Lógica proposicional
Clase 3

Gustavo Landfried

Grupo Antropocaos

24 de septiembre de 2018

Hay dos partes que vamos a atacar:

0. **Especificación**. Vamos a describir los problemas a resolver en un lenguaje formal:
    - Preciso
    - Claro
    - Abstracto

Hay dos partes que vamos a atacar:

0. **Especificación**. Vamos a describir los problemas a resolver en un lenguaje formal:
    - Preciso
    - Claro
    - Abstracto
1. **Implementación**. Vamos a escribir programas que *funcionen* utilizando un lenguaje imperativo (`Python` y luego `C`):
    - Es uno de los paradigmas más utilizados (hay otros).
    - Tienen una buena eficiencia en la ejecución de los programas.
    - La mayor parte de las herramientas que se utilizan están desarrolladas en algún lenguaje imperativo.

- ¿Cómo se encara?
  - Dar las propiedades de una solución
  - Dijkstra, Hoare (años 70)
- Algoritmo: descripción de la solución *fuera* de la computadora
- Programa: es la implementación *dentro* de la computadora

Se describe los aspectos externos al problema a resolver, pero haciendo foco en una solución algorítmica:

- Datos de entrada que recibe.
- Detalles o características de los datos de entrada.
- El resultado que se devuelve en base a estos parámetros. Es decir, las propiedades que tienen los resultados.
- Como lenguaje, vamos a usar un *dialecto* de la lógica de primer orden para describir **formalmente** lo que se espera que el programa haga.

Ejemplos de ambigüedad

La ambigüedad está relacionada con la interpretación múltiple:

- Se venden zapatos de piel de señora.

Ejemplos de ambigüedad

La ambigüedad está relacionada con la interpretación múltiple:

- Se venden zapatos de piel de señora.
- El chancho está listo para el desayuno.

Ejemplos de ambigüedad

La ambigüedad está relacionada con la interpretación múltiple:

- Se venden zapatos de piel de señora.
- El chancho está listo para el desayuno.
- Si el usuario presiona el botón rojo, debe activarse la alarma sonora.

Ejemplos de ambigüedad

La ambigüedad está relacionada con la interpretación múltiple:

- Se venden zapatos de piel de señora.
- El chancho está listo para el desayuno.
- Si el usuario presiona el botón rojo, debe activarse la alarma sonora.
- Sumar dos números, multiplicar por -1. Poner negativo el componente de signo si da positivo la operación.

Lógica proposicional

El lenguaje de la lógica proposicional (LP) se construye a partir del siguiente conjunto de símbolos primitivos que constituyen su alfabeto básico (usualmente notado como Σ):

0. Un conjunto no vacío[1] de *símbolos de proposición* (también llamadas variables proposicionales), que notaremos como p, q, r, ... o con subíndices p_1, p_2, p_3, ...

[1] posiblemente infinito aunque numerable

Lógica proposicional

El lenguaje de la lógica proposicional (LP) se construye a partir del siguiente conjunto de símbolos primitivos que constituyen su alfabeto básico (usualmente notado como Σ):

0. Un conjunto no vacío[1] de *símbolos de proposición* (también llamadas variables proposicionales), que notaremos como p, q, r, ... o con subíndices p_1, p_2, p_3, ...
1. Los *símbolos* constantes `True` y `False` (también llamadas constantes proposicionales), los operadores básicos $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$, y los separadores '(' y ')'.

[1] posiblemente infinito aunque numerable

Fórmulas bien formadas

- Las *oraciones* de este lenguaje se forman combinando los símbolos que acabamos de mencionar.

Fórmulas bien formadas

- Las *oraciones* de este lenguaje se forman combinando los símbolos que acabamos de mencionar.
- Sin embargo, solo nos interesan aquellas que cumplen con ser *lindas* o *interesantes*, lo que en nuestro contexto se define como una *fórmula bien formada* (fbf) de la Lógica Proposicional.

Fórmulas bien formadas

- Las *oraciones* de este lenguaje se forman combinando los símbolos que acabamos de mencionar.
- Sin embargo, solo nos interesan aquellas que cumplen con ser *lindas* o *interesantes*, lo que en nuestro contexto se define como una *fórmula bien formada* (fbf) de la Lógica Proposicional.
- Hay una serie de reglas básicas para saber si una fórmula es una fbf o no.

