

# Algoritmos y Estructura de Datos (Algo 2)

1C 2024

Teóricas: Diego Garbervetsky

# Objetivos

- Presentar los conceptos básicos que hacen a la **solución algorítmica de problemas correcta y eficiente**
  - especificación, diseño, implementación, complejidad de cómputo.
- Introducir tipos **abstractos de datos**:
  - su especificación, diseño, análisis e implementación eficiente usando estructuras de datos.
- Presentar las soluciones concretas más reconocidas para los problemas fundamentales de **búsqueda y ordenamiento**.
- Resolver por computadora proyectos pequeños y medianos donde se apliquen las herramientas y técnicas aprendidas.

# Correlativas (son muy importantes)

- Intro a la Programación (Algo 1)
- Algebra

Consultas administrativas: [mariangonzalez@dc.uba.ar](mailto:mariangonzalez@dc.uba.ar)

# Estructura

- **Teóricas**
  - **martes, 17 a 22 hs (Aula Magna Pab1)**
- **Prácticas**
  - miércoles, 17 a 22 hs (**Aula aMagna Pab1**)
- **Laboratorios**
  - viernes, 17 a 19hs (**Aula Magna Pab1**) y 19 a 22 (**Labo a confirmar**)

# Evaluación y Aprobación

- 2 parciales teórico prácticos (con sus dos recuperatorios)
- 2 trabajos prácticos (con sus dos recuperatorios)
- Talleres obligatorios

# De IP a Algoritmos

- Contratos
  - Descripción en Lenguaje **Natural** vs Lenguaje **Formal**
  - **Testing** (ejecuta **algunos** casos) vs **Verificación Formal** (**analiza todos** los casos)
  - Funciones vs Funciones y Componentes (APIs / TADs)
- Programas sobre **secuencias** vs Estructuras de datos **más complejas**
- Contar operaciones vs análisis de complejidad
- Tipos Abstracto de Datos
  - Especificación
  - **Diseño** e Implementación
  - Verificación
  - Análisis de Complejidad
- Técnicas algorítmicas

# Especificación

- Foco en el qué y no el como (lo vimos en IP)
- Lenguaje formal
  - Para evitar confusiones
  - Para poder razonar sobre la especificación
  - Para poder verificar
- Lenguaje de la materia
  - Basado en Lógica de Primer Orden
  - Instanciado con tipos de datos que usamos en la materia
    - Enteros, Secuencias, Conjuntos, etc

# Ejemplo

```
proc menorElemDistintos(in s: seq<Z>): Z {  
    requiere { noNegativos(s) ∧ noHayRepetidos(s) }  
    asegura {  
        /* result es un indice válido de s */  
         $0 \leq result < |s| \wedge_L$   
        /* s[result] es el menor elemento de s */  
         $(\forall i : \mathbb{Z})((0 \leq i < |s|) \rightarrow_L s[result] \leq s[i])$   
    }  
}
```

```
pred noNegativos(s: seq<Z>) {  
     $(\forall i : \mathbb{Z}) (0 \leq i < |s|) \rightarrow_L s[i] \geq 0$   
}
```

```
pred noNegativos(s: seq<Z>) {  
     $(\forall e : \mathbb{Z}) e \in s \rightarrow_L \#apariciones(e, s) == 1$   
}
```

# Algoritmos y Estructuras de Datos

Repaso de Lógica Proposicional

2024

## Bibliografía

- ▶ Michael Huth y Mark Ryan, Logic in computer science. Modelling and Reasoning about Systems, Cambridge University Press, 2004.
- ▶ Dirk Van Dalen, Logic and Structure, Series Universitext, Springer, 4th edition, 2008.
- ▶ Steve Reeves y Michael Clarke, Logic for computer science, Addison-Wesley, 1990.
- ▶ Michael Genesereth y Eric Kao (Synthesis Lectures on Computer Science), Introduction to Logic, Morgan & Claypool Publishers, 2012.

# Por qué estudiar lógica

- ▶ Queremos usar lógica en nuestras especificaciones
- ▶ usamos lógica en nuestros programas
- ▶ Queremos lenguajes para modelar situaciones
- ▶ Queremos poder razonar y argumentar
- ▶ Queremos poder hacer esto formalmente
- ▶ y vamos a entender más sobre la computación y sus raíces

# Lógica proposicional (PROP) - sintaxis

- ▶ símbolos

$\neg$  ,  $\wedge$  ,  $\vee$  ,  $\rightarrow$  ,  $\leftrightarrow$  , (, )

- ▶ variables proposicionales (infinitas)

$p$  ,  $q$  ,  $r$  , ...

