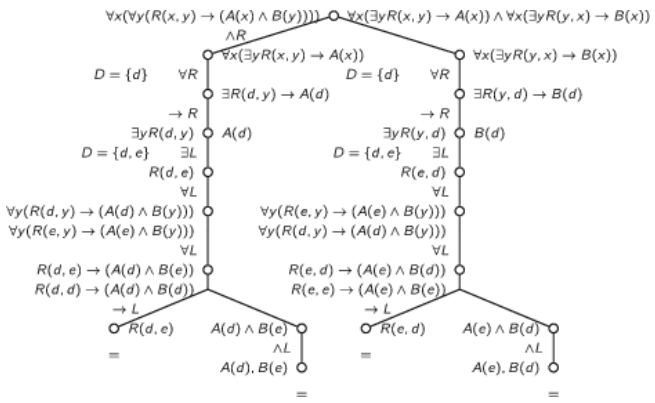
Tema 2:
Tableros Semánticos



Joaquín Borrego Díaz, Fernando Sancho Caparrini

Lógica Informática – Tecnologías Informáticas
Dpto. Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla

6 de octubre de 2022

Contenido

Introducción

Tableros en LP

Fórmulas α y β

Tableros completos

Búsqueda de modelos

Consecuencia lógica

Tableros en LPO

Fórmulas δ y γ

Tableros completos

Búsqueda de modelos

Consecuencia lógica

Razonamiento con igualdad

Razonamiento con igualdad

Sistemas

Referencias

Introducción

- Algoritmo para estudiar la satisfactibilidad de un conjunto de fórmulas proposicionales y de primer orden.
- Trabaja directamente sobre el conjunto de fórmulas, **sin preprocesamiento**.
- Basado en la sintaxis de las fórmulas.
- Reduce la satisfactibilidad de las formulas consideradas a la de ciertos conjuntos de literales, que proporcionan modelos.
- Se representará gráficamente mediante un árbol binario.
- Es muy flexible y puede adaptarse a otras lógicas (descriptivas, modales, etc.).

Tableros semánticos en LP

El método de tableros semánticos en LP:

1. Clasifica las fórmulas en dos clases:
   - Las fórmulas α, que se comportan como conjunciones.
   - Las fórmulas β, que se comportan como disyunciones.
2. Asocia a cada fórmula, F, otras dos fórmulas más sencillas (sus **componentes**) de modo que la satisfactibilidad de F se reduce a la de sus componentes.

Fórmulas de tipo α

Las fórmulas de tipo α son las siguientes:

α	α_1	α_2
$\neg\neg F$	F	
$F_1 \wedge F_2$	F_1	F_2
$\neg(F_1 \vee F_2)$	$\neg F_1$	$\neg F_2$
$\neg(F_1 \rightarrow F_2)$	F_1	$\neg F_2$
$F_1 \leftrightarrow F_2$	$F_1 \rightarrow F_2$	$F_2 \rightarrow F_1$

- Las fórmulas α_1 y α_2 son las componentes de α.
- Si F es de tipo α, entonces $F \equiv \alpha_1 \wedge \alpha_2$.
- Para satisfacer una fórmula de tipo α es necesario y suficiente satisfacer **simultáneamente** sus dos componentes α_1 y α_2.

Fórmulas de tipo β

Las fórmulas de tipo β son las siguientes:

β	β_1	β_2
$F_1 \vee F_2$	F_1	F_2
$\neg(F_1 \wedge F_2)$	$\neg F_1$	$\neg F_2$
$(F_1 \rightarrow F_2)$	$\neg F_1$	F_2
$\neg(F_1 \leftrightarrow F_2)$	$\neg(F_1 \rightarrow F_2)$	$\neg(F_2 \rightarrow F_1)$

- Las fórmulas β_1 y β_2 son las componentes de β.
- Si F es de tipo β, entonces $F \equiv \beta_1 \vee \beta_2$
- Para satisfacer una fórmula de tipo β es necesario y suficiente satisfacer sólo una de sus componentes β_1 y β_2.

Reglas α y β

Reducen la consistencia de un conjunto de fórmulas a la de otro conjunto formado por fórmulas más sencillas.

- **Regla** α: Si $F \in U$ es de tipo α, entonces

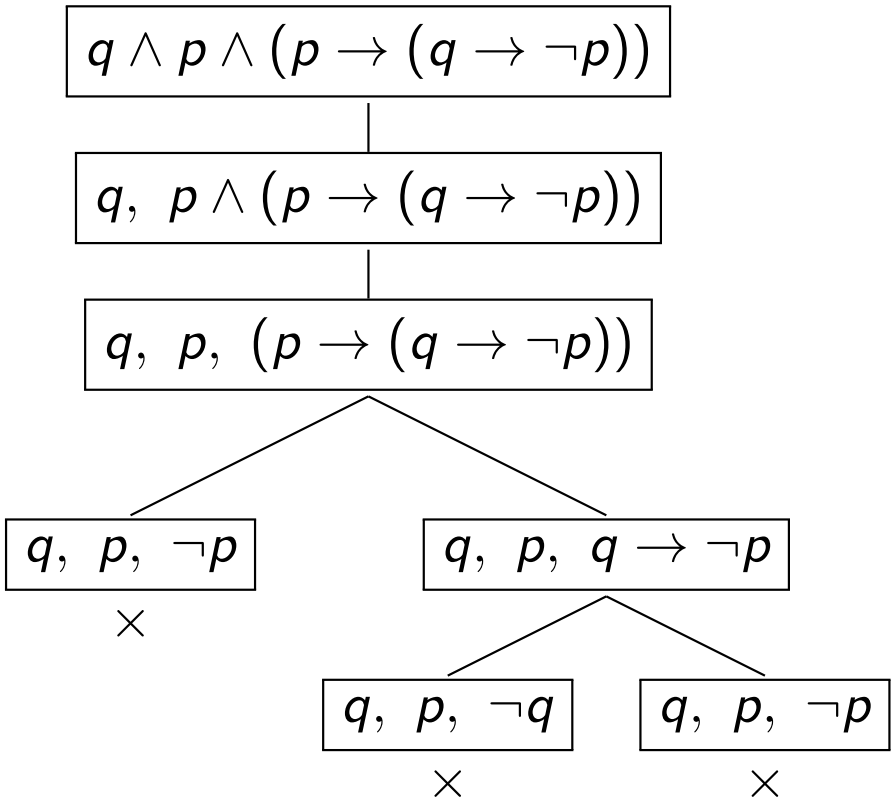
$$U \text{ satisfactible} \iff (U - \{F\}) \cup \{\alpha_1, \alpha_2\} \text{ satisfactible}$$

