estado} \leftarrow \text{C}$

Algoritmo primero en anchura



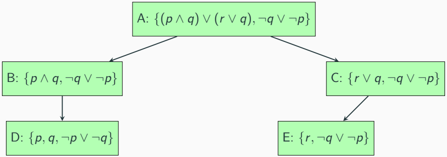
Frontera

$\text{FIFO} = \boxed{\text{D}}$

estado $\leftarrow$ C

hijo $\leftarrow$ E (hoja abierta? No)

Algoritmo primero en anchura



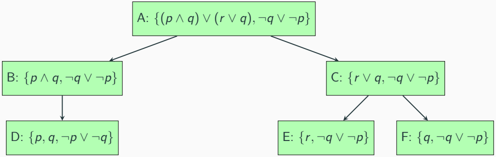
Frontera

$\mathrm{FIFO} = \boxed{\mathrm{D}}\boxed{\mathrm{E}}$

estado $\leftarrow$ C

ADD(*frontera*, E)

Algoritmo primero en anchura

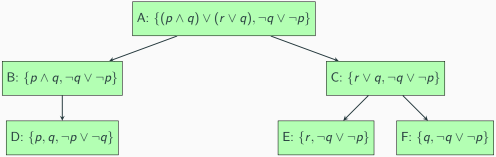


Frontera

FIFO $= \boxed{\text{D}}\boxed{\text{E}}$

hijo $\leftarrow$ F (hoja abierta? No)

Algoritmo primero en anchura

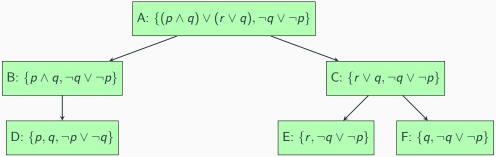


Frontera

$$\text{FIFO} = \boxed{\text{D}}\boxed{\text{E}}\boxed{\text{F}}$$

ADD(*frontera*, F)

Algoritmo primero en anchura

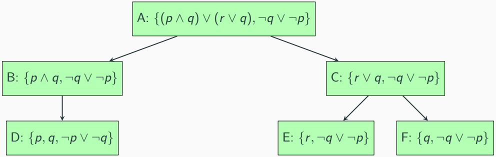


Frontera

$$\text{FIFO} = \begin{array}{|c|c|c|}\hline \text{D} & \text{E} & \text{F} \\ \hline\end{array}$$

$\textit{estado} \leftarrow \text{POP}(\text{Frontera})$

Algoritmo primero en anchura

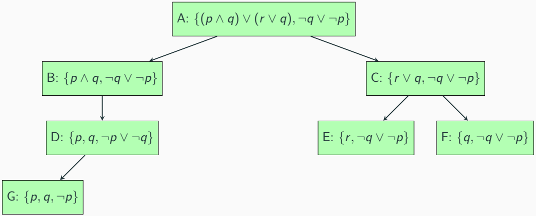


Frontera

$$\mathrm{FIFO} = \boxed{\mathrm{E} \mid \mathrm{F}}$$

$\mathit{estado} \leftarrow \mathrm{D}$

Algoritmo primero en anchura



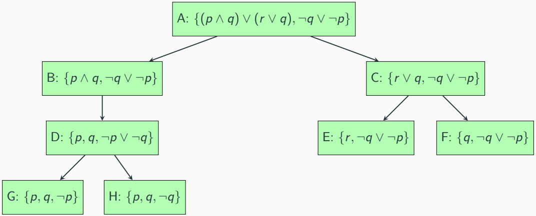
Frontera

$\text{FIFO} = \boxed{\text{E}}\boxed{\text{F}}$

estado $\leftarrow$ D

hijo $\leftarrow$ G (hoja cerrada)

