Lógica de primer orden
Clase 4

Gustavo Landfried

Grupo Antropocaos

24 de septiembre de 2018

Lógica proposicional

- Aunque les haya parecido horrible, creanme que es **MUY** simple.

Lógica proposicional

- Aunque les haya parecido horrible, creanme que es **MUY** simple.
- No sirve para especificar (excepto problemas triviales).

Lógica proposicional

- Aunque les haya parecido horrible, creanme que es **MUY** simple.
- No sirve para especificar (excepto problemas triviales).
- Sí sirve para aprender los conectivos, cómo se escriben algunas cosas, cómo se trata formalmente la definición de un lenguaje y empezar a razonar.

Lógica proposicional

- Aunque les haya parecido horrible, creanme que es **MUY** simple.
- No sirve para especificar (excepto problemas triviales).
- Sí sirve para aprender los conectivos, cómo se escriben algunas cosas, cómo se trata formalmente la definición de un lenguaje y empezar a razonar.
- Vamos a ponerle un poquito más de sabor a la lógica proposicional...

Lógica proposicional

- Aunque les haya parecido horrible, creanme que es **MUY** simple.
- No sirve para especificar (excepto problemas triviales).
- Sí sirve para aprender los conectivos, cómo se escriben algunas cosas, cómo se trata formalmente la definición de un lenguaje y empezar a razonar.
- Vamos a ponerle un poquito más de sabor a la lógica proposicional... ¿Alguno probó la salsa Tabasco?...

Un poco de motivación

¿Qué resultado da la siguiente expresión ($x \neq 0 \rightarrow t = y/x$)?

Un poco de motivación

¿Qué resultado da la siguiente expresión ($x \neq 0 \rightarrow t = y/x$)?

- El objetivo atrás de esto es especificar problemas que puedan resolverse con un algoritmo.
- Existe la posibilidad de que un programa pueda realizar una operación inválida.

Un poco de motivación

¿Qué resultado da la siguiente expresión $(x \neq 0 \rightarrow t = y/x)$?

- El objetivo atrás de esto es especificar problemas que puedan resolverse con un algoritmo.
- Existe la posibilidad de que un programa pueda realizar una operación inválida. Por ejemplo:
  - Como ya vimos, intentar dividir por cero.
  - Hacer la raíz cuadrada de un número negativo (si nos manejamos en $\mathbb{R}$).

Un poco de motivación

¿Qué resultado da la siguiente expresión $(x \neq 0 \rightarrow t = y/x)$?

- El objetivo atrás de esto es especificar problemas que puedan resolverse con un algoritmo.
- Existe la posibilidad de que un programa pueda realizar una operación inválida. Por ejemplo:
  - Como ya vimos, intentar dividir por cero.
  - Hacer la raíz cuadrada de un número negativo (si nos manejamos en $\mathbb{R}$).
- ¿Cuál es el resultado de estas operaciones?

Un poco de motivación

¿Qué resultado da la siguiente expresión ($x \neq 0 \to t = y/x$)?

- El objetivo atrás de esto es especificar problemas que puedan resolverse con un algoritmo.
- Existe la posibilidad de que un programa pueda realizar una operación inválida. Por ejemplo:
  - Como ya vimos, intentar dividir por cero.
  - Hacer la raíz cuadrada de un número negativo (si nos manejamos en $\mathbb{R}$).
- ¿Cuál es el resultado de estas operaciones? No tienen un resultado *definido*.
- Los lenguajes de programación tienen construcciones que permiten que el programador evite o maneje este tipo de situaciones.

Un poco de motivación

¿Qué resultado da la siguiente expresión ($x \neq 0 \rightarrow t = y/x$)?

- El objetivo atrás de esto es especificar problemas que puedan resolverse con un algoritmo.
- Existe la posibilidad de que un programa pueda realizar una operación inválida. Por ejemplo:
  - Como ya vimos, intentar dividir por cero.
  - Hacer la raíz cuadrada de un número negativo (si nos manejamos en $\mathbb{R}$).
- ¿Cuál es el resultado de estas operaciones? No tienen un resultado *definido*.
- Los lenguajes de programación tienen construcciones que permiten que el programador evite o maneje este tipo de situaciones.
- Queremos poder hacerlo también en la especificación, por ejemplo, para poder describir qué rango de valores *acepta* una función.

Definición considerando `Bottom`

- Lo que vamos a usar para tratar la situación es:
  - Lógica proposicional **trivaluada** (se basa en lo que vimos).
  - Seguimos teniendo las constantes proposionales `True` y `False`.
  - Agregamos una nueva denominada `Bottom` ($\perp$).

Definición considerando `Bottom`

- Lo que vamos a usar para tratar la situación es:
  - Lógica proposicional **trivaluada** (se basa en lo que vimos).
  - Seguimos teniendo las constantes proposionales `True` y `False`.
  - Agregamos una nueva denominada `Bottom` ($\perp$).
- Los símbolos de nuestro lenguaje (se denomina a esto *alfabeto*) serán:

  0. Las constantes proposicionales: `True`, `False` y $\perp$.
  1. Los operadores básicos: $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$, y los separadores '(' y ')'.
  2. El conjunto no vacío de *símbolos de proposición* o *variables proposicionales*.