- ▶ fórmulas

- ▶ combinaciones apropiadas de símbolos y variables proposicionales
- ▶ Ejemplo de combinación inapropiada:  $(\wedge p($

# Lógica proposicional (PROP) - sintaxis

## Fórmulas

1. cualquier variable proposicional es una fórmula
2. si  $\phi$  es una fórmula,  $(\neg\phi)$  es una fórmula
3. si  $\phi$  y  $\psi$  son fórmulas,  $(\phi \wedge \psi)$  es una fórmula
4. si  $\phi$  y  $\psi$  son fórmulas,  $(\phi \vee \psi)$  es una fórmula
5. si  $\phi$  y  $\psi$  son fórmulas,  $(\phi \rightarrow \psi)$  es una fórmula
6. si  $\phi$  y  $\psi$  son fórmulas,  $(\phi \leftrightarrow \psi)$  es una fórmula

- ▶ Muy entre paréntesis: Las fórmulas son un ejemplo de un [conjunto inductivo](#)
- ▶ Vienen provistos de
  - ▶ Esquema de prueba para probar propiedades sobre ellos ([inducción estructural](#))
  - ▶ Esquema de recursión para definir funciones sobre el conjunto ([recursión estructural](#))
- ▶ No es tema primario del curso, quizás lo veremos de pasada, pero quería que lo supieran

# Lógica proposicional - sintaxis

## Ejemplos

$$((p \wedge q) \rightarrow r) \quad (p \vee p)$$

- ▶ ¿Y estas expresiones son fórmulas?

$$p(\wedge q), \neg p$$

- ▶ Convenciones de notación

- ▶ Precedencia:  $\wedge$  y  $\vee$  ligan más fuerte que  $\rightarrow$  y  $\leftrightarrow$ ,  $\neg$  liga más fuerte que los demás
- ▶ Omisión de paréntesis más externos y los de negaciones
- ▶ Asociatividad de  $\wedge$  y  $\vee$

## Semántica clásica

- ▶ Consiste en asignarle **valores de verdad** a las fórmulas
- ▶ El conjunto de valores de verdad es
$$\{\mathbf{T}, \mathbf{F}\}$$
- ▶ Dos enfoques para darle semántica a las fórmulas de PROP
  1. Tablas de verdad
  2. Valuaciones
- ▶ Son equivalentes

# Tablas de verdad

Conociendo el valor de las variables proposicionales de una fórmula, conocemos el valor de verdad de la fórmula

$\phi$	$(\neg\phi)$
T	F
F	T

$\phi$	$\psi$	$(\phi \wedge \psi)$
T	T	T
T	F	F
F	T	F
F	F	F

$\phi$	$\psi$	$(\phi \vee \psi)$
T	T	T
T	F	T
F	T	T
F	F	F

$\phi$	$\psi$	$(\phi \rightarrow \psi)$
T	T	T
T	F	F
F	T	T
F	F	T

$\phi$	$\psi$	$(\phi \leftrightarrow \psi)$
T	T	T
T	F	F
F	T	F
F	F	T

Ejemplo: tabla de verdad para  $((p \wedge q) \rightarrow r)$

$p$	$q$	$r$	$(p \wedge q)$	$((p \wedge q) \rightarrow r)$
T	T	T	T	T
T	T	F	T	F
T	F	T	F	T
T	F	F	F	T
F	T	T	F	T
F	T	F	F	T
F	F	T	F	T
F	F	F	F	T

# Ejemplo

Escribir la siguiente frase como una fórmula de lógica proposicional.

“Si Juan está cursando y no conoce a nadie entonces Juan todavía no tiene grupo”

## Solución 1:

$p$  = Juan está cursando

$q$  = Juan no conoce a nadie

$r$  = Juan no tiene grupo

$$(p \wedge q) \rightarrow r$$

## Solución 2:

$p$  = Juan está cursando

$q$  = Juan conoce a alguien

$r$  = Juan tiene grupo

$$(p \wedge \neg q) \rightarrow \neg r$$

# Valuaciones

- ▶ Una **valuación** es una función  $v : \mathcal{V} \rightarrow \{\mathbf{T}, \mathbf{F}\}$  que asigna valores de verdad a las variables proposicionales
- ▶ Una valuación **satisface** una proposición  $\phi$  si  $v \models \phi$  donde:

$$v \models p \quad \text{sii} \quad v(p) = \mathbf{T}$$

$$v \models \neg\phi \quad \text{sii} \quad v \not\models \phi \ (\text{i.e. no } v \models \phi)$$

$$v \models \phi \vee \psi \quad \text{sii} \quad v \models \phi \text{ o } v \models \psi$$

$$v \models \phi \wedge \psi \quad \text{sii} \quad v \models \phi \text{ y } v \models \psi$$

$$v \models \phi \rightarrow \psi \quad \text{sii} \quad v \not\models \phi \text{ o } v \models \psi$$

$$v \models \phi \leftrightarrow \psi \quad \text{sii} \quad (v \models \phi \text{ sii } v \models \psi)$$

