

# Paradigmas de Programación

## Resolución lógica

**1er cuatrimestre de 2025**

Departamento de Computación

Facultad de Ciencias Exactas y Naturales

Universidad de Buenos Aires

Breve introducción a Prolog

Resolución para lógica proposicional

Resolución para lógica de primer orden

# Introducción a Prolog

## Ejemplo — genealogía del panteón mitológico griego

```
padre(cronos, zeus).  
padre(zeus, atenea).  
padre(zeus, hefesto).  
padre(zeus, ares).
```

```
abuelo(X, Y) :- padre(X, Z), padre(Z, Y).
```

?- padre(zeus, atenea).	?- abuelo(cronos, X).
>> true.	>> X = atenea ;
?- padre(zeus, cronos).	>> X = hefesto ;
>> false.	>> X = ares.
?- abuelo(X, atenea).	?- abuelo(X, Y).
>> X = cronos.	>> X = cronos, Y = atenea ;
?- abuelo(X, zeus).	>> X = cronos, Y = hefesto ;
>> false.	>> X = cronos, Y = ares.

# Introducción a Prolog

Prolog opera con **términos de primer orden**:

X Y succ(succ(zero)) bin(I, R, D) ...

Las **fórmulas atómicas** son de la forma **pred**( $t_1, \dots, t_n$ ):

padre(zeus, atenea) suma(zero, X, X)

# Introducción a Prolog

Un programa es un conjunto de **reglas**. Cada regla es de la forma:

$$\sigma \quad :- \quad \tau_1, \dots, \tau_n.$$

Ej.: **abuelo**(X, Y) **:-** **padre**(X, Z), **padre**(Z, Y).

donde  $\sigma, \tau_1, \dots, \tau_n$  son fórmulas atómicas.

Las reglas en las que  $n = 0$  se llaman **hechos** y se escriben:

$$\sigma. \quad \text{Ej.: } \text{padre}(\text{zeus}, \text{ares}).$$

Las reglas tienen la siguiente interpretación lógica:

$$\forall X_1 \dots \forall X_k. ((\tau_1 \wedge \dots \wedge \tau_n) \Rightarrow \sigma)$$

donde  $X_1, \dots, X_k$  son todas las variables libres de las fórmulas.

$$\text{Ej.: } \forall X. \forall Y. \forall Z. ((\text{padre}(X, Z) \wedge \text{padre}(Z, Y)) \Rightarrow \text{abuelo}(X, Y))$$

# Introducción a Prolog

Una **consulta** es de la forma:

$?- \sigma_1, \dots, \sigma_n$   
Ej.:  $?- \text{abuelo}(X, \text{ares})$ .

Las consultas tienen la siguiente interpretación lógica:

$$\exists X_1 \dots \exists X_k. (\sigma_1 \wedge \dots \wedge \sigma_n)$$

donde  $X_1, \dots, X_k$  son todas las variables libres de las fórmulas.

El entorno de Prolog busca demostrar la fórmula  $\tau$  de la consulta.

En realidad busca *refutar*  $\neg\tau$ , o sea, demostrar  $\neg\tau \Rightarrow \perp$

La búsqueda de la refutación se basa en el **método de resolución**.

Breve introducción a Prolog

Resolución para lógica proposicional

Resolución para lógica de primer orden

# Resolución para lógica proposicional

Entrada: una fórmula  $\sigma$  de la lógica proposicional.

Salida: un booleano que indica si  $\sigma$  es válida.

## Método de resolución

1. Escribir  $\neg\sigma$  como un conjunto  $\mathcal{C}$  de **cláusulas**.  
(Pasar a *forma clausal*).

2. Buscar una **refutación** de  $\mathcal{C}$ .

Una refutación de  $\mathcal{C}$  es una derivación de  $\mathcal{C} \vdash \perp$ .

Si se encuentra una refutación de  $\mathcal{C}$ :

Vale  $\neg\sigma \vdash \perp$ . Es decir,  $\neg\sigma$  es insatisfactible/contradicción.

Luego vale  $\vdash \sigma$ . Es decir,  $\sigma$  es válida/tautología.

Si no se encuentra una refutación de  $\mathcal{C}$ :

No vale  $\neg\sigma \vdash \perp$ . Es decir,  $\neg\sigma$  es satisfactible.

Luego no vale  $\vdash \sigma$ . Es decir,  $\sigma$  no es válida/tautología.



## Pasaje a forma clausal

Una fórmula se pasa a forma clausal aplicando las siguientes reglas. Todas las reglas transforman la fórmula en otra equivalente.

