Estructuras Discretas 2023-1

Práctica 5: Tableaux.

Lourdes del Carmen González Huesca
María Fernanda Mendoza Castillo
Juan Alfonso Garduño Solís
Alan Moreno de la Rosa

7 noviembre de 2022
Fecha de entrega: Sábado 19 de noviembre de 2022

Introducción.

Un tableau corresponde a un árbol cuya función es buscar una interpretación para una determinada fórmula, este mecanismo permite de manera eficiente y segura determinar si una fórmula es tautlogía, contradicción o contingencia.

Implementación.

Como mencionamos antes, un tableau es un árbol, cuyos nodos están etiquetados con una fórmula: el nodo raíz está etiquetado por la fomula original, mientras que los nodos internos con subfórmulas obtenidas a partir de las reglas α, β, σ.
Para nuestra implementación los nodos estarán etiquetados con listas de fórmulas generadas mediantes las reglas de expansión α, β, σ, de la manera descrita a continuación:

- **Reglas** α

Teoría

$\alpha_1 \wedge \alpha_2$

$\downarrow$

α_1

$\downarrow$

α_2

Práctica

$[(\alpha_1 \wedge \alpha_2) : xs]$

$\downarrow$

$[\alpha_1 : \alpha_2 : xs]$

$\neg(\alpha_1 \vee \alpha_2)$

$\downarrow$

$\neg\alpha_1$

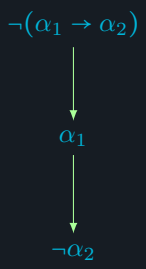
$\downarrow$

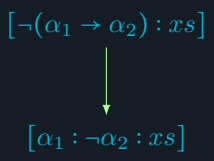
$\neg\alpha_2$

$[\neg(\alpha_1 \vee \alpha_2) : xs]$

$\downarrow$

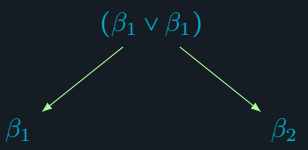
$[\neg\alpha_1 : \neg\alpha_2 : xs]$

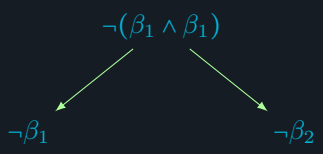


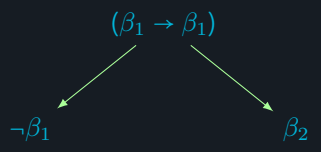


- **Reglas β**

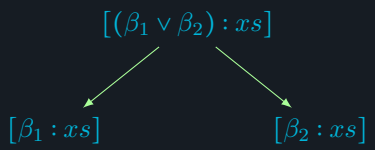
Teoría

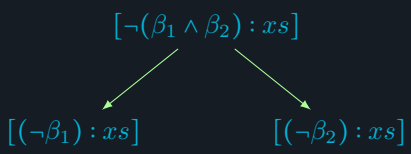


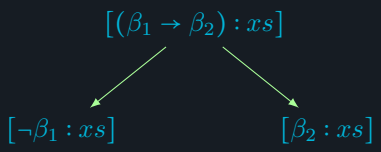




Práctica



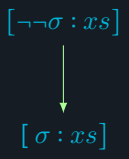


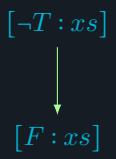


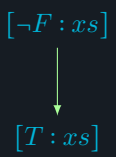
- **Reglas σ**

Teoría

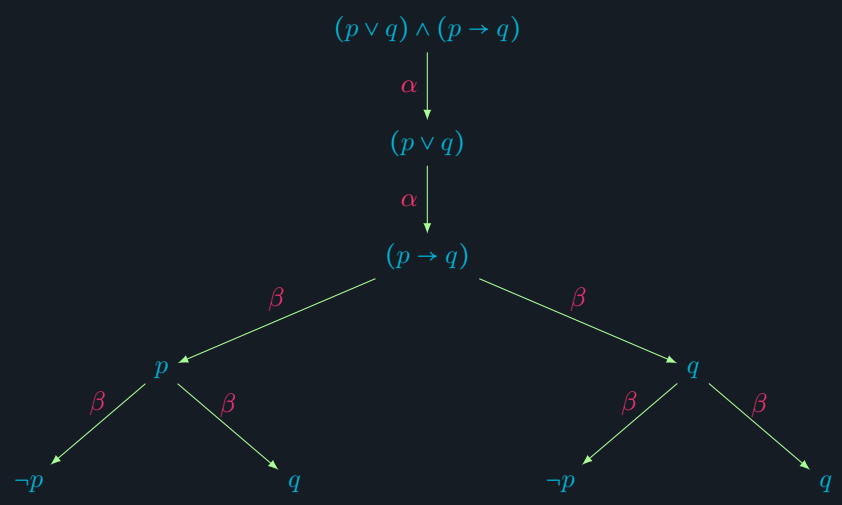
Práctica



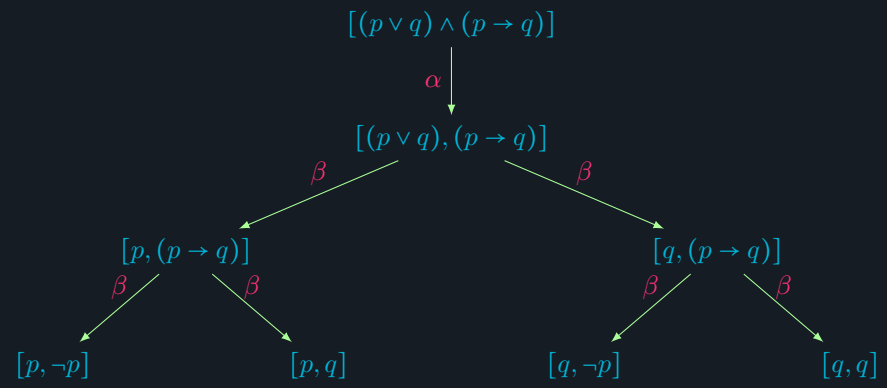




Es importante notar que si bien los esquemas anteriores muestran una expansión en listas unitarias de fórmulas, pueden ser aplicadas a un elemento dentro de cualquier lista de fórmulas. Esto quedará más claro con el siguiente ejemplo.
En la teoría la fórmula $(p \vee q) \wedge (p \to q)$ genera el siguiente Tableaux:



Este árbol corresponde en la implementación al siguiente árbol.



Como podemos observar, en nuestra implementación las expansiones consisten en modificar la lista que está en el nodo pardre, empezando por la raíz.
Los siguientes tipos de datos representarán las fomulas y los tableaux. Con base en ellos, realiza los ejercicios que se piden.

```
--Alias para variables
type Nombre = String
--Tipo para las proposiciones
data LProp = T | F | VarP Nombre | Conj Lprop LProp | Disy LProp LProp |
           Impl LProp LProp | Syss LProp LProp | Neg LProp deriving (Show, Eq)
--Tipo para el Tableaux
