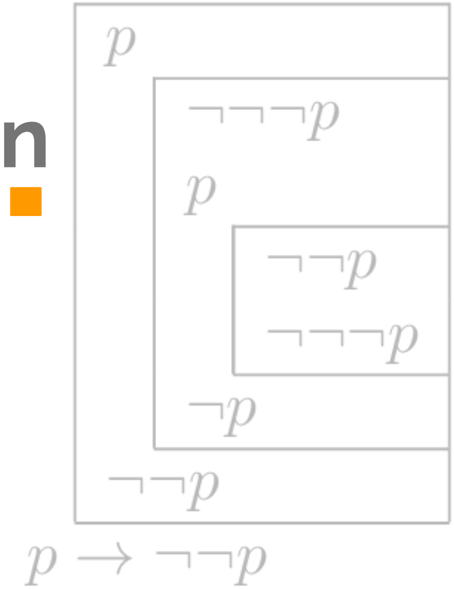
Departamento de Ingeniería Informática
Universidad de Santiago de Chile

Lógica y Teoría de la Computación

Primer semestre 2022

Daniel Vega Araya



Santiago, 2022

LPO - **Método de Resolución**

- ¿Cómo demostramos que $\Sigma \models \varphi$?

LPO - **Método de Resolución**

- ¿Cómo demostramos que $\Sigma \models \varphi$?
- Necesitamos un método que nos ayude a esto.

LPO - **Método de Resolución**

- Necesitamos un método que nos ayude a encontrar, de ser posible, el unificador de máxima generalidad.

- Propuesto en 1965 por J. A. Robinson.

Algoritmo de Robinson

Sean E y F dos términos que queremos unificar. Consideramos inicialmente $\sigma_0 = \{\}$ una sustitución vacía, es decir, que no cambia ninguna variable. Dado que vamos a realizar un proceso iterativo, consideramos inicialmente $E_0 = \sigma_0(E)$ y $F_0 = \sigma_0(F)$. En cada iteración k del algoritmo se realizan los siguientes pasos:

1. Si $E_k = F_k$ entonces las cláusulas E y F son unificables y un unificador de máxima generalidad es $\sigma = \sigma_k \circ \cdots \circ \sigma_0$. Además, el término E_k es el término unificado. En este caso el proceso termina aquí.
2. Si $E_k \neq F_k$ entonces se busca el primer par de discordancia entre E_k y F_k. Sea éste D_k.
3. Si D_k contiene una variable y un término (pueden ser dos variables y una de ellas hace de término) pasamos al siguiente paso. En otro caso los términos no son unificables y terminamos el proceso.

Algoritmo de Robinson

4. Si la variable aparece en el término se produce un ***occur check*** por lo que E y F no unifican y terminamos. Si esto no ocurre pasamos al siguiente paso.
5. Construimos una nueva sustitución que vincule la variable con el término de D_k. Sea esta sustitución σ_{k+1}. Construimos ahora dos nuevos términos $E_{k+1} = \sigma_{k+1}(E_k)$ y $F_{k+1} = \sigma_{k+1}(F_k)$ y volvemos al paso 1.

Este algoritmo siempre termina para dos términos cualesquiera. Si los términos no eran unificables terminará indicándose así y si eran unificables devolverá un unificador de máxima generalidad y el término resultante unificado.

Algoritmo de Robinson

Ejemplo

1. Sean los términos p(a, X) y p(X , Y).

Algoritmo de Robinson

Ejemplo

1. Sean los términos p(a, X) y p(X , Y).

Sean E y F dos términos que queremos unificar. Consideramos inicialmente $\sigma_0 = \{\}$ una sustitución vacía, es decir, que no cambia ninguna variable. Dado que vamos a realizar un proceso iterativo, consideramos inicialmente $E_0 = \sigma_0(E)$ y $F_0 = \sigma_0(F)$. En cada iteración k del algoritmo se realizan los siguientes pasos:

1. Si $E_k = F_k$ entonces las cláusulas E y F son unificables y un unificador de máxima generalidad es $\sigma = \sigma_k \circ \cdots \circ \sigma_0$. Además, el término E_k es el término unificado. En este caso el proceso termina aquí.
2. Si $E_k = F_k$ entonces se busca el primer par de discordancia entre E_k y F_k. Sea éste D_k.
3. Si D_k contiene una variable y un término (pueden ser dos variables y una de ellas hace de término) pasamos al siguiente paso. En otro caso los términos no son unificables y terminamos el proceso.
4. Si la variable aparece en el término se produce un ***occur check*** por lo que E y F no unifican y terminamos. Si esto no ocurre pasamos al siguiente paso.
5. Construimos una nueva sustitución que vincule la variable con el término de D_k. Sea esta sustitución σ_{k+1}. Construimos ahora dos nuevos términos $E_{k+1} = \sigma_{k+1}(E_k)$ y $F_{k+1} = \sigma_{k+1}(F_k)$ y volvemos al paso 1.