**Paso 1.** Deshacerse del conectivo “ $\Rightarrow$ ”:

$$\sigma \Rightarrow \tau \quad \longrightarrow \quad \neg\sigma \vee \tau$$

La fórmula resultante sólo usa los conectivos  $\{\neg, \vee, \wedge\}$ .

**Paso 2.** Empujar el conectivo “ $\neg$ ” hacia adentro:

$$\neg(\sigma \wedge \tau) \quad \longrightarrow \quad \neg\sigma \vee \neg\tau$$

$$\neg(\sigma \vee \tau) \quad \longrightarrow \quad \neg\sigma \wedge \neg\tau$$

$$\neg\neg\sigma \quad \longrightarrow \quad \sigma$$

La fórmula resultante está en **forma normal negada** (NNF):

$$\sigma_{\text{nnf}} ::= \mathbf{P} \mid \neg\mathbf{P} \mid \sigma_{\text{nnf}} \wedge \sigma_{\text{nnf}} \mid \sigma_{\text{nnf}} \vee \sigma_{\text{nnf}}$$

# Pasaje a forma clausal

**Paso 3.** Distribuir  $\vee$  sobre  $\wedge$ :

$$\begin{aligned}\sigma \vee (\tau \wedge \rho) &\longrightarrow (\sigma \vee \tau) \wedge (\sigma \vee \rho) \\ (\sigma \wedge \tau) \vee \rho &\longrightarrow (\sigma \vee \rho) \wedge (\tau \vee \rho)\end{aligned}$$

La fórmula resultante está en **forma normal conjuntiva** (CNF).  
Una fórmula en CNF es conjunción de disyunciones de literales  
(asumiendo que permitimos asociar libremente  $\wedge$  y  $\vee$ ):

Fórmulas en CNF	$\sigma_{\text{cnf}}$	$::=$	$(\kappa_1 \wedge \kappa_2 \wedge \dots \wedge \kappa_n)$
Cláusulas	$\kappa$	$::=$	$(\ell_1 \vee \ell_2 \vee \dots \vee \ell_m)$
Literales	$\ell$	$::=$	$\mathbf{P} \mid \neg \mathbf{P}$

## Pasaje a forma clausal

Por último, usando el hecho de que la disyunción ( $\vee$ ) es:

$$\text{asociativa} \quad \sigma \vee (\tau \vee \rho) \iff (\sigma \vee \tau) \vee \rho$$

$$\text{conmutativa} \quad \sigma \vee \tau \iff \tau \vee \sigma$$

$$\text{idempotente} \quad \sigma \vee \sigma \iff \sigma$$

notamos una cláusula (disyunción de literales) como un conjunto:

$$(\ell_1 \vee \ell_2 \vee \dots \vee \ell_n) \quad \text{se representa} \quad \{\ell_1, \ell_2, \dots, \ell_n\}$$

Análogamente, usando el hecho de que la conjunción ( $\wedge$ ) es asociativa, conmutativa e idempotente notamos una conjunción de cláusulas como un conjunto:

$$(\kappa_1 \wedge \kappa_2 \wedge \dots \wedge \kappa_n) \quad \text{se representa} \quad \{\kappa_1, \kappa_2, \dots, \kappa_n\}$$

# Pasaje a forma clausal

## Resumen — pasaje a forma clausal

1. Reescribir  $\Rightarrow$  usando  $\neg$  y  $\vee$ .
2. Pasar a f.n. negada, empujando  $\neg$  hacia adentro.
3. Pasar a f.n. conjuntiva, distribuyendo  $\vee$  sobre  $\wedge$ .

# Pasaje a forma clausal

## Ejemplo — pasaje a forma clausal

Queremos ver que  $\sigma \equiv (((\mathbf{P} \Rightarrow (\mathbf{Q} \wedge \mathbf{R})) \wedge \mathbf{P}) \Rightarrow \mathbf{Q})$  es válida.

Primero la negamos:  $\neg\sigma \equiv \neg(((\mathbf{P} \Rightarrow (\mathbf{Q} \wedge \mathbf{R})) \wedge \mathbf{P}) \Rightarrow \mathbf{Q})$ .