data Tableaux = Hoja [LProp] | Alpha [LProp] Tableaux |
                Beta [LProp] Tableaux Tableaux deriving (Show, Eq)
```

Ejercicios.

Puedes utilizar tu instancia de la clase show de la práctica pasada.

literales ***(1 punto)*** Función que nos dice si en una lista de fórmulas, todas son literales.

```
literales [VarP "s", Neg (VarP "q")] = True
literales [VarP "s", (Impl (VarP "q") (VarP "s"))] = False
```

nextF ***(1 punto)*** Función que regresa la primera fórmula que no es literal de una lista de fórmulas.

```
nextF [(Conj (VarP "p") (varP "q")), Neg (VarP "r")] = Neg (VarP "r")
```

alpha ***(1 punto)*** Nos dice si una fórmula f es una fórmula α

```
alpha (Conj (VarP "p") (VarP "q")) = True
alpha (Disy (VarP "p") (Conj (VarP "r") (VarP "s"))) = False
alpha (Neg (Neg (Conj (VarP "p") (VarP "q")))) = False
```

beta ***(1 punto)*** Nos dice si una fórmula f es una fórmula β

```
beta (Conj (VarP "p") (VarP "q")) = False
beta (Disy (VarP "p") (Conj (VarP "r") (VarP "s"))) = True
beta (Neg (Neg (Conj (VarP "p") (VarP "q")))) = False
```

sigma ***(1 punto)*** Nos dice si una fórmula f es una fórmula σ

```
sigma (Conj (VarP "p") (VarP "q")) = False
sigma (Disy (VarP "p") (Conj (VarP "r") (VarP "s"))) = False
sigma (Neg (Neg (Conj (VarP "p") (VarP "q")))) = True
```

expAlpha ***(1 punto)*** Dada una lista de fórmulas l y una fórmula f realiza la expansión alpha de f dentro la lista l.

```
expAlpha [(Conj (VarP "p") (VarP "q")), Neg (VarP "r")] (Conj(VarP "p") (VarP "q"))
= [(VarP "p"), (VarP "q"), Neg (VarP "r")]
= [p,q,!(r)]
```

expBeta ***(1 punto)*** Dada una lista de fórmulas l y una fórmula f realiza la expansión beta de f spbre la lista l.

```
expBeta [VarP "f", (Disy (VarP "p") (VarP "q")), VarP "r"] (Disy(VarP "p") (VarP "q"))
= ([VarP "p", VarP "f", VarP "r"], [VarP "q", VarP "f", VarP "r"])
= ([p,r],[q,r])
```

expSigma ***(1 punto)*** Dada una lista de fórmulas l y una fórmula f, realiza la expansión sigma de f sobre la lista l.

```
expSigma [Neg(Neg(Conj (VarP "p") (VarP "q")))] (Neg(Neg(Conj(VarP "p") (VarP "q"))))
= [(Conj (VarP "p") (VarP "q"))]
= [(p ^ q)]
```

consTableaux ***(3 puntos)*** Construye el tableau a partir de una fórmula.

```
consTableaux (Disy (VarP "p") (Impl (Conj (VarP "p") (Neg (VarP "q"))) (VarP "r")))
= Beta [Disy (VarP "p") (Imp (Conj (VarP"p") (Neg (VarP"q"))) (VarP "r"))]
        (Hoja [VarP"p"])
        (Beta [Imp (Conj (VarP "p") (Neg (VarP "q"))) (VarP "r")]
        (Beta [Neg (Conj (VarP "p") (Neg (VarP "q")))]
        (Hoja [Neg (VarP "p")])
        (Hoja [VarP "q"]))
        (Hoja [VarP "r"]))
= Beta [(p v ((p ^ !(q)) -> r))]
        (Hoja [p])
        (Beta [((p ^ !(q)) -> r)]
                (Beta [!((p ^ !(q)))]
                    (Hoja [!(p)])
                    (Hoja [q]))
                (Hoja [r]))
```