- **Regla** β: Si $F \in U$ es de tipo β, entonces

$$U \text{ satisfactible} \iff \begin{cases} (U - \{F\}) \cup \{\beta_1\} \text{ satisfactible} \\ \text{o} \\ (U - \{F\}) \cup \{\beta_2\} \text{ satisfactible} \end{cases}$$

Ejemplo

Tablero semántico para la fórmula $q \land p \land (p \to (q \to \neg p))$:



Construcción de un tablero completo

Un **tablero** para $\{A_1, \ldots, A_n\}$ es un árbol T, con nodos etiquetados por conjuntos de fórmulas, que se ha construido siguiendo los siguientes pasos:

- La raíz r de T se etiqueta con $U_r = \{A_1, \ldots, A_n\}$.
- Mientras T tenga hojas no marcadas, seleccionar una hoja n de T, con etiqueta U_n, no marcada, y hacer:
  1. Si U_n es un conjunto de literales, entonces:
     - 1.1 Si existe un par de literales complementarios en U_n, marcar con $\times$ (y se denomina **hoja cerrada**).
     - 1.2 Si no existe tal par, marcar con $\bigcirc$ (**hoja abierta**).
  2. Si U_n no es un conjunto de literales, elegir A de U_n no literal.
     - 2.1 Si A es una α–fórmula, entonces añadir un hijo m a n con $U_m = (U_n \setminus \{A\}) \cup \{\alpha_1, \alpha_2\}$.
     - 2.2 Si A es una β–fórmula, entonces añadir dos hijos n_1, n_2 con etiquetas $U_{n_1} = (U_n \setminus \{A\}) \cup \{\beta_1\}$ y $U_{n_2} = (U_n \setminus \{A\}) \cup \{\beta_2\}$.

Propiedades de los tableros completos

- La construcción siempre termina (se puede probar por inducción).
- El tablero final se denomina **tablero completo**.
- Un tablero se dice **cerrado** si todas sus hojas son cerradas, y **abierto** en caso contrario (basta que una de sus hojas sea abierta).

Propiedades de los tableros completos

Teorema. (Corrección y Completitud)
Sea S un conjunto de fórmulas y T un tablero completo para S.

1. **Corrección**: Si T es cerrado, entonces S es insatisfactible.
2. **Completitud**: Si S es insatisfactible, entonces T es cerrado.

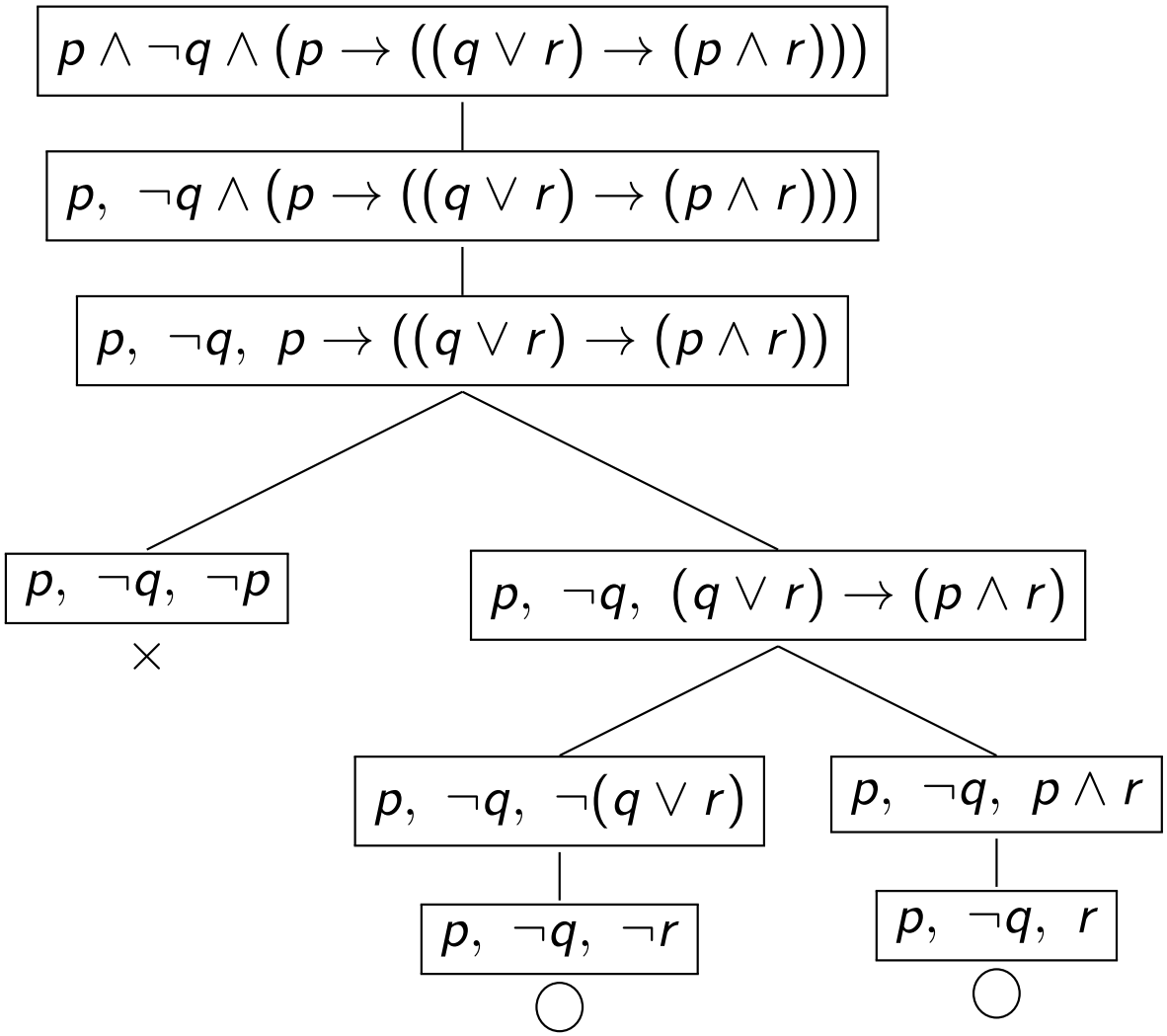
Extrayendo modelos de un tablero completo