Definición considerando `Bottom`

- Lo que vamos a usar para tratar la situación es:
  - Lógica proposicional **trivaluada** (se basa en lo que vimos).
  - Seguimos teniendo las constantes proposionales `True` y `False`.
  - Agregamos una nueva denominada `Bottom` ($\perp$).
- Los símbolos de nuestro lenguaje (se denomina a esto *alfabeto*) serán:
  0. Las constantes proposicionales: `True`, `False` y $\perp$.
  1. Los operadores básicos: $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$, y los separadores '(' y ')'.
  2. El conjunto no vacío de *símbolos de proposición* o *variables proposicionales*.

Indefinición

Para poder tratar esta situación, agregamos una nueva *letra* al alfabeto, a continuación le daremos significado.

Tablas de verdad trivaluadas

Veamos cómo quedan las tablas de verdad considerando ahora a $\perp$ como un nuevo posible valor que toman las variables.

ϕ	$\neg\phi$
True	False
False	True
$\perp$	$\perp$

ϕ	ψ	$(\phi \wedge \psi)$
False	False	False
True	False	False
False	True	False
True	True	True
False	$\perp$	False
True	$\perp$	$\perp$
$\perp$	False	$\perp$
$\perp$	True	$\perp$
$\perp$	$\perp$	$\perp$

ϕ	ψ	$(\phi \vee \psi)$
False	False	False
True	False	True
False	True	True
True	True	True
False	$\perp$	$\perp$
True	$\perp$	True
$\perp$	False	$\perp$
$\perp$	True	$\perp$
$\perp$	$\perp$	$\perp$

Tablas de verdad trivaluadas

Y las que faltan nos quedan así:

ϕ	ψ	$(\phi \to \psi)$
False	False	True
True	False	False
False	True	True
True	True	True
False	$\perp$	True
True	$\perp$	$\perp$
$\perp$	False	$\perp$
$\perp$	True	$\perp$
$\perp$	$\perp$	$\perp$

ϕ	ψ	$(\phi \leftrightarrow \psi)$
False	False	True
True	False	False
False	True	False
True	True	True
False	$\perp$	$\perp$
True	$\perp$	$\perp$
$\perp$	False	$\perp$
$\perp$	True	$\perp$
$\perp$	$\perp$	$\perp$

¿Cómo queda la definición de tautología y contradicción?

Tenemos que ajustar estas definiciones ante la aparición de la nueva constante ($\perp$):

- **Tautologías:** Decimos que una fórmula ϕ es una tautología si su evaluación *da* `True` como resultado independientemente de los valores **definidos** (i.e. que no sean $\perp$) que tengan las variables proposicionales.
  Ejemplo: ($p \vee \neg p$).

¿Cómo queda la definición de tautología y contradicción?

Tenemos que ajustar estas definiciones ante la aparición de la nueva constante ($\perp$):

- **Tautologías:** Decimos que una fórmula ϕ es una tautología si su evaluación *da* `True` como resultado independientemente de los valores **definidos** (i.e. que no sean $\perp$) que tengan las variables proposicionales.
  Ejemplo: $(p \vee \neg p)$.

- **Contradicciones:** Decimos que una fórmula ϕ con valores **definidos** (i.e. que no sean $\perp$) es una contradicción si su negación $\neg\phi$ es una tautología.
  Ejemplo: $(p \wedge \neg p)$.

¿Cómo queda la definición de tautología y contradicción?

Tenemos que ajustar estas definiciones ante la aparición de la nueva constante ($\perp$):

- **Tautologías:** Decimos que una fórmula ϕ es una tautología si su evaluación *da* `True` como resultado independientemente de los valores **definidos** (i.e. que no sean $\perp$) que tengan las variables proposicionales.
  Ejemplo: $(p \vee \neg p)$.

- **Contradicciones:** Decimos que una fórmula ϕ con valores **definidos** (i.e. que no sean $\perp$) es una contradicción si su negación $\neg\phi$ es una tautología.
  Ejemplo: $(p \wedge \neg p)$.

- **Contingencias:** Decimos que una fórmula ϕ con valores **definidos** (i.e. que no sean $\perp$) es una contingencia si no es tautología ni contradicción.
  Ejemplo: $(p \vee q)$

Relación de fuerza

- Decimos (todo significan lo mismo y se usan de acuerdo a gusto del consumidor):
    - “A es más fuerte que B”
    - “A fuerza B”
    - “B es más débil que A”

Relación de fuerza

- Decimos (todo significan lo mismo y se usan de acuerdo a gusto del consumidor):
  - “A es más fuerte que B”
  - “A fuerza B”
  - “B es más débil que A”
- Si ($A \rightarrow B$) es tautología.
- Esto es algo que se debe probar formalmente... también podemos usar tablas de verdad.

Relación de fuerza

- Decimos (todo significan lo mismo y se usan de acuerdo a gusto del consumidor):
  - "A es más fuerte que B"
  - "A fuerza B"
  - "B es más débil que A"
- Si $(A \rightarrow B)$ es tautología.
- Esto es algo que se debe probar formalmente... también podemos usar tablas de verdad.
- Ejemplos
  - p es más fuerte que $(p \vee q)$
  - $(p \wedge q)$ es más fuerte que p

Importante

Vamos a hacer pequeñas demostraciones durante el curso, hay que justificar los pasos, una demostración sin justificación **no sirve**.