Algoritmo primero en anchura



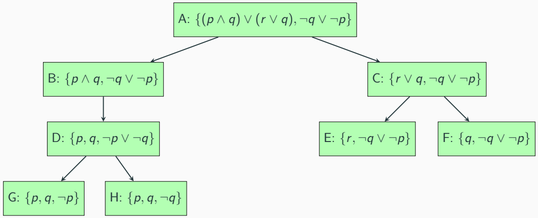
Frontera

$\text{FIFO} = \boxed{\text{E} \mid \text{F}}$

estado $\leftarrow$ D

hijo $\leftarrow$ H (hoja cerrada)

Algoritmo primero en anchura

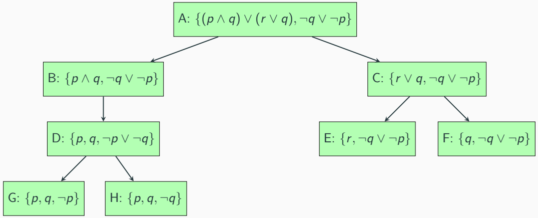


Frontera

$\text{FIFO} = \boxed{\text{E} \mid \text{F}}$

$\textit{estado} \leftarrow \text{POP(Frontera)}$

Algoritmo primero en anchura

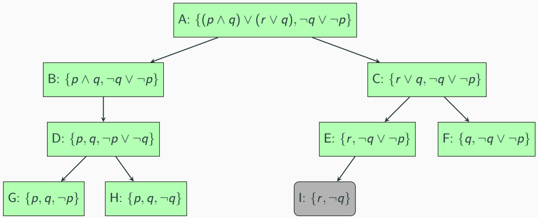


Frontera

$\text{FIFO} = \boxed{\text{F}}$

$estado \leftarrow \text{E}$

Algoritmo primero en anchura



Frontera

FIFO = $\boxed{\text{F}}$

estado $\leftarrow$ E

hijo $\leftarrow$ I (¡hoja abierta!)

Presentación

Pseudocódigo

```
función primero_profundidad(nodo):
    estado ← nodo
    Si estado es hoja cerrada
            retornar None
    Si no, si estado es hoja abierta
            retornar estado.interp()
    frontera ← lista LIFO [estado]
    Mientras frontera no vacía
            estado ← POP(frontera)
            Para cada hijo en estado.expandir()
                    Si hijo es hoja abierta
                        retornar hijo.interp()
                    Si no ADD(frontera, hijo)
    retornar None
```

Algoritmo primero en profundidad

A: $\{(p \wedge q) \vee (r \vee q), \neg q \vee \neg p\}$

Frontera

LIFO $=$

Algoritmo primero en profundidad

A: $\{(p \wedge q) \vee (r \vee q), \neg q \vee \neg p\}$

Frontera

$\text{LIFO} = \boxed{\text{A}}$

Algoritmo primero en profundidad

A: $\{(p \wedge q) \vee (r \vee q), \neg q \vee \neg p\}$

Frontera

$\text{LIFO} = \boxed{\text{A}}$

$estado \leftarrow \text{POP(Frontera)}$

Algoritmo primero en profundidad

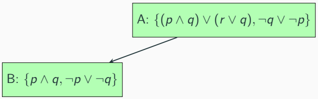
A: $\{(p \wedge q) \vee (r \vee q), \neg q \vee \neg p\}$

Frontera

LIFO $=$

$estado \leftarrow$ A

Algoritmo primero en profundidad



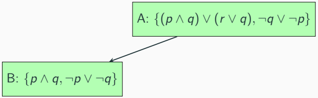
Frontera

LIFO $=$

estado $\leftarrow$ A

hijo $\leftarrow$ B (hoja abierta? No)

Algoritmo primero en profundidad



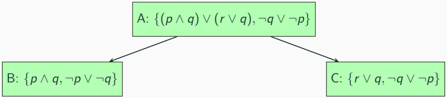
Frontera

LIFO $= \boxed{\text{B}}$

estado $\leftarrow$ A

ADD(*frontera*, B)