Pasamos  $\neg\sigma$  a forma clausal:

$$\begin{aligned} & \neg(((\mathbf{P} \Rightarrow (\mathbf{Q} \wedge \mathbf{R})) \wedge \mathbf{P}) \Rightarrow \mathbf{Q}) \\ \rightarrow & \neg(\neg(\neg(\neg\mathbf{P} \vee (\mathbf{Q} \wedge \mathbf{R})) \wedge \mathbf{P}) \vee \mathbf{Q}) \\ \rightarrow & (\neg\neg(\neg(\neg\mathbf{P} \vee (\mathbf{Q} \wedge \mathbf{R})) \wedge \mathbf{P}) \wedge \neg\mathbf{Q}) \\ \rightarrow & (((\neg\mathbf{P} \vee (\mathbf{Q} \wedge \mathbf{R})) \wedge \mathbf{P}) \wedge \neg\mathbf{Q}) \\ \rightarrow & (((\neg\mathbf{P} \vee \mathbf{Q}) \wedge (\neg\mathbf{P} \vee \mathbf{R})) \wedge \mathbf{P}) \wedge \neg\mathbf{Q}) \\ \rightarrow & (\neg\mathbf{P} \vee \mathbf{Q}) \wedge (\neg\mathbf{P} \vee \mathbf{R}) \wedge \mathbf{P} \wedge \neg\mathbf{Q} \end{aligned}$$

La forma clausal es:

$$\mathcal{C} = \{\{\neg\mathbf{P}, \mathbf{Q}\}, \{\neg\mathbf{P}, \mathbf{R}\}, \{\mathbf{P}\}, \{\neg\mathbf{Q}\}\}$$

# Refutación

Una vez obtenido un conjunto de cláusulas  $\mathcal{C} = \{\kappa_1, \dots, \kappa_n\}$ , se busca una **refutación**, es decir, una demostración de  $\mathcal{C} \vdash \perp$ .

El método de refutación se basa en la siguiente regla de deducción:

## Regla de resolución

$$\frac{\mathbf{P} \vee \ell_1 \vee \dots \vee \ell_n \quad \neg \mathbf{P} \vee \ell'_1 \vee \dots \vee \ell'_m}{\ell_1 \vee \dots \vee \ell_n \vee \ell'_1 \vee \dots \vee \ell'_m}$$

Escrita con notación de cláusulas:

$$\frac{\{\mathbf{P}, \ell_1, \dots, \ell_n\} \quad \{\neg \mathbf{P}, \ell'_1, \dots, \ell'_m\}}{\{\ell_1, \dots, \ell_n, \ell'_1, \dots, \ell'_m\}}$$

La conclusión se llama la **resolvente** de las premisas.

# Refutación

Entrada: un conjunto de cláusulas  $\mathcal{C}_0 = \{\kappa_1, \dots, \kappa_n\}$ .

Salida: **SAT/INSAT** indicando si  $\mathcal{C}_0$  es insatisfactible ( $\mathcal{C}_0 \vdash \perp$ ).

## Algoritmo de refutación

Sea  $\mathcal{C} := \mathcal{C}_0$ . Repetir mientras sea posible:

1. Si  $\{\}$   $\in \mathcal{C}$ , devolver **INSAT**.
2. Elegir dos cláusulas  $\kappa, \kappa' \in \mathcal{C}$ , tales que:

$$\kappa = \{\mathbf{P}, \ell_1, \dots, \ell_n\}$$

$$\kappa' = \{\neg \mathbf{P}, \ell'_1, \dots, \ell'_m\}$$

La resolvente  $\rho = \{\ell_1, \dots, \ell_n, \ell'_1, \dots, \ell'_m\}$  no está en  $\mathcal{C}$ .

Si no es posible, devolver **SAT**.

3. Tomar  $\mathcal{C} := \mathcal{C} \cup \{\rho\}$  y volver al paso 1.

# Refutación

## Ejemplo — método de resolución

Queremos demostrar  $\sigma \equiv (((\mathbf{P} \Rightarrow (\mathbf{Q} \wedge \mathbf{R})) \wedge \mathbf{P}) \Rightarrow \mathbf{Q})$ .

Equivalentemente, veamos que  $\neg\sigma \vdash \perp$ .

La forma clausal de  $\neg\sigma$  era:

$$\mathcal{C} = \underbrace{\{\{\neg\mathbf{P}, \mathbf{Q}\}\}}_{\boxed{1}}, \underbrace{\{\{\neg\mathbf{P}, \mathbf{R}\}\}}_{\boxed{2}}, \underbrace{\{\{\mathbf{P}\}\}}_{\boxed{3}}, \underbrace{\{\{\neg\mathbf{Q}\}\}}_{\boxed{4}}$$

- ▶ De  $\boxed{1}$  y  $\boxed{3}$  obtenemos la resolvente  $\boxed{5} = \{\mathbf{Q}\}$ .
- ▶ De  $\boxed{4}$  y  $\boxed{5}$  obtenemos la resolvente  $\{\}$ .
- ▶ Luego  $\mathcal{C} \vdash \perp$ .  
Luego  $\neg\sigma \vdash \perp$ .  
Luego  $\vdash \sigma$ .