Tipos de datos

- Empezamos con unos tipos básicos
- Después veremos otros más complejos

Tipos de datos

- Empezamos con unos tipos básicos
- Después veremos otros más complejos
- Para hablar de un elemento de un tipo en nuestro lenguaje, escribimos un **término**:

Tipos de datos

- Empezamos con unos tipos básicos
- Después veremos otros más complejos
- Para hablar de un elemento de un tipo en nuestro lenguaje, escribimos un **término**:
  - Variable

Tipos de datos

- Empezamos con unos tipos básicos
- Después veremos otros más complejos
- Para hablar de un elemento de un tipo en nuestro lenguaje, escribimos un **término**:
  - Variable
  - Constante

Tipos de datos

- Empezamos con unos tipos básicos
- Después veremos otros más complejos
- Para hablar de un elemento de un tipo en nuestro lenguaje, escribimos un **término**:
  - Variable
  - Constante
  - Función (operación) aplicada a otros términos

Tipos de datos

- Empezamos con unos tipos básicos
- Después veremos otros más complejos
- Para hablar de un elemento de un tipo en nuestro lenguaje, escribimos un **término**:
  - Variable
  - Constante
  - Función (operación) aplicada a otros términos
- Todos los tipos tienen un elemento distinguido:
  - Indefinido
  - Puede representarse por la constante $\perp$ (que también está en todos los tipos)

$\mathbb{Z}$ - Números enteros

- Constantes:
  0; 1; –1; 2; –2; 3; –3; ...

$\mathbb{Z}$ - Números enteros

- Constantes:
  0; 1; –1; 2; –2; 3; –3; ...
- Funciones:
  - $a + b$
  - $a - b$
  - $-a$
  - $a \cdot b$
  - a/b (cociente de la división entera)
  - $a \ \mathtt{mod}\ b$ (resto de la división entera)
  - a^b

$\mathbb{R}$ - Números con coma

Lo llamaremos $\mathbb{R}$, pero estamos abusando un poco.

- Constantes:
  0; 1; 5; 2,7; 92,896; π; ...
  Algunas son compartidas con los enteros ($\mathbb{Z}$), se deduce el tipo que corresponde por el contexto de uso.

$\mathbb{R}$ - Números con coma

Lo llamaremos $\mathbb{R}$, pero estamos abusando un poco.

- Constantes:
  0; 1; 5; 2,7; 92,896; π; ...
  Algunas son compartidas con los enteros ($\mathbb{Z}$), se deduce el tipo que corresponde por el contexto de uso.
- Funciones:
  - Las mencionadas para los enteros (salvo `div` y `mod`).
  - a/b
  - $log_a(b)$
  - trigonométricas (`sin`, `cos`, etc).
  - $\lfloor a \rfloor$ (parte entera de un número con coma, redondea al entero inferior)
  - $\lceil a \rceil$ (parte entera de un número con coma, redondea al entero superior)

$\mathbb{R}$ - Números con coma

Lo llamaremos $\mathbb{R}$, pero estamos abusando un poco.

- Constantes:
  0; 1; 5; 2,7; 92,896; π; ...
  Algunas son compartidas con los enteros ($\mathbb{Z}$), se deduce el tipo que corresponde por el contexto de uso.
- Funciones:
  - Las mencionadas para los enteros (salvo `div` y `mod`).
  - a/b
  - $log_a(b)$
  - trigonométricas (`sin`, `cos`, etc).
  - $\lfloor a \rfloor$ (parte entera de un número con coma, redondea al entero inferior)
  - $\lceil a \rceil$ (parte entera de un número con coma, redondea al entero superior)
- En general, los términos de tipo entero pueden usarse como términos de este tipo.

Carácteres - $\mathbb{C}_h$

- Constantes:
  'A', 'B', ..., 'Z' , ..., 'a', 'b', ..., 'z', ..., '0', '1', ..., '9', etc.
- Función `ord`:
  - Está definida de los $\mathbb{C}_h$ a los $\mathbb{Z}$.
  - Sirve para *numerar* los caracteres. Es decir, a cada letra le asigna un valor numérico.
  - No importa mucho cuál es el valor de cada uno, pero nos ordena los caracteres de la manera que *esperamos*.

Carácteres - $\mathbb{C}_h$

- Constantes:
  'A', 'B', ..., 'Z' , ..., 'a', 'b', ..., 'z', ..., '0', '1', ..., '9', etc.
- Función `ord`:
  - Está definida de los $\mathbb{C}_h$ a los $\mathbb{Z}$.
  - Sirve para *numerar* los caracteres. Es decir, a cada letra le asigna un valor numérico.
  - No importa mucho cuál es el valor de cada uno, pero nos ordena los caracteres de la manera que *esperamos*.
  - Ejemplos:
    - ord('A') + 1 = ord('B')
    - ord('a') + 1 = ord('b')
    - ord('1') + 1 = ord ('2')

Carácteres - $\mathbb{C}_h$

- Constantes:
  'A', 'B', ..., 'Z' , ..., 'a', 'b', ..., 'z', ..., '0', '1', ..., '9', etc.
- Función `ord`:
  - Está definida de los $\mathbb{C}_h$ a los $\mathbb{Z}$.
  - Sirve para *numerar* los caracteres. Es decir, a cada letra le asigna un valor numérico.
  - No importa mucho cuál es el valor de cada uno, pero nos ordena los caracteres de la manera que *esperamos*.
  - Ejemplos:
    - ord('A') + 1 = ord('B')
    - ord('a') + 1 = ord('b')
    - ord('1') + 1 = ord ('2')
  - La inversa (ord^{-1}) sirve para obtener un caracter teniendo un número entero:
    - $\text{ord}^{-1}(\text{ord}(c)) = c$
    - $\text{ord}(\text{ord}^{-1}(n)) = n$
    - Importante: no todos los enteros tiene un caracter asociado.