Algoritmo primero en profundidad



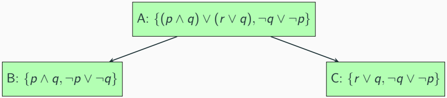
Frontera

LIFO $= \boxed{\text{B}}$

estado $\leftarrow$ A

hijo $\leftarrow$ C (hoja abierta? No)

Algoritmo primero en profundidad



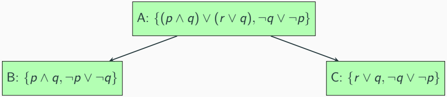
Frontera

$$\text{LIFO} = \boxed{\text{B} \mid \text{C}}$$

estado $\leftarrow$ A

ADD(*frontera*, C)

Algoritmo primero en profundidad

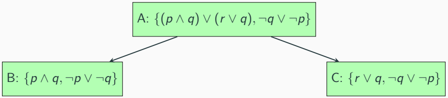


Frontera

$\text{LIFO} = \boxed{\text{B}}\boxed{\text{C}}$

$estado \leftarrow \text{POP(Frontera)}$

Algoritmo primero en profundidad

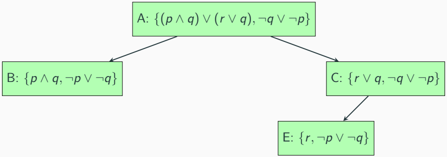


Frontera

$\text{LIFO} = \boxed{\text{B}}$

estado $\leftarrow$ C

Algoritmo primero en profundidad



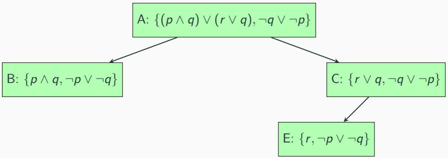
Frontera

$\text{LIFO} = \boxed{\text{B}}$

estado $\leftarrow$ C

hijo $\leftarrow$ E (hoja abierta? No)

Algoritmo primero en profundidad



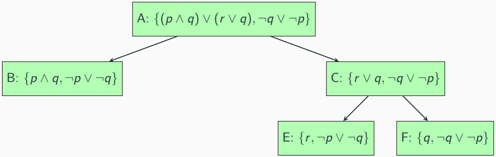
Frontera

LIFO = | B | E |

estado $\leftarrow$ C

ADD(*frontera*, E)

Algoritmo primero en profundidad



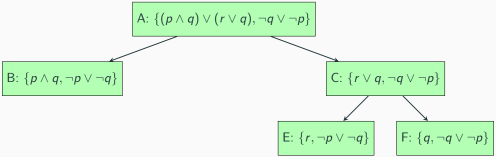
Frontera

$\text{LIFO} = \boxed{\text{B} \mid \text{E}}$

estado $\leftarrow$ C

hijo $\leftarrow$ F (hoja abierta? No)

Algoritmo primero en profundidad



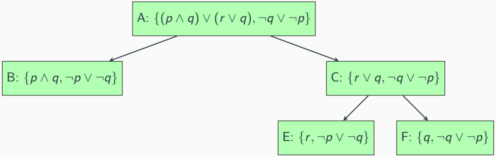
Frontera

$\text{LIFO} = \boxed{\text{B}}\boxed{\text{E}}\boxed{\text{F}}$

estado $\leftarrow$ C

ADD(*frontera*, F)