# Corrección del método de resolución proposicional

## Teorema (corrección del pasaje a forma clausal)

Dada una fórmula  $\sigma$ :

1. El pasaje a forma clausal termina.
2. El conjunto de cláusulas  $\mathcal{C}$  obtenido es equivalente a  $\sigma$ .  
Es decir,  $\vdash \sigma \iff \mathcal{C}$ .

# Corrección del método de resolución proposicional

## Teorema (corrección del algoritmo de refutación)

Dado un conjunto de cláusulas  $\mathcal{C}_0$ :

1. El algoritmo de refutación termina.
2. El algoritmo retorna **INSAT** si y sólo si  $\mathcal{C}_0 \vdash \perp$ .

Ideas de la demostración:

1. Si en  $\mathcal{C}_0$  aparecen  $n$  literales distintos, se pueden formar  $2^n$  cláusulas posibles. Cada paso agrega una cláusula. Luego el algoritmo no puede tomar más de  $2^n$  pasos.
2. ( $\Rightarrow$ ). El algoritmo preserva el invariante de que para cada cláusula  $\kappa \in \mathcal{C}$  se tiene que  $\mathcal{C}_0 \vdash \kappa$ . La observación clave es que si  $\kappa, \kappa' \in \mathcal{C}$  y  $\rho$  es la resolvente, entonces  $\kappa, \kappa' \vdash \rho$ .
2. ( $\Leftarrow$ ). Más difícil. Se puede probar por inducción en el número de variables proposicionales que aparecen en  $\mathcal{C}_0$ .

Ver *Handbook of Proof Theory*. Samuel R. Buss (editor). Elsevier, 1998. Sección 2.6.

Breve introducción a Prolog

Resolución para lógica proposicional

Resolución para lógica de primer orden

# Resolución para lógica de primer orden

Entrada: una fórmula  $\sigma$  de la lógica de primer orden.

Salida: un booleano indicando si  $\sigma$  es válida.

**Si  $\sigma$  es válida, el método siempre termina.**

**Si  $\sigma$  no es válida, el método puede no terminar.**

Método de resolución de primer orden  
(Procedimiento de semi-decisión)

1. Escribir  $\neg\sigma$  como un conjunto  $\mathcal{C}$  de **cláusulas**.
2. Buscar una **refutación** de  $\mathcal{C}$ .

Si existe alguna refutación, el método encuentra alguna.

Si no existe una refutación, el método puede “colgarse”.

## Pasaje a forma clausal en lógica de primer orden

Una fórmula se pasa a forma clausal aplicando las siguientes reglas.

**Paso 1.** Deshacerse del conectivo “ $\Rightarrow$ ”:

$$\sigma \Rightarrow \tau \longrightarrow \neg \sigma \vee \tau$$

La fórmula resultante sólo usa los conectivos  $\{\neg, \vee, \wedge, \forall, \exists\}$ .

**Paso 2.** Empujar el conectivo “ $\neg$ ” hacia adentro:

$$\neg(\sigma \wedge \tau) \longrightarrow \neg \sigma \vee \neg \tau$$

$$\neg(\sigma \vee \tau) \longrightarrow \neg \sigma \wedge \neg \tau$$

$$\neg \neg \sigma \longrightarrow \sigma$$

$$\neg \forall X. \sigma \longrightarrow \exists X. \neg \sigma$$

$$\neg \exists X. \sigma \longrightarrow \forall X. \neg \sigma$$

La fórmula resultante está en **forma normal negada** (NNF):

$$\begin{aligned} \sigma_{\text{nnf}} \quad ::= & \quad \mathbf{P}(t_1, \dots, t_n) \mid \neg \mathbf{P}(t_1, \dots, t_n) \mid \sigma_{\text{nnf}} \wedge \sigma_{\text{nnf}} \mid \sigma_{\text{nnf}} \vee \sigma_{\text{nnf}} \\ & \mid \forall X. \sigma_{\text{nnf}} \mid \exists X. \sigma_{\text{nnf}} \end{aligned}$$

## Pasaje a forma clausal en lógica de primer orden

**Paso 3.** Extraer los cuantificadores ( $\forall/\exists$ ) hacia afuera.

Se asume siempre  $x \notin \text{fv}(\tau)$ :

$$\begin{array}{ll} (\forall x. \sigma) \wedge \tau \rightarrow \forall x. (\sigma \wedge \tau) & \tau \wedge (\forall x. \sigma) \rightarrow \forall x. (\tau \wedge \sigma) \\ (\forall x. \sigma) \vee \tau \rightarrow \forall x. (\sigma \vee \tau) & \tau \vee (\forall x. \sigma) \rightarrow \forall x. (\tau \vee \sigma) \\ (\exists x. \sigma) \wedge \tau \rightarrow \exists x. (\sigma \wedge \tau) & \tau \wedge (\exists x. \sigma) \rightarrow \exists x. (\tau \wedge \sigma) \\ (\exists x. \sigma) \vee \tau \rightarrow \exists x. (\sigma \vee \tau) & \tau \vee (\exists x. \sigma) \rightarrow \exists x. (\tau \vee \sigma) \end{array}$$

Todas las reglas transforman la fórmula en otra equivalente.