$\mathbb{B}$ - Bool

- Los términos de este tipo son las fórmulas de nuestro lenguaje de la lógica proposicional.
- Constantes:
  `True`, `False`

$\mathbb{B}$ - Bool

- Los términos de este tipo son las fórmulas de nuestro lenguaje de la lógica proposicional.
- Constantes:
  `True`, `False`
- Funciones:
  - Operaciones binarias:
    - $(a_1 \vee a_2)$
    - $(a_1 \wedge a_2)$
    - $(a \rightarrow b)$
    - $(a \leftrightarrow b)$
  - $\neg a$

$\mathbb{B}$ - Bool

- Los términos de este tipo son las fórmulas de nuestro lenguaje de la lógica proposicional.
- Constantes:
  `True`, `False`
- Funciones:
  - Operaciones binarias:
    - $(a_1 \lor a_2)$
    - $(a_1 \land a_2)$
    - $(a \rightarrow b)$
    - $(a \leftrightarrow b)$
  - $\neg a$
- Hay una función especial que sirve para obtener un número a partir de un valor de verdad (útil en algunos casos):
  - $\beta : \mathbb{B} \rightarrow \mathbb{Z}$.
  - $\beta(\mathtt{False}) = 0$
  - $\beta(\mathtt{True}) = 1$
  - $\beta(\bot) = \bot$

Cuantificadores

- Se extiende la lógica proposicional con dos nuevos símbolos: $\forall$ y $\exists$.

Cuantificadores

- Se extiende la lógica proposicional con dos nuevos símbolos: $\forall$ y $\exists$.
- En ambos casos definen términos de $\mathbb{B}$.
- Es decir, su valor puede ser `True` o `False`.

Cuantificadores

- Se extiende la lógica proposicional con dos nuevos símbolos: $\forall$ y $\exists$.
- En ambos casos definen términos de $\mathbb{B}$.
- Es decir, su valor puede ser `True` o `False`.
- $\forall$: Cuantificador universal. Se lee “para todo”
- Ejemplo: $(\forall x, a, j : \mathbb{Z})(x < j + k + 1)$

Cuantificadores

- Se extiende la lógica proposicional con dos nuevos símbolos: $\forall$ y $\exists$.
- En ambos casos definen términos de $\mathbb{B}$.
- Es decir, su valor puede ser `True` o `False`.
- $\forall$: Cuantificador universal. Se lee “para todo”
- Ejemplo: $(\forall x, a, j : \mathbb{Z})(x < j + k + 1)$
- $\exists$: Cuantificador existencial. Se lee “existe”
- Ejemplo: $(\exists b : \mathbb{B}, z : \mathbb{Z}, c : \mathbb{C}_h)(\beta(b) \cdot z + ord(c) = 7)$

Cuantificadores

- Se extiende la lógica proposicional con dos nuevos símbolos: $\forall$ y $\exists$.
- En ambos casos definen términos de $\mathbb{B}$.
- Es decir, su valor puede ser `True` o `False`.
- $\forall$: Cuantificador universal. Se lee "para todo"
- Ejemplo: $(\forall x, a, j : \mathbb{Z})(x < j + k + 1)$
- $\exists$: Cuantificador existencial. Se lee "existe"
- Ejemplo: $(\exists b : \mathbb{B}, z : \mathbb{Z}, c : \mathbb{C}_h)(\beta(b) \cdot z + ord(c) = 7)$
- Se construyen en base a:
  - una lista de variables tipadas: $b : \mathbb{B}, z : \mathbb{Z}, c : \mathbb{C}_h$
    Se explicita el tipo al cual pertenecen.

Cuantificadores

- Se extiende la lógica proposicional con dos nuevos símbolos: $\forall$ y $\exists$.
- En ambos casos definen términos de $\mathbb{B}$.
- Es decir, su valor puede ser `True` o `False`.
- $\forall$: Cuantificador universal. Se lee “para todo”
- Ejemplo: $(\forall x, a, j : \mathbb{Z})(x < j + k + 1)$
- $\exists$: Cuantificador existencial. Se lee “existe”
- Ejemplo: $(\exists b : \mathbb{B}, z : \mathbb{Z}, c : \mathbb{C}_h)(\beta(b) \cdot z + ord(c) = 7)$
- Se construyen en base a:
    - una lista de variables tipadas: $b : \mathbb{B}, z : \mathbb{Z}, c : \mathbb{C}_h$
      Se explicita el tipo al cual pertenecen.
    - un término de tipo $\mathbb{B}$: $\beta(b) \cdot z + ord(c) = 7$
      Tiene que ser `True` o `False`.

Semántica - Cuantificador universal

Veamos qué significado tienen los nuevos símbolos que agregamos.
Empecemos por el cuantificador universal:

- $B \equiv (\forall x_1, \ldots, x_m : T_1, x_{m+1}, \ldots, x_n : T_2, \ldots, x_j, \ldots, x_r : T_k)\,(A(x_1, x_2, \ldots, x_p))$

Semántica - Cuantificador universal

Veamos qué significado tienen los nuevos símbolos que agregamos. Empecemos por el cuantificador universal:

- $B \equiv (\forall x_1, \ldots, x_m : T_1, x_{m+1}, \ldots, x_n : T_2, \ldots, x_j, \ldots, x_r : T_k)\,(A(x_1, x_2, \ldots, x_p))$
- A es una fórmula.

Semántica - Cuantificador universal

Veamos qué significado tienen los nuevos símbolos que agregamos. Empecemos por el cuantificador universal:

- $B \equiv (\forall x_1, \ldots, x_m : T_1, x_{m+1}, \ldots, x_n : T_2, \ldots, x_j, \ldots, x_r : T_k)\,(A(x_1, x_2, \ldots, x_p))$
- A es una fórmula.
- Si **TODAS** las combinaciones de valores hacen que A sea `True`, entonces B es `True`.