- Un conjunto de fórmulas $S = \{A_1, \ldots, A_n\}$ admite un tablero completo abierto si y sólo si es un conjunto satisfactible.
- Además, cada rama abierta del tablero completo define un modelo (no necesariamente distinto) de S de la siguiente forma:

  Si U es la etiqueta de una hoja abierta, podemos obtener un modelo v del conjunto $\{A_1, \ldots, A_n\}$, como sigue

  - *Si $p \in U$, entonces $v(p) = 1$,*
  - *Si $\neg p \in U$, entonces $v(p) = 0$,*
  - *Si $p \notin U$ y $\neg p \notin U$, entonces $v(p)$ puede tomar cualquier valor, 0 o 1.*

Ejemplo



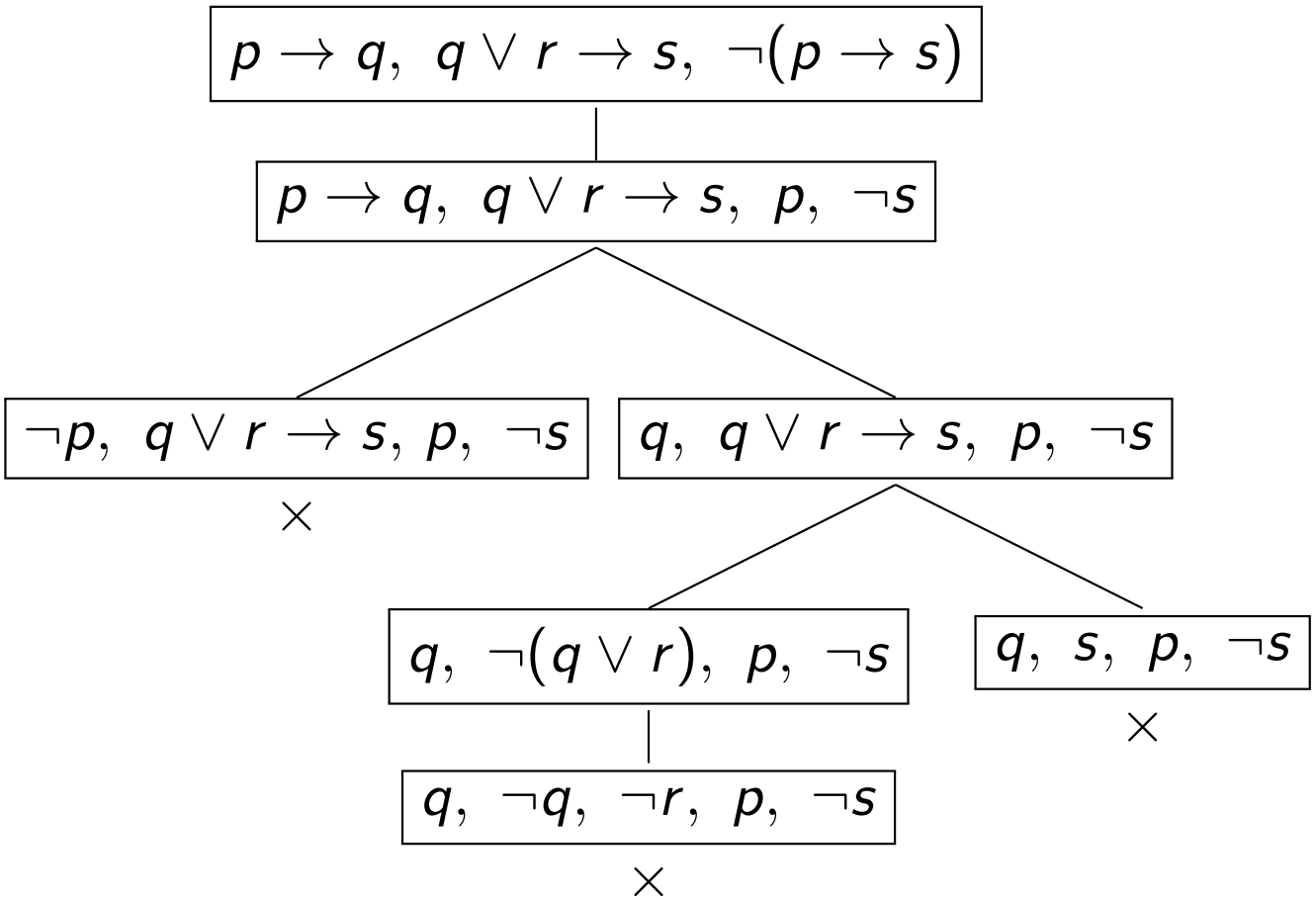
Obtenemos los modelos $v_1(p) = 1,\ v_1(r) = v_1(q) = 0$ y $v_2(p) = v_2(r) = 1,\ v_2(q) = 0$.

Consecuencia lógica

Corolario.

$\{A_1, \ldots A_n\} \models A$ si y solo si $\{A_1, \ldots, A_n, \neg A\}$ admite un tablero cerrado.

Por ejemplo: $\{p \to q, q \vee r \to s\} \models p \to s$.



Tableros Semánticos en LPO

- **IMPORTANTE: SE APLICA A FÓRMULAS CERRADAS**
- Recordemos que:
  - Las fórmulas α se comportan como conjunciones
  - Las fórmulas β se comportan como disyunciones

  y asociamos a cada fórmula, F, de estos tipos otras dos fórmulas más sencillas (sus **componentes**) de modo que la satisfactibilidad de F se reduce a la de sus componentes.
- Para extenderlo a LPO debemos proporcionar formas de trabajar con los cuantificadores: $\exists$ y $\forall$.
- Por ello, introduciremos dos nuevos tipos de fórmulas:
  - Las fórmulas de tipo γ, que se comportan como fórmulas cuantificadas universalmente, y
  - Las fórmulas de tipo δ, que se comportan como fórmulas cuantificadas existencialmente.