Algoritmo primero en profundidad

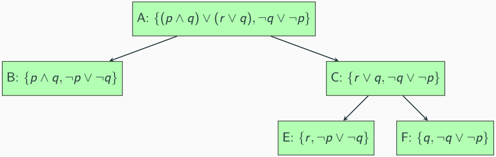


Frontera

$$\text{LIFO} = \boxed{\text{B}}\boxed{\text{E}}\boxed{\text{F}}$$

$\textit{estado} \leftarrow \text{POP(Frontera)}$

Algoritmo primero en profundidad

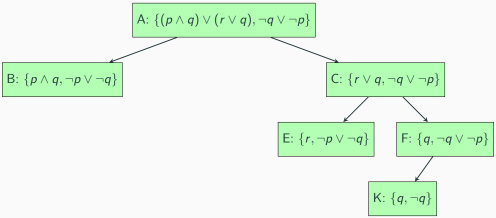


Frontera

$$\text{LIFO} = \boxed{\text{B}}\boxed{\text{E}}$$

estado $\leftarrow$ F

Algoritmo primero en profundidad



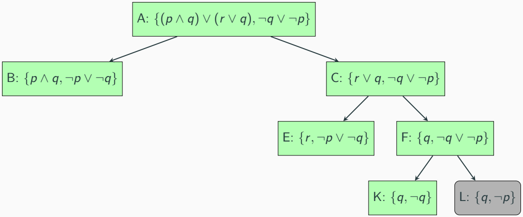
Frontera

LIFO = | B | E |

estado $\leftarrow$ F

hijo $\leftarrow$ K (hoja cerrada)

Algoritmo primero en profundidad



Frontera

LIFO $= \boxed{\text{B}}\boxed{\text{E}}$

hijo $\leftarrow$ L (¡hoja abierta!)

Presentación

Pseudocódigo

```
función backtracking(nodo):
    estado ← nodo
    Si estado es hoja cerrada
        retornar None
    Si no, si estado es hoja abierta
        retornar estado.interp()
    Para cada hijo en estado.expandir()
        resultado ← backtracking(hijo)
        Si resultado no es None
          retornar resultado
    retornar None
```

Algoritmo backtracking

$\boxed{\text{A: } \{(p \wedge q) \vee (r \vee q), \neg q \vee \neg p\}}$

backtracking(A) $=$?

Algoritmo backtracking

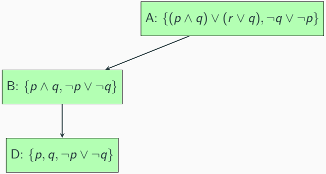
A: $\{(p \wedge q) \vee (r \vee q), \neg q \vee \neg p\}$

B: $\{p \wedge q, \neg p \vee \neg q\}$

backtracking(A) $=$?

backtracking(B) $=$?

Algoritmo backtracking

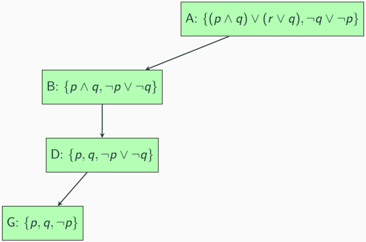


backtracking(A) $=$?

backtracking(B) $=$?

backtracking(D) $=$?

Algoritmo backtracking



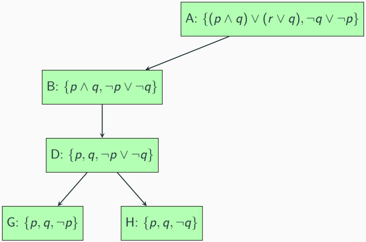
backtracking(A) $=$?

backtracking(B) $=$?

backtracking(D) $=$?

backtracking(G) $=$ None

Algoritmo backtracking



backtracking(A) = ?

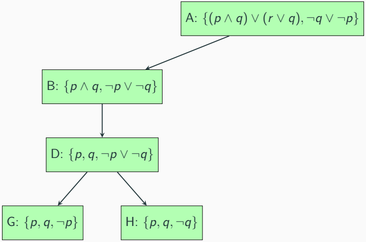
backtracking(B) = ?

backtracking(D) = ?

backtracking(G) = None

backtracking(H) = None

Algoritmo backtracking

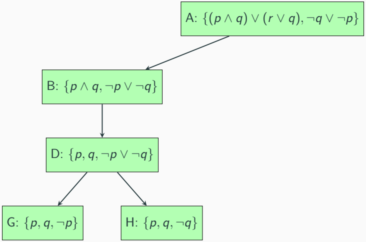


backtracking(A) = ?

backtracking(B) = ?