# Tautologías y satisfactibilidad

Dadas fórmulas  $\phi$  y  $\psi$

- ▶  $\phi$  es lógicamente equivalente a  $\psi$  cuando  $v \models \phi$  si y sólo si  $v \models \psi$

Una fórmula  $\phi$  es

- ▶ una tautología si  $v \models \phi$  para toda valuación  $v$
- ▶ satisfactible si existe una valuación  $v$  tal que  $v \models \phi$
- ▶ insatisfactible si no es satisfactible

Un conjunto de fórmulas  $S$  es

- ▶ satisfactible si existe una valuación  $v$  tal que para todo  $\phi \in S$ , se tiene  $v \models \phi$
- ▶ insatisfactible si no es satisfactible

# Ejemplos

## Tautologías

- ▶  $p \rightarrow p$
- ▶  $\neg\neg p \rightarrow p$
- ▶  $(p \rightarrow q) \leftrightarrow (\neg q \rightarrow \neg p)$

## Fórmulas insatisfactibles

- ▶  $(\neg p \vee q) \wedge (\neg p \vee \neg q) \wedge p$
- ▶  $(p \rightarrow q) \wedge p \wedge \neg q$

# Tautologías e insatisfactibilidad

## Teorema

Una fórmula  $\phi$  es una tautología si y sólo si  $\neg\phi$  es insatisfacible

## Demostración.

- . Si  $\phi$  es tautología, para toda valuación  $v$ ,  $v \models \phi$ .  
Entonces,  $v \not\models \neg\phi$  (i.e.  $v$  no satisface  $\neg\phi$ ).
- ←. Si  $\neg\phi$  es insatisfacible, para toda valuación  $v$ ,  
 $v \not\models \neg\phi$ . Luego  $v \models \phi$ .

□

## Observación

Este resultado sugiere un método **indirecto** para probar que una fórmula  $\phi$  es una tautología, que es probar que  $\neg\phi$  es **insatisfacible**

## Relación entre tablas de verdad y valuaciones

- ▶ Filas de una tabla se corresponden con las valuaciones

	$p$	$q$	$r$	$(p \wedge q)$	$((p \wedge q) \rightarrow r)$
$v_1$	T	T	T	T	T
$v_2$	T	T	F	T	F
$v_3$	T	F	T	F	T
$v_4$	T	F	F	F	T
$v_5$	F	T	T	F	T
$v_6$	F	T	F	F	T
$v_7$	F	F	T	F	T
$v_8$	F	F	F	F	T

- ▶ ¿Cuántas valuaciones diferentes existen para una fórmula de  $n$  variables?

# Equivalencias entre fórmulas

► **Teorema.** Las siguientes son tautologías.

1. Idempotencia

$$(p \wedge p) \leftrightarrow p$$

$$(p \vee p) \leftrightarrow p$$

2. Asociatividad

$$(p \wedge q) \wedge r \leftrightarrow p \wedge (q \wedge r)$$

$$(p \vee q) \vee r \leftrightarrow p \vee (q \vee r)$$

3. Comutatividad

$$(p \wedge q) \leftrightarrow (q \wedge p)$$

$$(p \vee q) \leftrightarrow (q \vee p)$$

4. Distributividad

$$p \wedge (q \vee r) \leftrightarrow (p \wedge q) \vee (p \wedge r)$$

$$p \vee (q \wedge r) \leftrightarrow (p \vee q) \wedge (p \vee r)$$

5. Reglas de De Morgan

$$\neg(p \wedge q) \leftrightarrow \neg p \vee \neg q$$

$$\neg(p \vee q) \leftrightarrow \neg p \wedge \neg q$$

## Semántica trivaluada

- ▶ Supongamos que contamos con un símbolo relacional  $==$  que nos permite comparar números reales
- ▶ ¿Valor de verdad de las siguientes fórmulas?

$$1 == 1$$

$$(1 + 1) == 2$$

$$0,5 == 2/4$$

- ▶ ¿Y esta?

$$1/0 == 2$$

## Semántica trivaluada

Pasos para determinar si  $e_1 == e_2$  es verdadero o falso

1. Obtener el número real  $r_1$  denotado por  $e_1$
2. Obtener el número real  $r_2$  denotado por  $e_2$
3. Comparar  $r_1$  con  $r_2$  para determinar si son iguales o no

Consideremos

$$1/0 == 2$$

- ▶ Trabado en paso 1
- ▶ Expresión  $1/0$  no denota **ningún** número
- ▶  $1/0 == 2$  no es ni verdadera ni falsa porque no contamos con los números a comparar
- ▶ Le damos un valor especial:  $\perp$  (**indefinido**)

## Semántica trivaluada (secuencial)

Se llama **secuencial** porque ...