Fórmulas bien formadas

- Las *oraciones* de este lenguaje se forman combinando los símbolos que acabamos de mencionar.
- Sin embargo, solo nos interesan aquellas que cumplen con ser *lindas* o *interesantes*, lo que en nuestro contexto se define como una *fórmula bien formada* (fbf) de la Lógica Proposicional.
- Hay una serie de reglas básicas para saber si una fórmula es una fbf o no.
- Vamos a dar las reglas de creación de una fbf, lo cual es una manera de definir **todas** las fbf's.

Fórmulas bien formadas

- Las *oraciones* de este lenguaje se forman combinando los símbolos que acabamos de mencionar.
- Sin embargo, solo nos interesan aquellas que cumplen con ser *lindas* o *interesantes*, lo que en nuestro contexto se define como una *fórmula bien formada* (fbf) de la Lógica Proposicional.
- Hay una serie de reglas básicas para saber si una fórmula es una fbf o no.
- Vamos a dar las reglas de creación de una fbf, lo cual es una manera de definir **todas** las fbf's.
    - **BF1**: `True`, `False` son fbf's.
    - **BF2**: Toda variable proposicional es una fbf.
    - **BF3**: Si ϕ es una fbf, entonces $\neg\phi$ es una fbf.
    - **BF4**: Si ϕ y ψ son fbf, entonces $(\phi \wedge \psi)$, $(\phi \vee \psi)$, $(\phi \rightarrow \psi)$ y $(\phi \leftrightarrow \psi)$ son fbf's.

Fórmulas bien formadas

- Las *oraciones* de este lenguaje se forman combinando los símbolos que acabamos de mencionar.
- Sin embargo, solo nos interesan aquellas que cumplen con ser *lindas* o *interesantes*, lo que en nuestro contexto se define como una *fórmula bien formada* (fbf) de la Lógica Proposicional.
- Hay una serie de reglas básicas para saber si una fórmula es una fbf o no.
- Vamos a dar las reglas de creación de una fbf, lo cual es una manera de definir **todas** las fbf's.
  - **BF1**: `True`, `False` son fbf's.
  - **BF2**: Toda variable proposicional es una fbf.
  - **BF3**: Si ϕ es una fbf, entonces $\neg\phi$ es una fbf.
  - **BF4**: Si ϕ y ψ son fbf, entonces $(\phi \wedge \psi)$, $(\phi \vee \psi)$, $(\phi \to \psi)$ y $(\phi \leftrightarrow \psi)$ son fbf's.

Atención

¿Qué tiene de particular la definición de fórmula bien formada?

Fórmulas bien formadas

- Las *oraciones* de este lenguaje se forman combinando los símbolos que acabamos de mencionar.
- Sin embargo, solo nos interesan aquellas que cumplen con ser *lindas* o *interesantes*, lo que en nuestro contexto se define como una *fórmula bien formada* (fbf) de la Lógica Proposicional.
- Hay una serie de reglas básicas para saber si una fórmula es una fbf o no.
- Vamos a dar las reglas de creación de una fbf, lo cual es una manera de definir **todas** las fbf's.
  - **BF1**: `True`, `False` son fbf's.
  - **BF2**: Toda variable proposicional es una fbf.
  - **BF3**: Si ϕ es una fbf, entonces $\neg\phi$ es una fbf.
  - **BF4**: Si ϕ y ψ son fbf, entonces $(\phi \wedge \psi)$, $(\phi \vee \psi)$, $(\phi \rightarrow \psi)$ y $(\phi \leftrightarrow \psi)$ son fbf's.

Atención

¿Qué tiene de particular la definición de fórmula bien formada?
¡Está definida en función de sí misma!

Fórmulas bien formadas

- Ejemplos de fórmulas bien formadas:
  - `False`: BF1

Fórmulas bien formadas

- Ejemplos de fórmulas bien formadas:
  - `False`: BF1
  - *p*: BF2

Fórmulas bien formadas

- Ejemplos de fórmulas bien formadas:
  - `False`: BF1
  - p: BF2
  - q: BF2

Fórmulas bien formadas

- Ejemplos de fórmulas bien formadas:
  - `False`: BF1
  - p: BF2
  - q: BF2
  - $\neg p$: BF3, dado que p es una fbf por BF2.