Fórmulas de tipo γ

Las fórmulas de tipo γ son las siguientes:

γ	γ_t
$\forall x\, F$	$F[x/t]$ (t es un término cerrado)
$\neg\exists x\, F$	$\neg F[x/t]$ (t es un término cerrado)

- Las fórmulas γ_t son las componentes de γ.
- Para satisfacer una fórmula de tipo γ es necesario satisfacer **simultáneamente** todas sus componentes γ_t, para todo término cerrado t.
- En este caso, dependiendo del lenguaje, puede haber una cantidad infinita de componentes (por ejemplo, basta que el lenguaje tenga un símbolo de constante y un símbolo de función).

Fórmulas de tipo δ

Las fórmulas de tipo δ son las siguientes:

δ	δ_a
$\exists x\, F$	$F[x/a]$ (a es una nueva constante)
$\neg\forall x\, F$	$\neg F[x/a]$ (a es una nueva constante)

- Las fórmulas δ_a son las componentes de δ.
- Para satisfacer una fórmula de tipo δ es necesario y suficiente satisfacer alguna de sus componentes δ_a, para alguna nueva constante a.

Reglas γ y δ

Reducen la consistencia de un conjunto de fórmulas a la de otro conjunto formado por fórmulas más sencillas.

- **Regla** γ: Si $F \in U$ es de tipo γ, entonces

  U consistente $\Leftrightarrow U \cup \{\gamma_t : \ t \text{ término cerrado}\}$ consistente

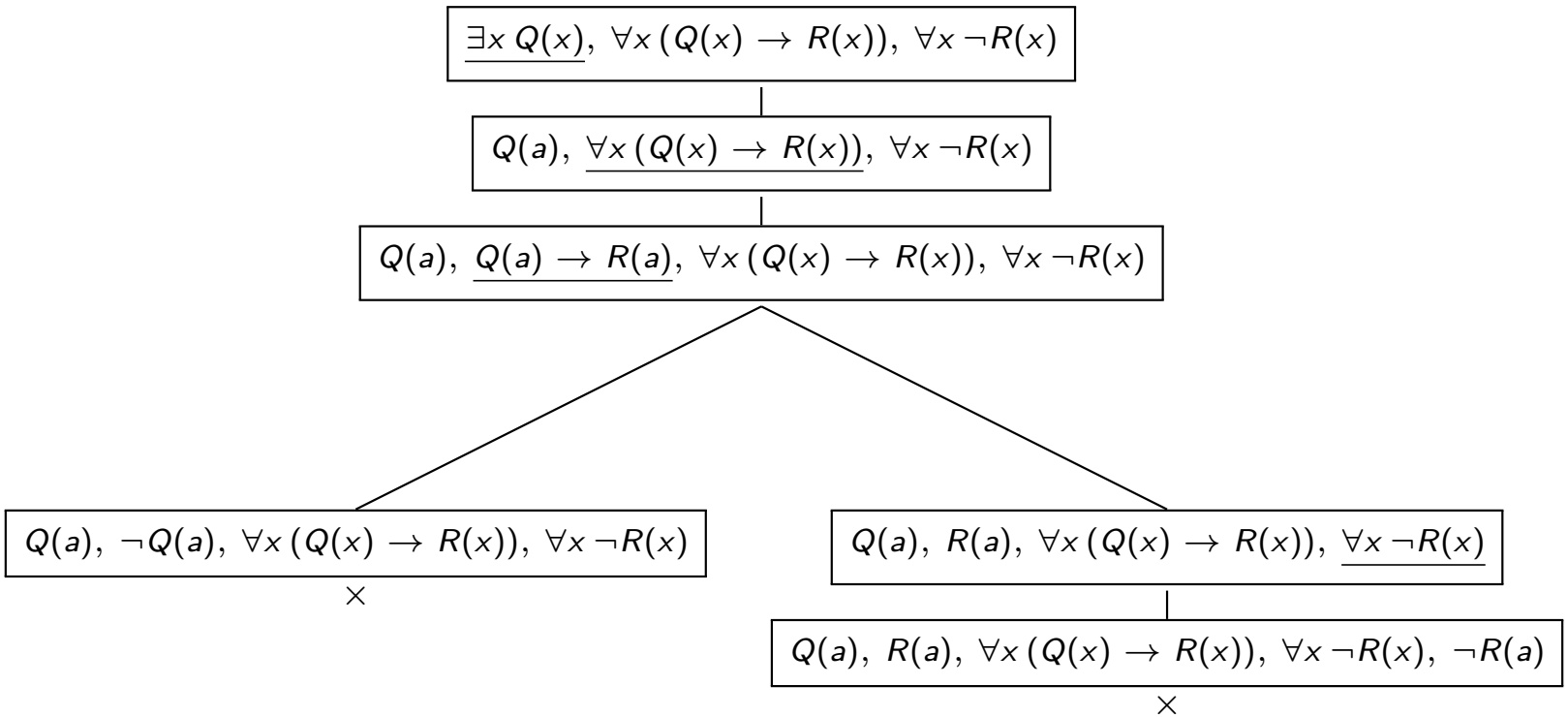
- **Regla** δ: Si $F \in U$ es de tipo δ, entonces para cada constante nueva, a, se tiene:

  U consistente $\quad \Longleftrightarrow \quad (U - \{F\}) \cup \{\delta_a\}$ consistente

Ejemplo

Tablero Semántico para el conjunto de fórmulas

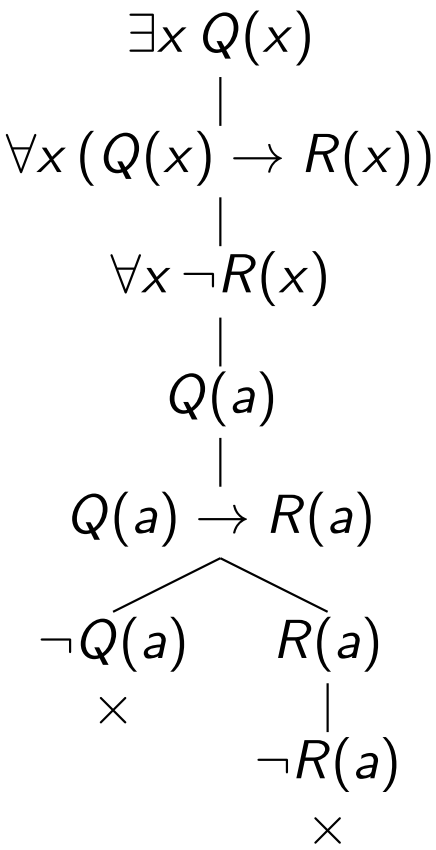
$$U = \{\exists x\, Q(x),\ \forall x\, (Q(x) \to R(x)),\ \forall x\, \neg R(x)\}$$



En cada paso subrayamos la fórmula sobre la que se aplica una regla.