- ▶ los términos se evalúan de izquierda a derecha,
- ▶ la evaluación termina cuando se puede deducir el valor de verdad, aunque el resto esté indefinido.

Introducimos los operadores lógicos  $\wedge_L$  (y-luego, o *conditional and*, o **cand**),  $\vee_L$  (o-luego o *conditional or*, o **cor**).

$p$	$q$	$(p \wedge_L q)$
T	T	T
T	F	F
F	T	F
F	F	F
T	$\perp$	$\perp$
F	$\perp$	F
$\perp$	T	$\perp$
$\perp$	F	$\perp$
$\perp$	$\perp$	$\perp$

$p$	$q$	$(p \vee_L q)$
T	T	T
T	F	T
F	T	T
F	F	F
T	$\perp$	T
F	$\perp$	$\perp$
$\perp$	T	$\perp$
$\perp$	F	$\perp$
$\perp$	$\perp$	$\perp$

# Algoritmos y Estructuras de Datos

Lógica de predicados

2024

# Predicados

Considerar la frase

*Todo estudiante es más joven que algún profesor*

- ▶ En PROP lo representaríamos con una variable proposicional  $p$
- ▶ Se pierde información sobre la estructura lógica de la frase
  - ▶ ser estudiante
  - ▶ ser profesor
  - ▶ ser más joven que

# Individuos, predicados, variables y cuantificadores

*Todo estudiante es más joven que algún profesor*

$$\forall x.(E(x) \rightarrow (\exists y(P(y) \wedge J(x,y))))$$

- ▶ Individuos (entidad distintiva e indivisible):
  - ▶ Estudiantes y profesores
  - ▶ Denotados por las variables  $x$  e  $y$
- ▶ Predicados (predican sobre individuos):
  - ▶  $E(x)$ :  $x$  es estudiante
  - ▶  $P(x)$ :  $x$  es profesor
  - ▶  $J(x, y)$ :  $x$  es más joven que  $y$
- ▶ Cuantificadores
  - ▶  $\forall$ : Para todo
  - ▶  $\exists$ : Existe (para algún)

# Funciones

*Toda persona es menor que su madre biológica*

- ▶  $G(x)$ :  $x$  es persona
- ▶  $M(y, x)$ :  $y$  es la madre de  $x$

$$\forall x. \forall y. (G(x) \wedge M(y, x) \rightarrow J(x, y))$$

Funciones:

- ▶ Permiten representar objetos de manera más directa
- ▶ En lugar de escribir  $M(y, x)$  podemos denotar a  $y$  con  $m(x)$

$$\forall x. (G(x) \rightarrow J(x, m(x)))$$

Otro ejemplo: *Andrea y Pedro tienen la misma abuela materna*

$$m(m(andrea)) = m(m(pedro))$$

# Términos y fórmulas

La lógica de predicados habla sobre dos clases de cosas:

- ▶ Individuos: Personas, números, colores, bolitas, grafos, árboles, etc.
  - ▶ Las expresiones que denotan individuos se llaman **términos**
  - ▶ Ej: las variables
  - ▶ Ej: Constantes como *andrea* y expresiones como *m(andrea)*
- ▶ Valores de verdad
  - ▶ Las expresiones que denotan valores de verdad se llaman **fórmulas**
  - ▶ Ej: *J(x, y)*

## Cuantificadores

- ▶  $(\forall x) P(x)$ : Fórmula lógica. Afirma que **todos** los elementos cumplen la propiedad  $P$ .
  - ▶ Se lee “Para todo  $x$  se cumple  $P(x)$ ”
- ▶  $(\exists x) P(x)$ : Fórmula lógica. Afirma que **al menos un** elemento cumple la propiedad  $P$ .
  - ▶ Se lee “Existe al menos un  $x$  que cumple  $P(x)$ ”

## Cuantificadores

- ▶  $(\forall x) P(x)$ : Fórmula lógica. Afirma que **todos** los elementos cumplen la propiedad  $P$ .
  - ▶ Se lee “Para todo  $x$  se cumple  $P(x)$ ”
- ▶  $(\exists x) P(x)$ : Fórmula lógica. Afirma que **al menos un** elemento cumple la propiedad  $P$ .
  - ▶ Se lee “Existe al menos un  $x$  que cumple  $P(x)$ ”

## Cuantificadores tipados

*Syntax sugar* para nuestro lenguaje de especificación (más en la próxima clase) vamos a aplicar cuantificadores a elementos de un tipo de datos.