Semántica - Cuantificador universal

Veamos qué significado tienen los nuevos símbolos que agregamos. Empecemos por el cuantificador universal:

- $B \equiv (\forall x_1, \ldots, x_m : T_1, x_{m+1}, \ldots, x_n : T_2, \ldots, x_j, \ldots, x_r : T_k)\,(A(x_1, x_2, \ldots, x_p))$
- A es una fórmula.
- Si **TODAS** las combinaciones de valores hacen que A sea `True`, entonces B es `True`.
- Si hay alguna combinación de valores **definidos** de los tipos T_1 a T_k que indefina A, entonces B es indefinida($\perp$).

Semántica - Cuantificador universal

Veamos qué significado tienen los nuevos símbolos que agregamos. Empecemos por el cuantificador universal:

- $B \equiv (\forall x_1, \ldots, x_m : T_1, x_{m+1}, \ldots, x_n : T_2, \ldots, x_j, \ldots, x_r : T_k)\,(A(x_1, x_2, \ldots, x_p))$
- A es una fórmula.
- Si **TODAS** las combinaciones de valores hacen que A sea `True`, entonces B es `True`.
- Si hay alguna combinación de valores **definidos** de los tipos T_1 a T_k que indefina A, entonces B es indefinida($\perp$).
- Si ninguna la indefine, pero hay alguna que da `False`, entonces B es `False`.

Semántica - Cuantificador existencial

Veamos ahora cómo entender el existencial:

- $B \equiv (\exists x_1, \ldots, x_m : T_1, x_{m+1}, \ldots, x_n : T_2, \ldots, x_j, \ldots, x_r : T_k)\,(A(x_1, x_2, \ldots, x_p))$

Semántica - Cuantificador existencial

Veamos ahora cómo entender el existencial:

- $B \equiv (\exists x_1, \ldots, x_m : T_1, x_{m+1}, \ldots, x_n : T_2, \ldots, x_j, \ldots, x_r : T_k)\,(A(x_1, x_2, \ldots, x_p))$
- Tiene el mismo valor que $\neg(\forall x_1, \ldots, x_m : T_1, x_{m+1}, \ldots, x_n : T_2, \ldots, x_j, \ldots, x_r : T_k)\,(\neg A(x_1, x_2, \ldots, x_p))$

Semántica - Cuantificador existencial

Veamos ahora cómo entender el existencial:

- $B \equiv (\exists x_1, \ldots, x_m : T_1, x_{m+1}, \ldots, x_n : T_2, \ldots, x_j, \ldots, x_r : T_k)\,(A(x_1, x_2, \ldots, x_p))$
- Tiene el mismo valor que $\neg(\forall x_1, \ldots, x_m : T_1, x_{m+1}, \ldots, x_n : T_2, \ldots, x_j, \ldots, x_r : T_k)\,(\neg A(x_1, x_2, \ldots, x_p))$
- Si hay alguna combinación de valores **definidos** de los tipos T_1 a T_k que indefina A, entonces B es indefinida($\perp$)
- Si **ALGUNA** de las combinaciones de valores hacen que A sea `True`, entonces B es `True`
- Si ninguna la indefine, pero tampoco ninguna la hace `True`, entonces B es `False`.

Variables libres y ligadas

- Una variable puede aparecer varias veces en una fórmula: $x + 3 = 8 \cdot x$
- Cada una se llama una *aparición*

Variables libres y ligadas

- Una variable puede aparecer varias veces en una fórmula: $x + 3 = 8 \cdot x$
- Cada una se llama una *aparición*
- Una aparición de x puede ser **ligada**: aparece bajo el *alcance* de un cuantificador universal ($\forall$) o existencial ($\exists$).
- En el otro caso, se dice que es una aparición **libre**.

Variables libres y ligadas

- Una variable puede aparecer varias veces en una fórmula: $x + 3 = 8 \cdot x$
- Cada una se llama una *aparición*
- Una aparición de x puede ser **ligada**: aparece bajo el *alcance* de un cuantificador universal ($\forall$) o existencial ($\exists$).
- En el otro caso, se dice que es una aparición **libre**.
- Ejemplos:
  - $(\forall y : \mathbb{Z})((\exists z : \mathbb{C}_h)(ord(z) = y + x))$

Variables libres y ligadas

- Una variable puede aparecer varias veces en una fórmula: $x + 3 = 8 \cdot x$
- Cada una se llama una *aparición*
- Una aparición de x puede ser **ligada**: aparece bajo el *alcance* de un cuantificador universal ($\forall$) o existencial ($\exists$).
- En el otro caso, se dice que es una aparición **libre**.
- Ejemplos:
    - $(\forall y : \mathbb{Z})((\exists z : \mathbb{C}_h)(ord(z) = y + x))$
      ligadas: y, z; libres: x (son las únicas apariciones)

Variables libres y ligadas

- Una variable puede aparecer varias veces en una fórmula: $x + 3 = 8 \cdot x$
- Cada una se llama una *aparición*
- Una aparición de x puede ser **ligada**: aparece bajo el *alcance* de un cuantificador universal ($\forall$) o existencial ($\exists$).
- En el otro caso, se dice que es una aparición **libre**.
- Ejemplos:
  - $(\forall y : \mathbb{Z})((\exists z : \mathbb{C}_h)(ord(z) = y + x))$
    ligadas: y, z; libres: x (son las únicas apariciones)
  - $(\forall x : \mathbb{Z})((\exists z : \mathbb{C}_h)(ord(z) = y + x))$