Fórmulas bien formadas

- Ejemplos de fórmulas bien formadas:
  - `False`: BF1
  - p: BF2
  - q: BF2
  - $\neg p$: BF3, dado que p es una fbf por BF2.
  - $(True \vee \neg p)$: BF4, sabiendo que `True` y $\neg p$ son fbf.

Fórmulas bien formadas

- Ejemplos de fórmulas bien formadas:
  - `False`: BF1
  - p: BF2
  - q: BF2
  - $\neg p$: BF3, dado que p es una fbf por BF2.
  - $(True \vee \neg p)$: BF4, sabiendo que `True` y $\neg p$ son fbf.
  - $((p \vee ((q \rightarrow r) \wedge (p \rightarrow q))) \leftrightarrow r)$:

Fórmulas bien formadas

- Ejemplos de fórmulas bien formadas:
  - `False`: BF1
  - p: BF2
  - q: BF2
  - $\neg p$: BF3, dado que p es una fbf por BF2.
  - $(True \vee \neg p)$: BF4, sabiendo que `True` y $\neg p$ son fbf.
  - $((p \vee ((q \rightarrow r) \wedge (p \rightarrow q))) \leftrightarrow r)$: bueno, se los dejo para que lo piensen...
- Estos son ejemplos de fórmulas que **no** son bien formadas. Veamos por qué:
  - $p \vee (q \rightarrow r)$
  - $\vee \neg q$

Fórmulas bien formadas

- Ejemplos de fórmulas bien formadas:
  - `False`: BF1
  - p: BF2
  - q: BF2
  - $\neg p$: BF3, dado que p es una fbf por BF2.
  - $(True \vee \neg p)$: BF4, sabiendo que `True` y $\neg p$ son fbf.
  - $((p \vee ((q \rightarrow r) \wedge (p \rightarrow q))) \leftrightarrow r)$: bueno, se los dejo para que lo piensen...
- Estos son ejemplos de fórmulas que **no** son bien formadas. Veamos por qué:
  - $p \vee (q \rightarrow r)$
  - $\vee \neg q$
  - $q \rightarrow r$
  - $((pp) \wedge q)$

¡Atención!

No alcanza con indicar si es una fbf o no, **hay** que justificar.

Tablas de verdad

Vamos a darle un poco de significado (i.e. semántica) a los operadores:

NEGACIÓN	
ϕ	$\neg\phi$
`True`	`False`
`False`	`True`

Tablas de verdad

Vamos a darle un poco de significado (i.e. semántica) a los operadores:

NEGACIÓN	
ϕ	$\neg\phi$
`True`	`False`
`False`	`True`

O - OR - DISYUNCIÓN		
ϕ	ψ	$(\phi \vee \psi)$
`False`	`False`	`False`
`True`	`False`	`True`
`False`	`True`	`True`
`True`	`True`	`True`

Tablas de verdad

Vamos a darle un poco de significado (i.e. semántica) a los operadores:

NEGACIÓN	
ϕ	$\neg\phi$
`True`	`False`
`False`	`True`

O - OR - DISYUNCIÓN		
ϕ	ψ	$(\phi \vee \psi)$
`False`	`False`	`False`
`True`	`False`	`True`
`False`	`True`	`True`
`True`	`True`	`True`

Y - AND - CONJUNCIÓN		
ϕ	ψ	$(\phi \wedge \psi)$
`False`	`False`	`False`
`True`	`False`	`False`
`False`	`True`	`False`
`True`	`True`	`True`

Tablas de verdad

Vamos a darle un poco de significado (i.e. semántica) a los operadores:

NEGACIÓN	
ϕ	$\neg\phi$
`True`	`False`
`False`	`True`

O - OR - DISYUNCIÓN		
ϕ	ψ	$(\phi \vee \psi)$
`False`	`False`	`False`
`True`	`False`	`True`
`False`	`True`	`True`
`True`	`True`	`True`