LPO - **Método de Resolución**

- Lo que queremos demostrar es:

$$¿\Sigma \models \varphi?$$

- Equivalente a mostrar que:

$$\Sigma \cup \{\neg\varphi\} \text{ es contradictorio (i.e. } \Sigma \models \square \text{)}$$

Nota: decimos que $\square$ es la cláusula vacía porque una **cláusula sin literales** no es satisfacible.

LPO - **Método de Resolución**

- Nos valdremos de la siguiente Regla de Resolución:

$$\frac{\begin{array}{c} p_1 + \ldots + p_j + \ldots + p_m \\ q_1 + \ldots + q_k + \ldots + q_n \end{array}}{(p_1 + \ldots + p_j + \ldots + p_m + q_1 + \ldots + q_k + \ldots + q_n)\boldsymbol{\theta}}$$

Donde p_j y q_k son uno la negación del otro (complementarios) y θ es su unificador

LPO - **Método de Resolución**

Suponga que quiere demostrar que φ es consecuencia lógica de Σ:

- Transforme Σ U {¬φ} a Forma Normal de Skolem.
- Usando la equivalencia $\forall x\,(A(x) * B(x)) \equiv \forall x\, A(x) * \forall x\, B(x)$
  - Transforme las fórmulas a un conjunto de cláusulas $C = \{c_1, ..., c_n\}$ (sin cuantificadores)
  - Mientras □ no pertenezca a C y existen c_j y c_k en C tales que la regla de resolución es aplicable:
    - Aplique la regla de resolución a c_j y c_k generando C’
    - Hacer C = C U {C’}

LPO - **Método de Resolución**

Ejercicio: Demostrar que

$$\models \exists x\, (P(x) \rightarrow \forall y\, P(y))$$

LPO - **Método de Resolución**

Ejercicio: Demostrar que

$$\models \exists x\,(P(x) \rightarrow \forall y\, P(y))$$

$$\Sigma \cup \{\neg\varphi\} = \{\,\} \cup \{\neg \exists x\,(P(x) \rightarrow \forall y\, P(y))\}$$

LPO - **Método de Resolución**

Ejercicio: Demostrar que

$$\models \exists x\,(P(x) \rightarrow \forall y\, P(y))$$

$$\Sigma \cup \{\neg\varphi\} = \{\,\} \cup \{\neg \exists x\,(P(x) \rightarrow \forall y\, P(y))\}$$

Transformamos las fórmulas a un conjunto de cláusulas

$$\neg \exists x\,(P(x) \rightarrow \forall y\, P(y))$$

LPO - **Método de Resolución**

Ejercicio: Demostrar que

$$\models \exists x\,(P(x) \rightarrow \forall y\, P(y))$$

$$\Sigma \cup \{\neg\varphi\} = \{\,\} \cup \{\neg \exists x\,(P(x) \rightarrow \forall y\, P(y))\}$$

Transformamos las fórmulas a un conjunto de cláusulas

$$\neg \exists x\,(P(x) \rightarrow \forall y\, P(y))$$

$$\equiv \neg \exists x\,(\neg P(x) \vee \forall y\, P(y)) \qquad \text{[eliminamos } \rightarrow\text{]}$$

LPO - **Método de Resolución**

Ejercicio: Demostrar que

$$\models \exists x\,(P(x) \rightarrow \forall y\, P(y))$$

$$\Sigma \cup \{\neg\varphi\} = \{\,\} \cup \{\neg \exists x\,(P(x) \rightarrow \forall y\, P(y))\}$$

Transformamos las fórmulas a un conjunto de cláusulas

$$\neg \exists x\,(P(x) \rightarrow \forall y\, P(y))$$

$$\equiv \neg \exists x\,(\neg P(x) \vee \forall y\, P(y))$$ [eliminamos →]

$$\equiv \forall x\, \neg(\neg P(x) \vee \forall y\, P(y))$$ [ingresamos ¬]

LPO - **Método de Resolución**

Ejercicio: Demostrar que

$$\models \exists x\,(P(x) \rightarrow \forall y\, P(y))$$

$$\Sigma \cup \{\neg\varphi\} = \{\,\} \cup \{\neg \exists x\,(P(x) \rightarrow \forall y\, P(y))\}$$