La fórmula resultante está en **forma normal prenexa**:

$$\sigma_{\text{pre}} ::= Q_1 x_1. Q_2 x_2. \dots Q_n x_n. \tau$$

donde cada  $Q_i$  es un cuantificador  $\{\forall, \exists\}$

y  $\tau$  representa una fórmula en NNF libre de cuantificadores.

# Pasaje a forma clausal en lógica de primer orden

**Paso 4.** Deshacerse de los cuantificadores existenciales ( $\exists$ ).  
Para ello se usa la siguiente técnica de Herbrand y Skolem:

- ▶ ¿Cómo eliminamos los  $\exists$  sin cambiar la satisfactibilidad?
- ▶ Introducimos “testigos” para ellos.
  - ▶ Todo cuantificador existencial se instancia en una constante o función de skolem.
  - ▶ Ejemplo:  $\exists x.P(x)$  se skolemiza a  $P(c)$  donde  $c$  es una nueva constante que se agrega al lenguaje de primer orden.
  - ▶ Estas funciones y constantes se suelen conocer como **parámetros**.

# Skolemización

Cada ocurrencia de una subfórmula

$$\exists x.B$$

en  $A$  se reemplaza por

$$B\{x := f(x_1, \dots, x_n)\}$$

donde:

- ▶  $\bullet\{\bullet := \bullet\}$  es la operación usual de Sustitución (sustituir todas las ocurrencias libres de una variable en una expresión - fórmula o término - por otra expresión).
- ▶  $f$  es un símbolo de función nuevo y las  $x_1, \dots, x_n$  son las variables de las que depende  $x$  en  $B$ .
- ▶ Si  $\exists x.B$  forma parte de una fórmula mayor,  $x$  solo depende de las **variables libres** de  $B$  (por ejemplo, en  $\forall z.\forall y.\exists x.P(y, x)$  la  $x$  depende de  $y$ ).



# Forma normal de Skolem

- ▶ Sea  $A$  una sentencia rectificada en **forma normal negada**:
  - ▶ Una fórmula está rectificada si todos sus cuantificadores ligan variables distintas entre sí, y a la vez distintas de todas las variables libres.
- ▶ Reemplazar sucesivamente cada ocurrencia de una subfórmula de la forma  $\exists X.B$  en  $A$  por  $B\{X := f_X(y_1, \dots, y_m)\}$  donde:
  - ▶  $f_V(B) = \{x, y_1, \dots, y_m\}$ .
  - ▶ Como  $A$  está rectificada, cada  $f_x$  es única.
  - ▶ Caso especial ( $m = 0$ ): Se utiliza una constante (o símbolo de función de aridad 0)  $c_x$ .  
 $\exists X.B$  se reemplaza por  $B\{X := c_X\}$

## Ejemplos

Considerar la fórmula

$$\forall x. \left( S(x) \vee \exists y. (R(y) \wedge \forall z. (P(y, z) \vee \exists u. Q(x, u))) \right) \vee \exists w. R(w)$$

La forma normal de Skolem es:

$$\forall x. (S(x) \vee (R(g(x)) \wedge \forall z. (P(g(x), z) \vee Q(x, f(x))))) \vee R(c)$$

# Ejemplos

- Considere la sentencia:

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

- 1. Alternativa 1 (rojo, azul)

- 1.1  $\forall x. \exists y. \exists z. R(x, y, z)$

- 1.2  $\forall x. \exists z. R(x, f(x), z)$

- 1.3  $\forall x. R(x, f(x), g(x))$

- 2. Alternativa 2 (azul, rojo)

- 2.1  $\forall x. \exists y. \exists z. R(x, y, z)$

- 2.2  $\forall x. \exists y. R(x, y, h(x, y))$

- 2.3  $\forall x. R(x, k(x), h(x, k(x)))$

- 3. La skolemización no es determinística.

- Es mejor skolemizar de afuera hacia adentro.

## Pasaje a forma clausal en lógica de primer orden

La Skolemización preserva la **satisfactibilidad**.  
Pero no siempre produce fórmulas equivalentes.  
Es decir **no preserva la validez**.