- ▶  $(\forall x : T) P(x)$ : Fórmula lógica. Afirma que **todos** los elementos de tipo  $T$  cumplen la propiedad  $P$ .
  - ▶ Se lee “Para todo  $x$  de tipo  $T$  se cumple  $P(x)$ ”
- ▶  $(\exists x : T) P(x)$ : Fórmula lógica. Afirma que **al menos un** elemento de tipo  $T$  cumple la propiedad  $P$ .
  - ▶ Se lee “Existe al menos un  $x$  de tipo  $T$  que cumple  $P(x)$ ”

## Cuantificadores tipados

*Syntax sugar* para nuestro lenguaje de especificación (más en la próxima clase) vamos a aplicar cuantificadores a elementos de un tipo de datos.

- ▶  $(\forall x : T) P(x)$ : Fórmula lógica. Afirma que **todos** los elementos de tipo  $T$  cumplen la propiedad  $P$ .
  - ▶ Se lee “Para todo  $x$  de tipo  $T$  se cumple  $P(x)$ ”
- ▶  $(\exists x : T) P(x)$ : Fórmula lógica. Afirma que **al menos un** elemento de tipo  $T$  cumple la propiedad  $P$ .
  - ▶ Se lee “Existe al menos un  $x$  de tipo  $T$  que cumple  $P(x)$ ”

## Ejemplos

- ▶ ¿qué dice el siguiente predicado?

## Ejemplos

- ▶ ¿qué dice el siguiente predicado?

$$n > 1 \wedge (\forall n' : \mathbb{Z})(1 < n' < n \rightarrow_L n \bmod n' \neq 0)$$

## Ejemplos

- ▶ ¿qué dice el siguiente predicado?

$$n > 1 \wedge (\forall n' : \mathbb{Z})(1 < n' < n \rightarrow_L n \bmod n' \neq 0)$$

(Observación:  $x \bmod y$  se indefine si  $y = 0$ )

- ▶ Todos los enteros entre 1 y 10 son pares:

$$(\forall n : \mathbb{Z})(1 \leq n \leq 10 \rightarrow n \bmod 2 = 0).$$

## Ejemplos

- ▶ ¿qué dice el siguiente predicado?

$$n > 1 \wedge (\forall n' : \mathbb{Z})(1 < n' < n \rightarrow_L n \bmod n' \neq 0)$$

(Observación:  $x \bmod y$  se indefine si  $y = 0$ )

- ▶ Todos los enteros entre 1 y 10 son pares:

$$(\forall n : \mathbb{Z})(1 \leq n \leq 10 \rightarrow n \bmod 2 = 0).$$

- ▶ Existe un entero entre 1 y 10 que es par:

$$(\exists n : \mathbb{Z})(1 \leq n \leq 10 \wedge n \bmod 2 = 0).$$

## Ejemplos

- ▶ ¿qué dice el siguiente predicado?

$$n > 1 \wedge (\forall n' : \mathbb{Z})(1 < n' < n \rightarrow_L n \bmod n' \neq 0)$$

(Observación:  $x \bmod y$  se indefine si  $y = 0$ )

- ▶ Todos los enteros entre 1 y 10 son pares:

$$(\forall n : \mathbb{Z})(1 \leq n \leq 10 \rightarrow n \bmod 2 = 0).$$

- ▶ Existe un entero entre 1 y 10 que es par:

$$(\exists n : \mathbb{Z})(1 \leq n \leq 10 \wedge n \bmod 2 = 0).$$

- ▶ En general, si queremos decir que todos los enteros  $x$  que cumplen  $P(x)$  también cumplen  $Q(x)$ , decimos:

$$(\forall x : \mathbb{Z})(P(x) \rightarrow Q(x)).$$

## Ejemplos

- ▶ ¿qué dice el siguiente predicado?

$$n > 1 \wedge (\forall n' : \mathbb{Z})(1 < n' < n \rightarrow_L n \bmod n' \neq 0)$$

(Observación:  $x \bmod y$  se indefine si  $y = 0$ )