Notación

En LPO representaremos los tableros etiquetando cada nodo con una fórmula. El tablero anterior se escribirá:



Notación: Si n es un nodo de T, E_n es la etiqueta de n (una sola fórmula), y U_n es el conjunto de etiquetas de todos los antecesores de n en T, incluyendo al propio nodo n.

Construcción de un tablero completo

Un **tablero** para F es un árbol T, con nodos etiquetados por fórmulas y raíz etiquetada por F, construido mediante las reglas: **Para cada hoja n de T, ni abierta ni cerrada,**:

1. Si existe un par de literales complementarios en U_n, marcar con $\times$ (**hoja cerrada**).
2. Si no, pero hay una fórmula, A, en U_n **no usada**:
   - 2.1 A de tipo α: añadir n_1 hijo de n, $E_{n_1} = \alpha_1$, y otro n_2 hijo de n_1, $E_{n_2} = \alpha_2$, y **marcar A como usada** en n_2.
   - 2.2 A de tipo β: añadir n_1, n_2 a n, $E_{n_1} = \beta_1$, $E_{n_2} = \beta_2$, y marcar A como usada en n_1 y n_2.
   - 2.3 A de tipo δ: añadir m a n, $E_m = \delta_a$ (a una nueva constante), y **marcar A como usada en m**.
   - 2.4 A de tipo γ y existe t, término cerrado del lenguaje del tablero, con $\gamma_t \notin U_n$: añadir m a n, $E_m = \gamma_t$ (**no se marca A como usada**).
3. Si **no es aplicable ninguna regla** anterior en U_n, marcar con $\bigcirc$ (**hoja abierta**).

Necesidad de las restricciones de las reglas y tableros infinitos

- **Importante**: si en el lenguaje no tiene constantes, y debemos aplicar una regla γ, añadimos una.
- Necesidad de usar **constantes nuevas para la regla delta**.

  Ejemplo: $\{\exists x\, P(x),\ \exists x\, \neg P(x)\}$
- Necesidad de **reutilizar las reglas gamma**.

  Ejemplo: $\{\forall x\, \neg P(x),\ P(a) \vee P(b) \vee P(c)\}$
- Posibilidad de **ramas infinitas**.

  Ejemplo: $\forall x\, \exists y\, P(x, y)$

Propiedades de los tableros completos

- Un tablero para un conjunto de fórmulas $\{A_1, \ldots, A_n\}$ es un tablero para la fórmula $A_1 \wedge \cdots \wedge A_n$.
- Un **tablero completo** es un tablero construido siguiendo las reglas anteriores y que no puede extenderse más.
- **Recuerda** que puede ocurrir que sea **infinito** (una rama puede crecer indefinidamente).
- Un tablero es **cerrado** si todas sus hojas son cerradas, y **abierto** en caso contrario.
- Los tableros completos con ramas infinitas no puede tener todas las hojas cerradas.

Propiedades de los tableros completos

- **Teorema.** (Corrección y Completitud)
  Sea S un conjunto de fórmulas cerradas:
  1. **Corrección**: Si S admite un tablero completo y cerrado, entonces S es inconsistente.
  2. **Completitud**: Si S es inconsistente, entonces S admite un tablero completo y cerrado.
- Si S es inconsistente, entonces admite un tablero pero no nos dice cómo se construiría.
- Si no encontramos un tablero cerrado ni abierto, **no sabemos qué ocurre**.

Extrayendo modelos de un tablero completo

Si un conjunto de fórmulas cerradas S es consistente, entonces pueden darse dos situaciones:

1. S admite un tablero completo abierto finito, y por tanto tiene una rama finita abierta.
2. El tablero para S tiene una rama infinita abierta, R, donde:
   - 2.1 Para toda fórmula de tipo α que etiqueta un nodo de R existen descendientes de dicho nodo en R etiquetados con las componentes de dicha fórmula.
   - 2.2 Para cada fórmula de tipo β o δ que etiqueta un nodo de R existe un descendiente de dicho nodo en R etiquetado con una de las componentes de la fórmula.
   - 2.3 Para toda fórmula de tipo γ que etiqueta un nodo de R y cada término cerrado t del lenguaje de R, existe un nodo en R etiquetado con γ_t.

En cualquier caso, podemos definir un modelo utilizando la rama abierta correspondiente.

Extrayendo modelos de un tablero completo (II)

- Un tablero finito completo para el conjunto

$$S = \{\forall x\, \forall y\, (P(x,y) \rightarrow P(y,x)),\ \forall x\, \neg P(x,x),\ \exists x\, \exists y\, P(x,y)\}$$

proporciona el siguiente modelo:
  - Universo $M = \{0,1\}$
  - $P^{\mathcal{M}} = \{(0,1),\ (1,0)\}$
- El conjunto

$$\{\exists x\, Q(x),\ \forall x\, P(x, f(x)),\ \forall x\, \neg P(x,x)\}$$

es consistente pero no admite ningún tablero completo **finito**. Sin embargo, es posible construir un tablero infinito en el que una rama infinita proporciona el siguiente modelo:
  - Universo: $M = \{0,1,2,\dots\}$
  - $Q^{\mathcal{M}} = \{0\}$
  - Para cada $j \in M$, $f^{\mathcal{M}}(j) = j+1$
  - $P^{\mathcal{M}} = \{(j, j+1) :\ j \in M\}$.