backtracking(D) = None

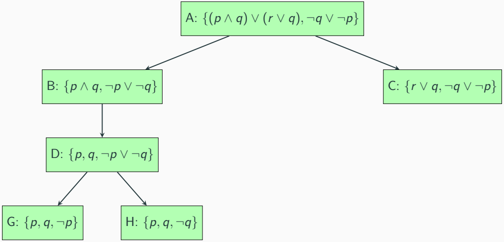
Algoritmo backtracking



backtracking(A) $=$?

backtracking(B) $=$ None

Algoritmo backtracking

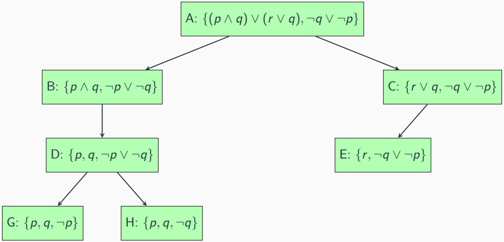


backtracking(A) = ?

backtracking(B) = None

backtracking(C) = ?

Algoritmo backtracking



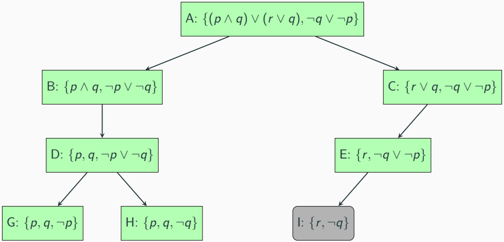
backtracking(A) = ?

backtracking(B) = None

backtracking(C) = ?

backtracking(E) = ?

Algoritmo backtracking



backtracking(A) $=$?

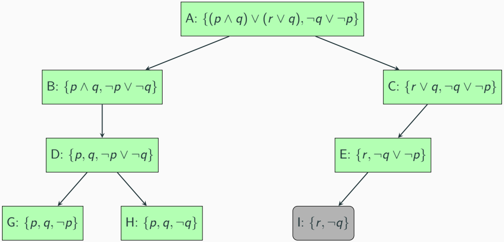
backtracking(B) $=$ None

backtracking(C) $=$?

backtracking(E) $=$?

backtracking(I) $= \{'q' : \texttt{False}, 'r' : \texttt{True}\}$

Algoritmo backtracking



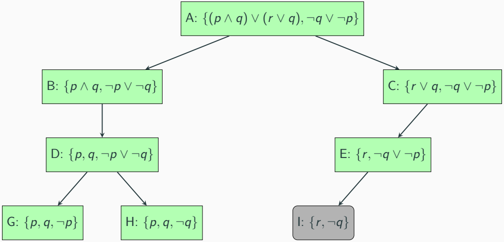
backtracking(A) $=$?

backtracking(B) $=$ None

backtracking(C) $=$?

backtracking(E) $= \{'q' : \texttt{False}, 'r' : \texttt{True}\}$

Algoritmo backtracking

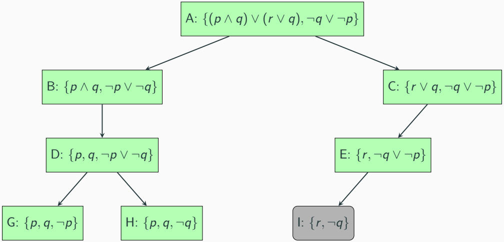


backtracking(A) = ?

backtracking(B) = None

backtracking(C) = $\{'q' : \texttt{False}, 'r' : \texttt{True}\}$

Algoritmo backtracking



$$\text{backtracking(A)} = \ \{'q' : \texttt{False}, 'r' : \texttt{True}\}$$

Fin de la sesión

En esta sesión usted ha aprendido a:

1. Visualizar el algoritmo de generación de tableaux mediante tres algoritmos: búsqueda en anchura, búsqueda en profundidad y backtracking.
2. Comparar empíricamente los tiempos de ejecución de los tres algoritmos de búsqueda.