Variables libres y ligadas

- Una variable puede aparecer varias veces en una fórmula: $x + 3 = 8 \cdot x$
- Cada una se llama una *aparición*
- Una aparición de x puede ser **ligada**: aparece bajo el *alcance* de un cuantificador universal ($\forall$) o existencial ($\exists$).
- En el otro caso, se dice que es una aparición **libre**.
- Ejemplos:
  - $(\forall y : \mathbb{Z})((\exists z : \mathbb{C}_h)(ord(z) = y + x))$
    ligadas: y, z; libres: x (son las únicas apariciones)
  - $(\forall x : \mathbb{Z})((\exists z : \mathbb{C}_h)(ord(z) = y + x))$
    ligadas: x, z; libres: y (son las únicas apariciones)

Variables libres y ligadas

- Una variable puede aparecer varias veces en una fórmula: $x + 3 = 8 \cdot x$
- Cada una se llama una *aparición*
- Una aparición de x puede ser **ligada**: aparece bajo el *alcance* de un cuantificador universal ($\forall$) o existencial ($\exists$).
- En el otro caso, se dice que es una aparición **libre**.
- Ejemplos:
    - $(\forall y : \mathbb{Z})((\exists z : \mathbb{C}_h)(ord(z) = y + x))$
      ligadas: y, z; libres: x (son las únicas apariciones)
    - $(\forall x : \mathbb{Z})((\exists z : \mathbb{C}_h)(ord(z) = y + x))$
      ligadas: x, z; libres: y (son las únicas apariciones)
    - $(x > 0 \wedge (\forall x : \mathbb{Z})(1 + x > x))$

Variables libres y ligadas

- Una variable puede aparecer varias veces en una fórmula: $x + 3 = 8 \cdot x$
- Cada una se llama una *aparición*
- Una aparición de x puede ser **ligada**: aparece bajo el *alcance* de un cuantificador universal ($\forall$) o existencial ($\exists$).
- En el otro caso, se dice que es una aparición **libre**.
- Ejemplos:
  - $(\forall y : \mathbb{Z})((\exists z : \mathbb{C}_h)(ord(z) = y + x))$
    ligadas: y, z; libres: x (son las únicas apariciones)
  - $(\forall x : \mathbb{Z})((\exists z : \mathbb{C}_h)(ord(z) = y + x))$
    ligadas: x, z; libres: y (son las únicas apariciones)
  - $(x > 0 \wedge (\forall x : \mathbb{Z})(1 + x > x))$
    ligadas: x (segunda y tercera aparición); libres: x (primera aparición)

Variables libres y ligadas

- Una variable puede aparecer varias veces en una fórmula: $x + 3 = 8 \cdot x$
- Cada una se llama una *aparición*
- Una aparición de x puede ser **ligada**: aparece bajo el *alcance* de un cuantificador universal ($\forall$) o existencial ($\exists$).
- En el otro caso, se dice que es una aparición **libre**.
- Ejemplos:
  - $(\forall y : \mathbb{Z})((\exists z : \mathbb{C}_h)(ord(z) = y + x))$
    ligadas: y, z; libres: x (son las únicas apariciones)
  - $(\forall x : \mathbb{Z})((\exists z : \mathbb{C}_h)(ord(z) = y + x))$
    ligadas: x, z; libres: y (son las únicas apariciones)
  - $(x > 0 \wedge (\forall x : \mathbb{Z})(1 + x > x))$
    ligadas: x (segunda y tercera aparición); libres: x (primera aparición)
- Las fórmulas pueden ser:
  - **cerradas (o sentencias)**: no tiene apariciones (o variables) libres.
  - **abiertas**: el resto.

Ejemplos

Demos el valor de verdad de las siguientes fórmulas

0. $(\forall x : \mathbb{Z})(x + x = 2 \cdot x)$

Ejemplos

Demos el valor de verdad de las siguientes fórmulas

0. $(\forall x : \mathbb{Z})(x + x = 2 \cdot x)$
   `True`
1. $(\exists x : \mathbb{Z})(x > 0)$

Ejemplos

Demos el valor de verdad de las siguientes fórmulas

0. $(\forall x : \mathbb{Z})(x + x = 2 \cdot x)$
   `True`
1. $(\exists x : \mathbb{Z})(x > 0)$
   `True` ¿Por qué?

Ejemplos

Demos el valor de verdad de las siguientes fórmulas

0. $(\forall x : \mathbb{Z})(x + x = 2 \cdot x)$
   `True`
1. $(\exists x : \mathbb{Z})(x > 0)$
   `True` ¿Por qué? porque $(\forall x : \mathbb{Z})(x \leq 0)$ es `False`.
2. $(\forall x : \mathbb{Z})((\exists y : \mathbb{Z})(x < y))$

Ejemplos

Demos el valor de verdad de las siguientes fórmulas

0. $(\forall x : \mathbb{Z})(x + x = 2 \cdot x)$
   `True`
1. $(\exists x : \mathbb{Z})(x > 0)$
   `True` ¿Por qué? porque $(\forall x : \mathbb{Z})(x \leq 0)$ es `False`.
2. $(\forall x : \mathbb{Z})((\exists y : \mathbb{Z})(x < y))$
   `True` ¿Por qué?