- ▶ Todos los enteros entre 1 y 10 son pares:

$$(\forall n : \mathbb{Z})(1 \leq n \leq 10 \rightarrow n \bmod 2 = 0).$$

- ▶ Existe un entero entre 1 y 10 que es par:

$$(\exists n : \mathbb{Z})(1 \leq n \leq 10 \wedge n \bmod 2 = 0).$$

- ▶ En general, si queremos decir que todos los enteros  $x$  que cumplen  $P(x)$  también cumplen  $Q(x)$ , decimos:

$$(\forall x : \mathbb{Z})(P(x) \rightarrow Q(x)).$$

- ▶ Para decir que existe un entero que cumple  $P(x)$  y que también cumple  $Q(x)$ , decimos:

$$(\exists x : \mathbb{Z})(P(x) \wedge Q(x)).$$

## Algunas reglas de deducción

- ▶ La **negación** de un cuantificador universal es un cuantificador existencial, y viceversa:

$$\neg(\forall n)P(n) \leftrightarrow (\exists n)\neg P(n).$$

$$\neg(\exists n)P(n) \leftrightarrow (\forall n)\neg P(n).$$

## Algunas reglas de deducción

- ▶ La **negación** de un cuantificador universal es un cuantificador existencial, y viceversa:

$$\neg(\forall n)P(n) \leftrightarrow (\exists n)\neg P(n).$$

$$\neg(\exists n)P(n) \leftrightarrow (\forall n)\neg P(n).$$

- ▶ Un cuantificador universal **generaliza la conjunción**:

$$(\forall n : \mathbb{Z})P(n) \leftrightarrow P(1) \wedge P(2) \wedge P(3) \wedge \dots$$

## Algunas reglas de deducción

- ▶ La **negación** de un cuantificador universal es un cuantificador existencial, y viceversa:

$$\neg(\forall n)P(n) \leftrightarrow (\exists n)\neg P(n).$$

$$\neg(\exists n)P(n) \leftrightarrow (\forall n)\neg P(n).$$

- ▶ Un cuantificador universal **generaliza la conjunción**:

$$(\forall n : \mathbb{Z})P(n) \leftrightarrow P(1) \wedge P(2) \wedge P(3) \wedge \dots$$

- ▶ Un cuantificador universal **generaliza la disyunción**:

$$(\exists n : \mathbb{Z})P(n) \leftrightarrow P(1) \vee P(2) \vee P(3) \vee \dots$$

## Algunas reglas de deducción

- ▶ La **negación** de un cuantificador universal es un cuantificador existencial, y viceversa:

$$\neg(\forall n)P(n) \leftrightarrow (\exists n)\neg P(n).$$

$$\neg(\exists n)P(n) \leftrightarrow (\forall n)\neg P(n).$$

- ▶ Un cuantificador universal **generaliza la conjunción**:

$$(\forall n : \mathbb{Z})P(n) \leftrightarrow P(1) \wedge P(2) \wedge P(3) \wedge \dots$$

- ▶ Un cuantificador universal **generaliza la disyunción**:

$$(\exists n : \mathbb{Z})P(n) \leftrightarrow P(1) \vee P(2) \vee P(3) \vee \dots$$

Sintaxis de PRED

Semántica de PRED

# Lenguaje de primer orden

Un lenguaje de primer orden (LPO)  $\mathcal{L}$  consiste en:

1. Un conjunto  $\mathcal{F}$  de símbolos de función cada uno con aridad<sup>1</sup>  $n > 0$ :  $f_0, f_1, \dots, f_k$
2. Un conjunto numerable  $\mathcal{C}$  de constantes:  $c_0, c_1, \dots$
3. Un conjunto  $\mathcal{P}$  de símbolos de predicado cada uno con aridad  $n \geq 0$ :  $P_0, P_1, \dots, P_m, \dot{=}$ .
  - ▶ El símbolo de predicado  $\dot{=}$  denominará igualdad.

Ejemplo: Lenguaje de primer orden para la aritmética

Símbolos de función:  $S, +, *$ ; Constantes:  $0$ ; Símbolos de predicado:  $\dot{=}, <$ .

---

<sup>1</sup>Aridad=Número de argumentos que toman

## Términos de primer orden

Sea  $\mathcal{V} = \{x_0, x_1, \dots\}$  un conjunto numerable de variables y  $\mathcal{L}$  un LPO. El conjunto de  $\mathcal{L}$ -términos se define inductivamente como:

1. Toda constante de  $\mathcal{L}$  y toda variable es un  $\mathcal{L}$ -término
2. Si  $t_1, \dots, t_n \in \mathcal{L}$ -términos y  $f$  es un símbolo de función de aridad  $n$ , entonces  $f(t_1, \dots, t_n) \in \mathcal{L}$ -términos

En notación abreviada:

$$t ::= c \mid x \mid f(t, \dots, t)$$

Ejemplo: Aritmética (cont.)