Consecuencia lógica

Para determinar si se tiene

$$\{\forall x\,(P(x) \to Q(x)),\ \forall y\,(Q(y) \vee R(y) \to S(a))\} \models \forall x\,(P(x) \to S(a))$$

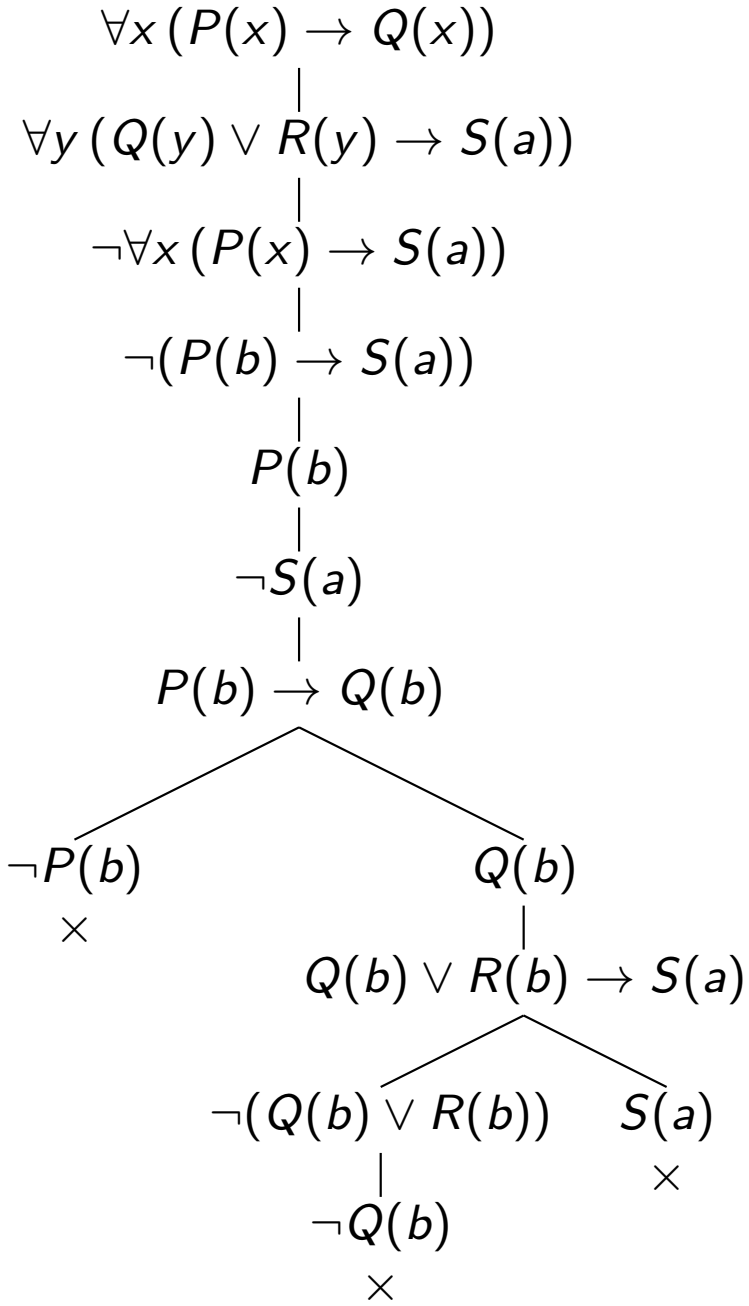
- Como $\forall x\,(P(x) \to S(a))$ es **cerrada** podemos trabajar con el conjunto

$$\Sigma = \{\forall x\,(P(x) \to Q(x)),\ \forall y\,(Q(y) \vee R(y) \to S(a)),$$

$$\neg\forall x\,(P(x) \to S(a))\}$$

- Si Σ es inconsistente la fórmula $\forall x\,(P(x) \to S(a))$ será consecuencia lógica del resto de fórmulas.
- Podemos verificarlo con un tablero:

Consecuencia lógica



Ejemplo de no consecuencia mediante tablero

$\forall x\,[P(x) \vee Q(x)] \not\models \forall x\,P(x) \vee \forall x\,Q(x)$

1 $\forall x\,[P(x) \vee Q(x)]$
2 $\neg(\forall x\,P(x) \vee \forall x\,Q(x))$
3 $\neg\forall x\,P(x)$ (2)
4 $\neg\forall x\,Q(x)$ (2)
5 $\neg P(a)$ (3)
6 $\neg Q(b)$ (4)
7 $P(a) \vee Q(a)$ (1)
8 $P(b) \vee Q(b)$ (1)

9 $P(a)$ (7)
Cerrada (9,5)

10 $Q(a)$ (7)

11 $P(b)$ (8)
Abierta

12 $Q(b)$ (8)
Cerrada (12, 6)

Contramodelo: $U = \{a, b\}, I(P) = \{b\}, I(Q) = \{a\}$.

Razonamiento con igualdad

- Si L es un LPO con igualdad, el razonamiento con fórmulas de L debe debe tener en cuenta que el predicado de igualdad necesita un tratamiento específico.
- Un posibilidad es incluir **axiomas** que describen las propiedades fundamentales del predicado de igualdad:
  - (Identidad) $\forall x\,(x = x)$.
  - (Sustitución) Si $t_1, \ldots, t_n$ son términos de L y $F(x_1, \ldots, x_n)$ es una fórmula **atómica** entonces

$$\forall x_1 \ldots \forall x_n\,(x_1 = t_1 \wedge \cdots \wedge x_n = t_n \rightarrow (F(x_1, \ldots, x_n) \rightarrow F(t_1, \ldots, t_n)))$$

- Otras propiedades de la igualdad pueden obtenerse a partir de las anteriores, por ejemplo:
  - (Simetría) $\forall x\,\forall y\,(x = y \rightarrow y = x)$.
  - (Transitividad) $\forall x\,\forall y\,(x = y \wedge y = z \rightarrow x = z)$.
- **Esto añade más complejidad**

Tableros e igualdad

El método de tableros se extiende con dos nuevas **reglas de igualdad** para incorporar el razonamiento con igualdad en la lógica de primer orden.