Ejemplos

Demos el valor de verdad de las siguientes fórmulas

0. $(\forall x : \mathbb{Z})(x + x = 2 \cdot x)$
   `True`
1. $(\exists x : \mathbb{Z})(x > 0)$
   `True` ¿Por qué? porque $(\forall x : \mathbb{Z})(x \leq 0)$ es `False`.
2. $(\forall x : \mathbb{Z})((\exists y : \mathbb{Z})(x < y))$
   `True` ¿Por qué? porque para cualquier entero a, vale la fórmula $(\exists y : \mathbb{Z})(a < y)$
3. $(\forall x : \mathbb{Z})(x \neq 0 \rightarrow x^2 \neq x)$

Ejemplos

Demos el valor de verdad de las siguientes fórmulas

0. $(\forall x : \mathbb{Z})(x + x = 2 \cdot x)$
   `True`
1. $(\exists x : \mathbb{Z})(x > 0)$
   `True` ¿Por qué? porque $(\forall x : \mathbb{Z})(x \leq 0)$ es `False`.
2. $(\forall x : \mathbb{Z})((\exists y : \mathbb{Z})(x < y))$
   `True` ¿Por qué? porque para cualquier entero a, vale la fórmula $(\exists y : \mathbb{Z})(a < y)$
3. $(\forall x : \mathbb{Z})(x \neq 0 \rightarrow x^2 \neq x)$
   `False` ¿Por qué?

Ejemplos

Demos el valor de verdad de las siguientes fórmulas

0. $(\forall x : \mathbb{Z})(x + x = 2 \cdot x)$
   `True`
1. $(\exists x : \mathbb{Z})(x > 0)$
   `True` ¿Por qué? porque $(\forall x : \mathbb{Z})(x \leq 0)$ es `False`.
2. $(\forall x : \mathbb{Z})((\exists y : \mathbb{Z})(x < y))$
   `True` ¿Por qué? porque para cualquier entero a, vale la fórmula $(\exists y : \mathbb{Z})(a < y)$
3. $(\forall x : \mathbb{Z})(x \neq 0 \rightarrow x^2 \neq x)$
   `False` ¿Por qué? porque $(1 \neq 0 \rightarrow 1^2 \neq 1)$ es `False`

Demos el valor de verdad de las siguientes fórmulas

0. $(\forall x : \mathbb{Z})(1/x \leq 0)$

Demos el valor de verdad de las siguientes fórmulas

0. $(\forall x : \mathbb{Z})(1/x \leq 0)$
   $\perp$ ¿Por qué?

Demos el valor de verdad de las siguientes fórmulas

0. $(\forall x : \mathbb{Z})(1/x \leq 0)$
   $\perp$ ¿Por qué? porque hay una combinación de valores para x ($x = 0$) que vuelve indefinida $(1/x \leq 0)$
1. $(\exists x : \mathbb{Z})(1/x > 0)$

Demos el valor de verdad de las siguientes fórmulas

0. $(\forall x : \mathbb{Z})(1/x \leq 0)$
   $\perp$ ¿Por qué? porque hay una combinación de valores para x ($x = 0$) que vuelve indefinida $(1/x \leq 0)$
1. $(\exists x : \mathbb{Z})(1/x > 0)$
   $\perp$ ¿Por qué?

Demos el valor de verdad de las siguientes fórmulas

0. $(\forall x : \mathbb{Z})(1/x \leq 0)$
   $\perp$ ¿Por qué? porque hay una combinación de valores para x ($x = 0$) que vuelve indefinida $(1/x \leq 0)$
1. $(\exists x : \mathbb{Z})(1/x > 0)$
   $\perp$ ¿Por qué? porque hay una combinación de valores para x ($x = 0$) que vuelve indefinida $(1/x \leq 0)$
2. $x + 1 = y$ (con x, y de tipo entero).

Demos el valor de verdad de las siguientes fórmulas

0. $(\forall x : \mathbb{Z})(1/x \leq 0)$
   $\perp$ ¿Por qué? porque hay una combinación de valores para x ($x = 0$) que vuelve indefinida $(1/x \leq 0)$
1. $(\exists x : \mathbb{Z})(1/x > 0)$
   $\perp$ ¿Por qué? porque hay una combinación de valores para x ($x = 0$) que vuelve indefinida $(1/x \leq 0)$
2. $x + 1 = y$ (con x, y de tipo entero). Depende:
   - $x = 3$; $y = 4$:

Demos el valor de verdad de las siguientes fórmulas

0. $(\forall x : \mathbb{Z})(1/x \leq 0)$
   $\perp$ ¿Por qué? porque hay una combinación de valores para x ($x = 0$) que vuelve indefinida $(1/x \leq 0)$
1. $(\exists x : \mathbb{Z})(1/x > 0)$
   $\perp$ ¿Por qué? porque hay una combinación de valores para x ($x = 0$) que vuelve indefinida $(1/x \leq 0)$
2. $x + 1 = y$ (con x, y de tipo entero). Depende:
   - $x = 3$; $y = 4$: `True`
   - $x = 3$; $y = 5$:

Demos el valor de verdad de las siguientes fórmulas

0. $(\forall x : \mathbb{Z})(1/x \leq 0)$
   $\perp$ ¿Por qué? porque hay una combinación de valores para x ($x = 0$) que vuelve indefinida $(1/x \leq 0)$
1. $(\exists x : \mathbb{Z})(1/x > 0)$
   $\perp$ ¿Por qué? porque hay una combinación de valores para x ($x = 0$) que vuelve indefinida $(1/x \leq 0)$
2. $x + 1 = y$ (con x, y de tipo entero). Depende:
   - $x = 3$; $y = 4$: `True`
   - $x = 3$; $y = 5$: `False`
   - x es $\perp$; $y = 5$:

Demos el valor de verdad de las siguientes fórmulas

0. $(\forall x : \mathbb{Z})(1/x \leq 0)$
   $\perp$ ¿Por qué? porque hay una combinación de valores para x ($x = 0$) que vuelve indefinida $(1/x \leq 0)$
1. $(\exists x : \mathbb{Z})(1/x > 0)$
   $\perp$ ¿Por qué? porque hay una combinación de valores para x ($x = 0$) que vuelve indefinida $(1/x \leq 0)$
2. $x + 1 = y$ (con x, y de tipo entero). Depende:
   - $x = 3$; $y = 4$: `True`
   - $x = 3$; $y = 5$: `False`
   - x es $\perp$; $y = 5$: $\perp$

Demos el valor de verdad de las siguientes fórmulas

0. $(\forall x : \mathbb{Z})(1/x \leq 0)$
   $\perp$ ¿Por qué? porque hay una combinación de valores para x ($x = 0$) que vuelve indefinida $(1/x \leq 0)$
1. $(\exists x : \mathbb{Z})(1/x > 0)$
   $\perp$ ¿Por qué? porque hay una combinación de valores para x ($x = 0$) que vuelve indefinida $(1/x \leq 0)$
2. $x + 1 = y$ (con x, y de tipo entero). Depende:
   - $x = 3$; $y = 4$: `True`
   - $x = 3$; $y = 5$: `False`
   - x es $\perp$; $y = 5$: $\perp$
3. $(\forall x : \mathbb{Z})(x \cdot y = x)$ (con y de tipo entero).

Demos el valor de verdad de las siguientes fórmulas

0. $(\forall x : \mathbb{Z})(1/x \leq 0)$
   $\perp$ ¿Por qué? porque hay una combinación de valores para x ($x = 0$) que vuelve indefinida ($1/x \leq 0$)
1. $(\exists x : \mathbb{Z})(1/x > 0)$
   $\perp$ ¿Por qué? porque hay una combinación de valores para x ($x = 0$) que vuelve indefinida ($1/x \leq 0$)
2. $x + 1 = y$ (con x, y de tipo entero). Depende:
   - $x = 3$; $y = 4$: `True`
   - $x = 3$; $y = 5$: `False`
   - x es $\perp$; $y = 5$: $\perp$
3. $(\forall x : \mathbb{Z})(x \cdot y = x)$ (con y de tipo entero). Depende:
   - y es $\perp$:

Demos el valor de verdad de las siguientes fórmulas

0. $(\forall x : \mathbb{Z})(1/x \leq 0)$
   $\perp$ ¿Por qué? porque hay una combinación de valores para x ($x = 0$) que vuelve indefinida $(1/x \leq 0)$
1. $(\exists x : \mathbb{Z})(1/x > 0)$
   $\perp$ ¿Por qué? porque hay una combinación de valores para x ($x = 0$) que vuelve indefinida $(1/x \leq 0)$
2. $x + 1 = y$ (con x, y de tipo entero). Depende:
   - $x = 3$; $y = 4$: `True`
   - $x = 3$; $y = 5$: `False`
   - x es $\perp$; $y = 5$: $\perp$
3. $(\forall x : \mathbb{Z})(x \cdot y = x)$ (con y de tipo entero). Depende:
   - y es $\perp$: $\perp$
   - $y = 1$:

Demos el valor de verdad de las siguientes fórmulas

0. $(\forall x : \mathbb{Z})(1/x \leq 0)$
   $\perp$ ¿Por qué? porque hay una combinación de valores para x ($x = 0$) que vuelve indefinida $(1/x \leq 0)$
1. $(\exists x : \mathbb{Z})(1/x > 0)$
   $\perp$ ¿Por qué? porque hay una combinación de valores para x ($x = 0$) que vuelve indefinida $(1/x \leq 0)$
2. $x + 1 = y$ (con x, y de tipo entero). Depende:
   - $x = 3$; $y = 4$: `True`
   - $x = 3$; $y = 5$: `False`
   - x es $\perp$; $y = 5$: $\perp$
3. $(\forall x : \mathbb{Z})(x \cdot y = x)$ (con y de tipo entero). Depende:
   - y es $\perp$: $\perp$
   - $y = 1$: `True`
   - Si no,

Demos el valor de verdad de las siguientes fórmulas

0. $(\forall x : \mathbb{Z})(1/x \leq 0)$
   $\perp$ ¿Por qué? porque hay una combinación de valores para x ($x = 0$) que vuelve indefinida $(1/x \leq 0)$
1. $(\exists x : \mathbb{Z})(1/x > 0)$
   $\perp$ ¿Por qué? porque hay una combinación de valores para x ($x = 0$) que vuelve indefinida $(1/x \leq 0)$
2. $x + 1 = y$ (con x, y de tipo entero). Depende:
   - $x = 3$; $y = 4$: `True`
   - $x = 3$; $y = 5$: `False`
   - x es $\perp$; $y = 5$: $\perp$
3. $(\forall x : \mathbb{Z})(x \cdot y = x)$ (con y de tipo entero). Depende:
   - y es $\perp$: $\perp$
   - $y = 1$: `True`
   - Si no, `False`

Importante

El valor de verdad de las fórmulas abiertas (las que tienen apariciones libres) depende de los valores de sus variables libres. Las sentencias tienen un único valor de verdad al no tener dependencias *externas*.