$S(0), +(S(0), S(S(0))), *(S(x_1), +(x_2, S(x_3)))$

## Fórmulas atómicas

Sea  $\mathcal{V}$  un conjunto numerable de variables y  $\mathcal{L}$  un LPO. El conjunto de  $\mathcal{L}$ -fórmulas atómicas se define inductivamente como:

1. Todo símbolo de predicado de aridad 0 es una  $\mathcal{L}$ -fórmula atómica
  2. Si  $t_1, \dots, t_n \in \mathcal{L}$ -términos y  $P$  es un símbolo de predicado de aridad  $n$ , entonces  $P(t_1, \dots, t_n)$  es una  $\mathcal{L}$ -fórmula atómica
- Por ejemplo: Si  $t_1, t_2 \in \mathcal{L}$ -términos, entonces  $t_1 \doteq t_2$  es una  $\mathcal{L}$ -fórmula atómica

Ejemplo: Aritmética (cont.)

$$<(0, S(0)), <(x_1, +(S(0), x_2)), \doteq(0, S(S(x_1)))$$

## Fórmulas de primer orden

Sea  $\mathcal{V}$  un conjunto numerable de variables y  $\mathcal{L}$  un LPO. El conjunto de  $\mathcal{L}$ -fórmulas se define inductivamente como:

1. Toda  $\mathcal{L}$ -fórmula atómica es una  $\mathcal{L}$ -fórmula
2. Si  $\phi, \psi \in \mathcal{L}$ -fórmulas, entonces  $(\phi \wedge \psi), (\phi \vee \psi), (\phi \rightarrow \psi), (\phi \leftrightarrow \psi)$  y  $(\neg\phi)$  son  $\mathcal{L}$ -fórmulas
3. Para toda variable  $x_i$  y cualquier  $\mathcal{L}$ -fórmula  $\phi$ ,  $(\forall x_i.\phi)$  y  $(\exists x_i.\phi)$  son  $\mathcal{L}$ -fórmulas

## Variables libres y ligadas

- ▶ Una ocurrencia de  $x$  en  $\phi$  es ligada si  $x$  ocurre en un subérmino de la forma  $\forall x.\psi$  o  $\exists x.\psi$ .
- ▶ Una ocurrencia es libre si no es ligada.
- ▶ Una variable es libre (ligada) en una fórmula si ocurre libre (ligada) en la fórmula.

### Ejemplo

$$P(x) \wedge \forall x.(R(x, y) \rightarrow \exists z.P(z))$$

- ▶  $y$  es libre
- ▶  $z$  es ligada
- ▶  $x$  es libre y ligada

## Variables libres y ligadas

- ▶ Usamos  $FV(\phi)$  y  $BV(\phi)$  para referirnos al conjunto de las variables libres y ligadas de  $\phi$ , respectivamente (Free y Bounded)
- ▶  $FV(\phi)$  y  $BV(\phi)$  se pueden definir por inducción estructural en  $\phi$

### Ejemplo

Si  $\phi = \forall x.(R(x, y) \rightarrow P(x))$ , entonces  $FV(\phi) = \{y\}$  y  $BV(\phi) = \{x\}$

- ▶ **Sentencia:** fórmula cerrada (i.e. sin variables libres)

Sintaxis de PRED

Semántica de PRED

## Estructura de primer orden

Dado un lenguaje de primer orden  $\mathcal{L}$ , una estructura para  $\mathcal{L}$ ,  $\mathcal{M}$ , es un par

$$\mathcal{M} = (M, I)$$

donde

- ▶  $M$  (universo) es un conjunto no vacío
- ▶  $I$  (función de interpretación) asigna funciones y predicados sobre  $M$  a símbolos de  $\mathcal{L}$  de la siguiente manera:
  1. Para toda constante  $c$ ,  $I(c) \in M$
  2. Para todo  $f$  de aridad  $n > 0$ ,  $I(f) : M^n \rightarrow M$
  3. Para todo predicado  $P$  de aridad  $n \geq 0$   $I(P) \subseteq M^n$
  4.  $I(\equiv)$  es la relación de identidad sobre  $M$