- **Regla 1**. Para cada término **cerrado**, t, se puede extender cualquier rama del tablero añadiendo un nuevo descendiente marcado con la fórmula

$$t = t$$

- **Regla 2**. Si t y s son términos **cerrados** y en una rama del tablero aparecen una fórmula **atómica** $P(t_1, \ldots, u[t], \ldots, t_n)$ y la fórmula $t = s$, entonces podemos extender dicha rama añadiendo un nuevo descendiente marcado con la fórmula

$$P(t_1, \ldots, u[s], \ldots, t_n)$$

Ejemplos (I)

$\forall x \forall y (x = y \to f(x) = f(y))$ es lógicamente válida:

$$\neg\forall x \forall y (x = y \to f(x) = f(y))$$
$$|$$
$$\neg\forall y (a = y \to f(a) = f(y))$$
$$|$$
$$\neg(a = b \to f(a) = f(b))$$
$$|$$
$$a = b$$
$$|$$
$$\neg f(a) = f(b)$$
$$|$$
$$\neg f(b) = f(b)$$
$$|$$
$$f(b) = f(b)$$
$$\times$$

Ejemplos (II)

$\forall x \forall y \forall z\,(x = y \wedge y = z \rightarrow x = z)$ es lógicamente válida:

$$\neg\forall x \forall y \forall z\,(x = y \wedge y = z \rightarrow x = z)$$
$$\neg\forall y \forall z\,(a = y \wedge y = z \rightarrow a = z)$$
$$\neg\forall z\,(a = b \wedge b = z \rightarrow a = z)$$
$$\neg(a = b \wedge b = c \rightarrow a = c)$$
$$a = b \wedge b = c$$
$$\neg a = c$$
$$a = b$$
$$b = c$$
$$a = c$$
$$\times$$

Ejemplos (III)

$$\{\exists x\,(f(x,x)=a),\ \forall x\,(f(x,a)=b)\} \models \exists x\,(f(x,f(x,x))=b)$$

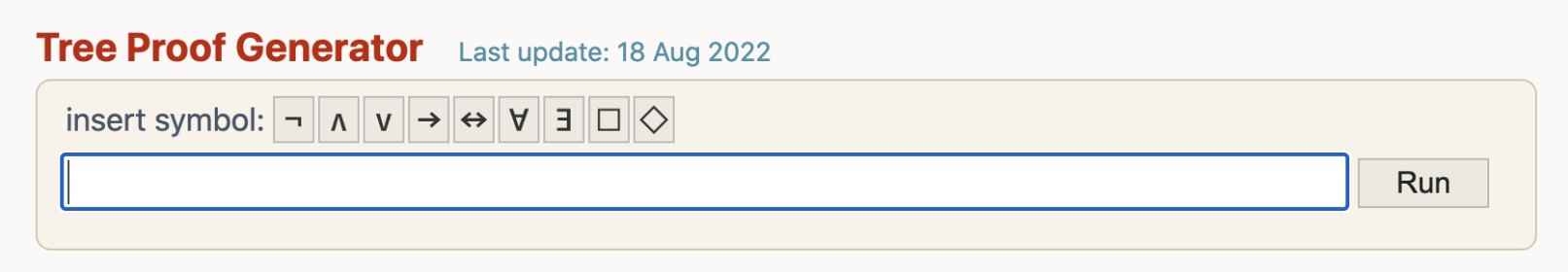
$$\exists x\,(f(x,x)=a)$$
$$|$$
$$\forall x\,(f(x,a)=b)$$
$$|$$
$$\neg\exists x\,(f(x,f(x,x))=b)$$
$$|$$
$$f(c,c)=a$$
$$|$$
$$\neg f(c,f(c,c))=b$$
$$|$$
$$\neg f(c,a)=b$$
$$|$$
$$f(c,a)=b$$
$$\times$$

Un sistema: Tree Proof Generator

- Hay más sistemas en el apartado **Sistemas** de la página de la asignatura:

`https://www.cs.us.es/~fsancho/?p=logica-informatica-2022-23`

- Usadlo para comprobar vuestras soluciones.
- `https://www.umsu.de/trees/`



- Sirve para resolver el problema de la **validez lógica**.
- Ejemplo: Para intentar demostrar si

$$\{\exists x\,(f(x,x)=a),\ \forall x\,(f(x,a)=b)\} \models \exists x\,(f(x,f(x,x))=b)$$

debemos preguntarnos si es tautología

$$\models \exists x(f(x,x)=a)\wedge\forall x(f(x,a)=b)\rightarrow\exists x(f(x,f(x,x))=b)$$

Ejecutando el ejemplo

$$\{\exists x\,(f(x,x)=a),\ \forall x\,(f(x,a)=b)\} \models \exists x\,(f(x,f(x,x))=b)$$

Tree Proof Generator Last update: 18 Aug 2022

insert symbol: ¬ ∧ ∨ → ↔ ∀ ∃ □ ◇

∃x (f (x, x) = a)∧∀x (f (x, a) = b) → ∃x (f (x, f (x, x)) = b) Run

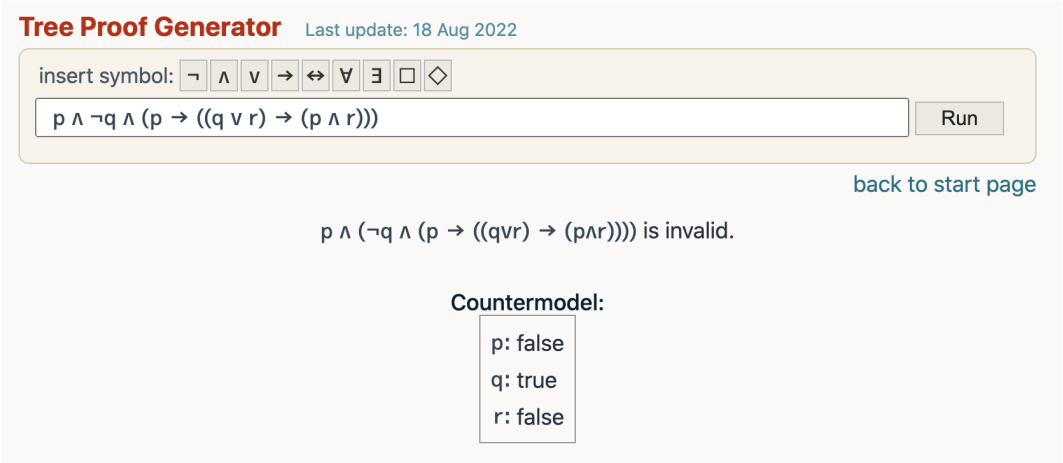
back to start page

(∃xf(x,x)=a ∧ ∀xf(x,a)=b) → ∃xf(x,f(x,x))=b is valid.