Ejemplo — la Skolemización no preserva la validez

$$\underbrace{\exists x. (P(0) \Rightarrow P(x))}_{\text{válida}}$$

$$\underbrace{P(0) \Rightarrow P(c)}_{\text{inválida}}$$

## Pasaje a forma clausal en lógica de primer orden

Dada una fórmula en forma normal prenexa y **cerrada**, se aplica la regla:

$$\forall X_1. \dots \forall X_n. \exists Y. \sigma \quad \longrightarrow \quad \forall X_1. \dots \forall X_n. \sigma \{Y := f(X_1, \dots, X_n)\}$$

donde  $f$  es un símbolo de función nuevo de aridad  $n \geq 0$ .

**Forma normal de Skolem:**

$$\sigma_{Sk} ::= \forall X_1 X_2 \dots X_n. \tau$$

donde  $\tau$  representa una fórmula en NNF libre de cuantificadores.

## Pasaje a forma clausal en lógica de primer orden

**Paso 5.** Dada una fórmula en forma normal de Skolem:

$$\forall X_1 X_2 \dots X_n. \varphi \quad (\varphi \text{ libre de cuantificadores})$$

se pasa  $\varphi$  a forma normal conjuntiva usando las reglas ya vistas:

$$\begin{aligned} \sigma \vee (\tau \wedge \rho) &\longrightarrow (\sigma \vee \tau) \wedge (\sigma \vee \rho) \\ (\sigma \wedge \tau) \vee \rho &\longrightarrow (\sigma \vee \rho) \wedge (\tau \vee \rho) \end{aligned}$$

El resultado es una fórmula de la forma:

$$\forall X_1 \dots X_n. \left( \begin{array}{l} (\ell_1^{(1)} \vee \dots \vee \ell_{m_1}^{(1)}) \\ \wedge (\ell_1^{(2)} \vee \dots \vee \ell_{m_2}^{(2)}) \\ \dots \\ \wedge (\ell_1^{(k)} \vee \dots \vee \ell_{m_k}^{(k)}) \end{array} \right)$$

## Pasaje a forma clausal en lógica de primer orden

**Paso 6.** Empujar los cuantificadores universales hacia adentro:

$$\forall \mathbf{X}_1 \dots \mathbf{X}_n. \left( \begin{array}{l} (\ell_1^{(1)} \vee \dots \vee \ell_{m_1}^{(1)}) \\ \wedge (\ell_1^{(2)} \vee \dots \vee \ell_{m_2}^{(2)}) \\ \dots \\ \wedge (\ell_1^{(k)} \vee \dots \vee \ell_{m_k}^{(k)}) \end{array} \right) \rightarrow \left( \begin{array}{l} \forall \mathbf{X}_1 \dots \mathbf{X}_n. (\ell_1^{(1)} \vee \dots \vee \ell_{m_1}^{(1)}) \\ \wedge \forall \mathbf{X}_1 \dots \mathbf{X}_n. (\ell_1^{(2)} \vee \dots \vee \ell_{m_2}^{(2)}) \\ \dots \\ \wedge \forall \mathbf{X}_1 \dots \mathbf{X}_n. (\ell_1^{(k)} \vee \dots \vee \ell_{m_k}^{(k)}) \end{array} \right)$$

Por último la **forma clausal** es:

$$\left\{ \begin{array}{l} \{\ell_1^{(1)}, \dots, \ell_{m_1}^{(1)}\}, \\ \{\ell_1^{(2)}, \dots, \ell_{m_2}^{(2)}\}, \\ \vdots \\ \{\ell_1^{(k)}, \dots, \ell_{m_k}^{(k)}\} \end{array} \right\}$$

# Pasaje a forma clausal en lógica de primer orden

## Resumen — pasaje a forma clausal en lógica de primer orden

1. Reescribir  $\Rightarrow$  usando  $\neg$  y  $\vee$ .
2. Pasar a f.n. negada, empujando  $\neg$  hacia adentro.
3. Pasar a f.n. prenexa, extrayendo  $\forall, \exists$  hacia afuera.
4. Pasar a f.n. de Skolem, Skolemizando los existenciales.
5. Pasar a f.n. conjuntiva, distribuyendo  $\vee$  sobre  $\wedge$ .
6. Empujar los cuantificadores hacia adentro de las conjunciones.

Cada paso produce una fórmula equivalente,  
excepto la Skolemización que sólo preserva satisfactibilidad.



# Pasaje a forma clausal en lógica de primer orden

## Ejemplo — pasaje a forma clausal

Queremos ver que  $\sigma \equiv \exists X. (\forall Y. \mathbf{P}(X, Y) \Rightarrow \forall Y. \mathbf{P}(Y, X))$  es válida.

Primero la negamos:  $\neg\sigma \equiv \neg\exists X. (\forall Y. \mathbf{P}(X, Y) \Rightarrow \forall Y. \mathbf{P}(Y, X))$ .