# Asignación

## Asignación

Sea  $\mathcal{M}$  una estructura para  $\mathcal{L}$ . Una **asignación** es una función  
 $s : \mathcal{V} \rightarrow M$

Dado  $s$  podemos definir  $\hat{s}$  que se puede aplicar a términos para obtener el individuo del universo que denota

## Extensión de una asignación a términos

$$\begin{aligned}\hat{s}(x) &\stackrel{\text{def}}{=} s(x) \\ \hat{s}(c) &\stackrel{\text{def}}{=} I(c) \\ \hat{s}(f(t_1, \dots, t_n)) &\stackrel{\text{def}}{=} I(f)(\hat{s}(t_1), \dots, \hat{s}(t_n))\end{aligned}$$

Nota: a veces abusamos de la notación y escribimos simplemente  $s$  en lugar de  $\hat{s}$

# Satisfactibilidad

## Satisfactibilidad

La relación  $s \models_{\mathcal{M}} \phi$  establece que la asignación  $s$  satisface la fórmula  $\phi$  en la estructura  $\mathcal{M}$

- ▶ Vamos a definir la relación  $s \models_{\mathcal{M}} \phi$  de manera formal usando inducción estructural en  $\phi$
- ▶ Si  $s$  es una asignación y  $a \in M$ , usamos la notación  $s[x \leftarrow a]$  para denotar la asignación que se comporta igual que  $s$  salvo en el elemento  $x$ , en cuyo caso retorna  $a$

# Satisfactibilidad

La relación  $s \models_M \phi$  se define inductivamente como:

$$s \models_M P(t_1, \dots, t_n) \quad \text{sii} \quad (\widehat{s}(t_1), \dots, \widehat{s}(t_n)) \in I(P)$$

$$s \models_M \neg\phi \quad \text{sii} \quad s \not\models_M \phi$$

$$s \models_M (\phi \wedge \psi) \quad \text{sii} \quad s \models_M \phi \text{ y } s \models_M \psi$$

$$s \models_M (\phi \vee \psi) \quad \text{sii} \quad s \models_M \phi \text{ o } s \models_M \psi$$

$$s \models_M (\phi \rightarrow \psi) \quad \text{sii} \quad s \not\models_M \phi \text{ o } s \models_M \psi$$

$$s \models_M (\phi \leftrightarrow \psi) \quad \text{sii} \quad (s \models_M \phi \text{ sii } s \models_M \psi)$$

$$s \models_M \forall x_i.\phi \quad \text{sii} \quad s[x_i \leftarrow a] \models_M \phi \text{ para todo } a \in M$$

$$s \models_M \exists x_i.\phi \quad \text{sii} \quad s[x_i \leftarrow a] \models_M \phi \text{ para algún } a \in M$$

- ▶ Una fórmula  $\phi$  es **satisfactible** en  $\mathcal{M}$  si existe una asignación  $s$  tal que

$$s \models_{\mathcal{M}} \phi$$

- ▶ Una fórmula  $\phi$  es **satisfactible** si existe un  $\mathcal{M}$  tal que  $\phi$  es satisfactible en  $\mathcal{M}$ . En caso contrario se dice que  $\phi$  es **insatisfactible**.

- ▶ Una fórmula  $\phi$  es **válida o verdadera** en  $\mathcal{M}$  si

$$s \models_{\mathcal{M}} \phi, \text{ para toda asignación } s$$

- ▶ Una fórmula  $\phi$  es **válida** si es válida en toda estructura  $\mathcal{M}$ .
- ▶ **Nota:**  $\phi$  es válida si  $\neg\phi$  es insatisfactible.

## Ejemplos de fórmulas válidas

- ▶  $\phi \wedge \neg\phi$
- ▶  $\forall x.\phi(x) \rightarrow \exists x.\phi(x)$
- ▶  $\forall x.\phi(x) \rightarrow \neg\forall x.\neg\phi(x)$
- ▶  $\forall x.\forall y.\phi(x, y) \rightarrow \forall y.\forall x.\phi(x, y)$
- ▶  $\forall x.(\phi(x) \wedge \psi(x)) \rightarrow \forall x.\phi(x) \wedge \forall x.\psi(x)$
- ▶ y las que vimos antes...

# Resumen de PRED

- ▶ Sintaxis
  - ▶ Lenguaje de primer orden  $\mathcal{L}$
  - ▶ Términos sobre  $\mathcal{L}$  (denotan individuos)
  - ▶ Fórmulas sobre  $\mathcal{L}$  (denotan valores de verdad)
- ▶ Semántica
  - ▶ Estructuras: universo+interpretación de los símbolos de  $\mathcal{L}$