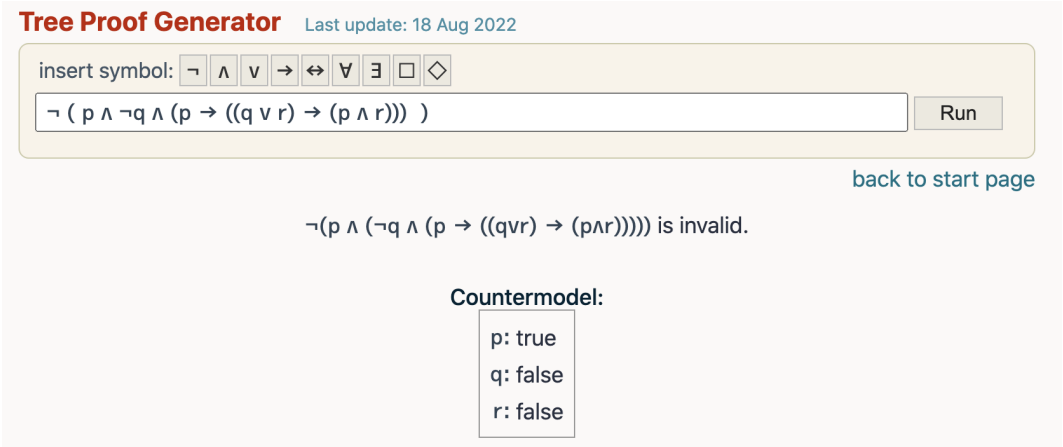
1.	¬((∃xf(x,x)=a ∧ ∀xf(x,a)=b) → ∃xf(x,f(x,x))=b)	
2.	∃xf(x,x)=a ∧ ∀xf(x,a)=b	(1)
3.	¬∃xf(x,f(x,x))=b	(1)
4.	∃xf(x,x)=a	(2)
5.	∀xf(x,a)=b	(2)
6.	f(c,c)=a	(4)
7.	f(c,a)=b	(5)
8.	¬f(c,f(c,c))=b	(3)
9.	¬f(c,a)=b	(6,8,LL)
10.	¬b=b	(7,9,LL)
	x	

Ejemplo proposicional

- $p \wedge \neg q \wedge (p \rightarrow ((q \vee r) \rightarrow (p \wedge r)))$ **no es una tautología**: Si lo fuera, su negación sería inconsistente. No lo es pues encuentra un modelo de la negación:



- Para ver si **es consistente**: se niega y preguntamos si es una tautología (no debe serlo):

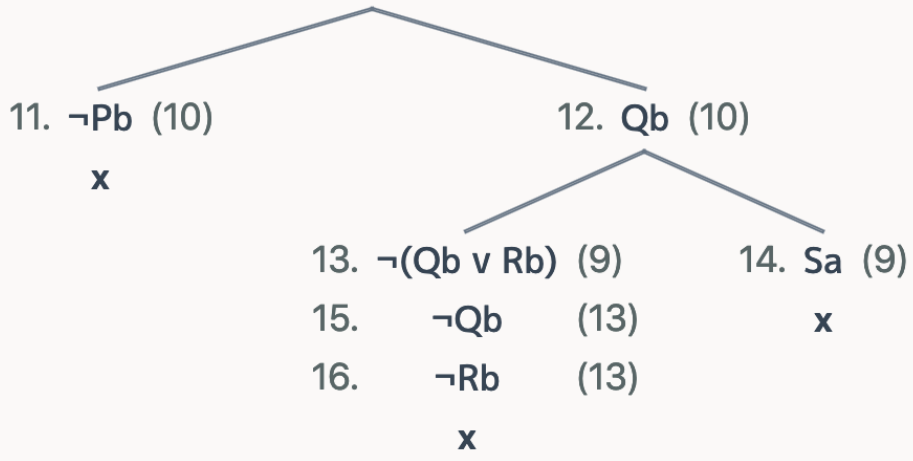


Otro ejemplo que hemos estudiado

- $\{\forall x\,(P(x) \to Q(x)),\ \forall y\,(Q(y) \vee R(y) \to S(a))\} \models \forall x\,(P(x) \to S(a))$

(∀x(Px → Qx) ∧ ∀y((Qy v Ry) → Sa)) → ∀x(Px → Sa) is valid.

1. ¬((∀x(Px → Qx) ∧ ∀y((Qy v Ry) → Sa)) → ∀x(Px → Sa))
2. ∀x(Px → Qx) ∧ ∀y((Qy v Ry) → Sa) (1)
3. ¬∀x(Px → Sa) (1)
4. ∀x(Px → Qx) (2)
5. ∀y((Qy v Ry) → Sa) (2)
6. ¬(Pb → Sa) (3)
7. Pb (6)
8. ¬Sa (6)
9. (Qb v Rb) → Sa (5)
10. Pb → Qb (4)



Bibliografía complementaria

1. M. Ben–Ari, *Mathematical logic for computer science (2nd ed.).* (Springer, 2001), Cap. 2 (Propositional calculus: formulas, models, tableaux). `https://link.springer.com/book/10.1007/978-1-4471-4129-7`
2. Trouvain, Mira Toshiko, *Propositional and First-Order Logic Tableaux: Concept and Heuristics*, `https://www21.in.tum.de/teaching/sar/SS20/2.pdf` (Technische Universität München 2020)
3. M. Huth y M. Ryan *Logic in computer science: modelling and reasoning about systems.* (Cambridge University Press, 2000) pp. 90–109 y 128–140.
4. L. Paulson *Logic and proof* (U. Cambridge, 2002)