Pasamos  $\neg\sigma$  a forma clausal:

$$\begin{aligned} & \neg\exists X. (\forall Y. \mathbf{P}(X, Y) \Rightarrow \forall Y. \mathbf{P}(Y, X)) \\ \rightarrow & \neg\exists X. (\neg\forall Y. \mathbf{P}(X, Y) \vee \forall Y. \mathbf{P}(Y, X)) \\ \rightarrow & \forall X. \neg(\neg\forall Y. \mathbf{P}(X, Y) \vee \forall Y. \mathbf{P}(Y, X)) \\ \rightarrow & \forall X. (\neg\neg\forall Y. \mathbf{P}(X, Y) \wedge \neg\forall Y. \mathbf{P}(Y, X)) \\ \rightarrow & \forall X. (\forall Y. \mathbf{P}(X, Y) \wedge \exists Y. \neg\mathbf{P}(Y, X)) \\ \rightarrow & \forall X. \exists Y. (\forall Y. \mathbf{P}(X, Y) \wedge \neg\mathbf{P}(Y, X)) \\ \rightarrow & \forall X. \exists Y. \forall Z. (\mathbf{P}(X, Z) \wedge \neg\mathbf{P}(Y, X)) \\ \rightarrow & \forall X. \forall Z. (\mathbf{P}(X, Z) \wedge \neg\mathbf{P}(f(X), X)) \\ \rightarrow & \forall X. \forall Z. \mathbf{P}(X, Z) \wedge \forall X. \forall Z. \neg\mathbf{P}(f(X), X) \end{aligned}$$

La forma clausal es:

$$\{\{\mathbf{P}(X, Z)\}, \{\neg\mathbf{P}(f(X), X)\}\} \equiv \{\{\mathbf{P}(X, Y)\}, \{\neg\mathbf{P}(f(Z), Z)\}\}$$

# Refutación en lógica de primer orden

Una vez obtenido un conjunto de cláusulas  $\mathcal{C} = \{\kappa_1, \dots, \kappa_n\}$ , se busca una **refutación**, es decir, una demostración de  $\mathcal{C} \vdash \perp$ .

Recordemos la regla de resolución proposicional:

$$\frac{\{\mathbf{P}, \ell_1, \dots, \ell_n\} \quad \{\neg \mathbf{P}, \ell'_1, \dots, \ell'_m\}}{\{\ell_1, \dots, \ell_n, \ell'_1, \dots, \ell'_m\}}$$

Queremos adaptarla a lógica de primer orden.

En lugar de una variable proposicional **P** vamos a tener una fórmula atómica **P**( $t_1, \dots, t_n$ ).

¿Podemos escribir la regla así?:

$$\frac{\{\mathbf{P}(t_1, \dots, t_n), \ell_1, \dots, \ell_n\} \quad \{\neg \mathbf{P}(t_1, \dots, t_n), \ell'_1, \dots, \ell'_m\}}{\{\ell_1, \dots, \ell_n, \ell'_1, \dots, \ell'_m\}}$$

## Refutación en lógica de primer orden

Consideremos la fórmula:

$$\forall x. P(x) \wedge \neg P(0)$$

Debería ser refutable, pues es insatisfactible.

Su forma clausal consta de dos cláusulas:

$$\{P(x)\} \quad \{\neg P(0)\}$$

La regla de resolución propuesta no aplica pues  $P(x) \neq P(0)$ .

Los términos no necesariamente tienen que ser iguales.

Relajamos la regla para permitir que sean **unificables**.

# Refutación en lógica de primer orden

La regla de resolución de primer orden es:

$$\frac{\begin{array}{c} \{\sigma_1, \dots, \sigma_p, \ell_1, \dots, \ell_n\} \quad \{\neg\tau_1, \dots, \neg\tau_q, \ell'_1, \dots, \ell'_m\} \\ \mathbf{S} = \text{mgu}(\{\sigma_1 \stackrel{?}{=} \sigma_2 \stackrel{?}{=} \dots \stackrel{?}{=} \sigma_p \stackrel{?}{=} \tau_1 \stackrel{?}{=} \tau_2 \stackrel{?}{=} \dots \stackrel{?}{=} \tau_q\}) \end{array}}{\mathbf{S}(\{\ell_1, \dots, \ell_n, \ell'_1, \dots, \ell'_m\})}$$

con  $p > 0$  y  $q > 0$ .

Se asume implícitamente que las cláusulas están renombradas de tal modo que  $\{\sigma_1, \dots, \sigma_p, \ell_1, \dots, \ell_n\}$  y  $\{\neg\tau_1, \dots, \neg\tau_q, \ell'_1, \dots, \ell'_m\}$  no tienen variables en común.