IMPLICACIÓN		
ϕ	ψ	$(\phi \rightarrow \psi)$
`False`	`False`	`True`
`True`	`False`	`False`
`False`	`True`	`True`
`True`	`True`	`True`

Y - AND - CONJUNCIÓN		
ϕ	ψ	$(\phi \wedge \psi)$
`False`	`False`	`False`
`True`	`False`	`False`
`False`	`True`	`False`
`True`	`True`	`True`

Tablas de verdad

Vamos a darle un poco de significado (i.e. semántica) a los operadores:

NEGACIÓN	
ϕ	$\neg\phi$
True	False
False	True

O - OR - DISYUNCIÓN		
ϕ	ψ	$(\phi \vee \psi)$
False	False	False
True	False	True
False	True	True
True	True	True

IMPLICACIÓN		
ϕ	ψ	$(\phi \rightarrow \psi)$
False	False	True
True	False	False
False	True	True
True	True	True

Y - AND - CONJUNCIÓN		
ϕ	ψ	$(\phi \wedge \psi)$
False	False	False
True	False	False
False	True	False
True	True	True

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

Tautologías, contradicciones y contingencias

Abusando un poco (bastante) de la notación formal, vamos a dar estas tres definiciones:

- **Tautologías:** Decimos que una fórmula ϕ es una tautología si su evaluación *da* `True` como resultado independientemente de los valores que tengan las variables proposicionales.
  Ejemplo: $(p \vee \neg p)$.

Tautologías, contradicciones y contingencias

Abusando un poco (bastante) de la notación formal, vamos a dar estas tres definiciones:

- **Tautologías:** Decimos que una fórmula ϕ es una tautología si su evaluación *da* `True` como resultado independientemente de los valores que tengan las variables proposicionales.
  Ejemplo: $(p \vee \neg p)$.
- **Contradicciones:** Decimos que una fórmula ϕ es una contradicción si su negación $\neg\phi$ es una tautología.
  Ejemplo: $(p \wedge \neg p)$.

Tautologías, contradicciones y contingencias

Abusando un poco (bastante) de la notación formal, vamos a dar estas tres definiciones:

- **Tautologías:** Decimos que una fórmula ϕ es una tautología si su evaluación *da* `True` como resultado independientemente de los valores que tengan las variables proposicionales.
  Ejemplo: $(p \vee \neg p)$.
- **Contradicciones:** Decimos que una fórmula ϕ es una contradicción si su negación $\neg\phi$ es una tautología.
  Ejemplo: $(p \wedge \neg p)$.
- **Contingencias:** Decimos que una fórmula ϕ es una contingencia si no es tautología ni contradicción.
  Ejemplo: $(p \vee q)$

Ejemplos

Queremos escribir en lógica proposicional los siguientes enunciados en lenguaje natural:

0. *El número natural 4 es mayor que el número natural 1.*

Ejemplos

Queremos escribir en lógica proposicional los siguientes enunciados en lenguaje natural:

0. *El número natural 4 es mayor que el número natural 1.*
1. *El número natural 4 es menor que el número natural 1.*

Ejemplos

Queremos escribir en lógica proposicional los siguientes enunciados en lenguaje natural:

0. *El número natural 4 es mayor que el número natural 1.*
1. *El número natural 4 es menor que el número natural 1.*
2. *Si x es un número par entonces 4+1 es un número impar.*

Ejemplos

Queremos escribir en lógica proposicional los siguientes enunciados en lenguaje natural:

0. *El número natural 4 es mayor que el número natural 1.*
1. *El número natural 4 es menor que el número natural 1.*
2. *Si x es un número par entonces 4+1 es un número impar.*
3. *Si x es un número par entonces x+1 es un número impar.*

Ejemplos

Queremos escribir en lógica proposicional los siguientes enunciados en lenguaje natural:

0. *El número natural 4 es mayor que el número natural 1.*
1. *El número natural 4 es menor que el número natural 1.*
2. *Si x es un número par entonces 4+1 es un número impar.*
3. *Si x es un número par entonces x+1 es un número impar.*
4. *Si x es un número par entonces x+1 es un número par.*