# Refutación en lógica de primer orden

El algoritmo de refutación se adapta sin mayores cambios.  
Se usa la nueva regla de resolución para calcular la resolvente.

# Refutación en lógica de primer orden

## Ejemplo — método de resolución

Queremos demostrar  $\sigma \equiv \exists X. (\forall Y. \mathbf{P}(X, Y) \Rightarrow \forall Y. \mathbf{P}(Y, X))$ .

Equivalentemente, veamos que  $\neg\sigma \vdash \perp$ .

La forma clausal de  $\neg\sigma$  era:

$$\mathcal{C} = \underbrace{\{\{\mathbf{P}(X, Y)\}\}}_{\boxed{1}}, \underbrace{\{\neg\mathbf{P}(f(Z), Z)\}\}}_{\boxed{2}}$$

► De  $\boxed{1}$  y  $\boxed{2}$  calculamos

$\mathbf{mgu}(\mathbf{P}(X, Y) \stackrel{?}{=} \mathbf{P}(f(Z), Z)) = \{X := f(Z), Y := Z\}$   
y se obtiene la resolvente  $\{\}$ .

# Refutación en lógica de primer orden

## Resolución binaria

Considerar la siguiente variante de la regla de resolución:

$$\frac{\{\sigma, \ell_1, \dots, \ell_n\} \quad \{\neg\tau, \ell'_1, \dots, \ell'_m\} \quad \mathbf{S} = \text{mgu}(\{\sigma \stackrel{?}{=} \tau\})}{\mathbf{S}(\{\ell_1, \dots, \ell_n, \ell'_1, \dots, \ell'_m\})}$$

**No es completa.**

## Ejemplo

$\{\{\mathbf{P}(\mathbf{X}), \mathbf{P}(\mathbf{Y})\}, \{\neg\mathbf{P}(\mathbf{Z}), \neg\mathbf{P}(\mathbf{W})\}\}$  es insatisfactible.

Pero no es posible alcanzar la cláusula vacía  $\{\}$  con resolución binaria.

# Corrección del método de resolución de **primer orden**

## Teorema (corrección del pasaje a forma clausal)

Dada una fórmula  $\sigma$ :

1. El pasaje a forma clausal termina.
2. El conjunto de cláusulas  $\mathcal{C}$  obtenido es **equisatisfactible** a  $\sigma$ .  
Es decir,  $\sigma$  es sat. si y sólo si  $\mathcal{C}$  es sat..



# Corrección del método de resolución de **primer orden**

## Teorema (corrección del algoritmo de refutación)

Dado un conjunto de cláusulas  $\mathcal{C}_0$ :

1. Si  $\mathcal{C}_0 \vdash \perp$ , existe una manera de elegir las cláusulas tal que el algoritmo de refutación termina.
2. El algoritmo retorna **INSAT** si y sólo si  $\mathcal{C}_0 \vdash \perp$ .

Si  $\mathcal{C}_0 \not\vdash \perp$ , no hay garantía de terminación.

# Resolución de primer orden

## Ejemplo — no terminación

La siguiente fórmula  $\sigma$  no es válida:

$$\forall X. (\mathbf{P}(\text{succ}(X)) \Rightarrow \mathbf{P}(X)) \Rightarrow \mathbf{P}(0)$$

Tratemos de probar que es válida usando el método de resolución.  
Para ello pasamos  $\neg\sigma$  a forma clausal:

$$\underbrace{\{\neg\mathbf{P}(\text{succ}(X)), \mathbf{P}(X)\}}_{\boxed{1}}, \underbrace{\{\neg\mathbf{P}(0)\}}_{\boxed{2}}$$

- ▶ De  $\boxed{1}$  y  $\boxed{2}$  obtenemos  $\boxed{3} = \{\neg\mathbf{P}(\text{succ}(0))\}$ .
- ▶ De  $\boxed{1}$  y  $\boxed{3}$  obtenemos  $\boxed{4} = \{\neg\mathbf{P}(\text{succ}(\text{succ}(0)))\}$ .
- ▶ De  $\boxed{1}$  y  $\boxed{4}$  obtenemos  $\boxed{5} = \{\neg\mathbf{P}(\text{succ}(\text{succ}(\text{succ}(0))))\}$ .

...

i i i i i i i i i ? ? ? ? ? ? ? ?

### Lectura recomendada

#### **Artículo original de Robinson.**

J. A. Robinson. *A Machine-Oriented Logic Based on the Resolution Principle.*

Journal of the Association for Computing Machinery, Vol. 12,  
No. 1 (enero de 1965), pp. 23-41.