Transformamos las fórmulas a un conjunto de cláusulas

$$\neg \exists x\,(P(x) \rightarrow \forall y\, P(y))$$
$$\equiv \neg \exists x\,(\neg P(x) \vee \forall y\, P(y)) \quad \text{[eliminamos } \rightarrow]$$
$$\equiv \forall x\, \neg(\neg P(x) \vee \forall y\, P(y)) \quad \text{[ingresamos } \neg]$$
$$\equiv \forall x\,(P(x) \wedge \neg \forall y\, P(y)) \quad \text{[ingresamos } \neg]$$

LPO - **Método de Resolución**

Ejercicio: Demostrar que

$$\models \exists x\,(P(x) \rightarrow \forall y\,P(y))$$

$$\Sigma \cup \{\neg\varphi\} = \{\,\} \cup \{\neg \exists x\,(P(x) \rightarrow \forall y\,P(y))\}$$

Transformamos las fórmulas a un conjunto de cláusulas

$$\neg \exists x\,(P(x) \rightarrow \forall y\,P(y))$$
$$\equiv \neg \exists x\,(\neg P(x) \lor \forall y\,P(y)) \qquad \text{[eliminamos} \rightarrow\text{]}$$
$$\equiv \forall x\,\neg(\neg P(x) \lor \forall y\,P(y)) \qquad \text{[ingresamos} \neg\text{]}$$
$$\equiv \forall x\,(P(x) \land \neg\forall y\,P(y)) \qquad \text{[ingresamos} \neg\text{]}$$
$$\equiv \forall x\,(P(x) \land \exists y\,\neg P(y)) \qquad \text{[ingresamos} \neg\text{]}$$

LPO - **Método de Resolución**

Ejercicio: Demostrar que

$$\models \exists x\,(P(x) \rightarrow \forall y\, P(y))$$

$$\Sigma \cup \{\neg\varphi\} = \{\,\} \cup \{\neg \exists x\,(P(x) \rightarrow \forall y\, P(y))\}$$

Transformamos las fórmulas a un conjunto de cláusulas

$$\neg \exists x\,(P(x) \rightarrow \forall y\, P(y))$$
$$\equiv \neg \exists x\,(\neg P(x) \vee \forall y\, P(y)) \quad \text{[eliminamos} \rightarrow\text{]}$$
$$\equiv \forall x\, \neg(\neg P(x) \vee \forall y\, P(y)) \quad \text{[ingresamos} \neg\text{]}$$
$$\equiv \forall x\, (P(x) \wedge \neg \forall y\, P(y)) \quad \text{[ingresamos} \neg\text{]}$$
$$\equiv \forall x\, (P(x) \wedge \exists y\, \neg P(y)) \quad \text{[ingresamos} \neg\text{]}$$
$$\equiv \forall x\, \exists y\, (P(x) \wedge \neg P(y)) \quad \text{[exteriorizar } \exists y\text{]}$$

LPO - **Método de Resolución**

Ejercicio: Demostrar que

$$\models \exists x\,(P(x) \rightarrow \forall y\, P(y))$$

$$\Sigma \cup \{\neg\varphi\} = \{\ \} \cup \{\neg \exists x\,(P(x) \rightarrow \forall y\, P(y))\}$$

Transformamos las fórmulas a un conjunto de cláusulas

$$\begin{aligned}
& \neg \exists x\,(P(x) \rightarrow \forall y\, P(y)) & \\
\equiv\ & \neg \exists x\,(\neg P(x) \vee \forall y\, P(y)) & \text{[eliminamos } \rightarrow\text{]} \\
\equiv\ & \forall x\, \neg(\neg P(x) \vee \forall y\, P(y)) & \text{[ingresamos } \neg\text{]} \\
\equiv\ & \forall x\, (P(x) \wedge \neg \forall y\, P(y)) & \text{[ingresamos } \neg\text{]} \\
\equiv\ & \forall x\, (P(x) \wedge \exists y\, \neg P(y)) & \text{[ingresamos } \neg\text{]} \\
\equiv\ & \forall x\, \exists y\, (P(x) \wedge \neg P(y)) & \text{[exteriorizar } \exists y\text{]} \\
\equiv\ & \forall x\, (P(x) \wedge \neg P(f(x))) & \text{[skolemizar]}
\end{aligned}$$

LPO - **Método de Resolución**

$$C = \{P(x), \neg P(f(x))\}$$

LPO - **Método de Resolución**

$$C = \{P(x), \neg P(f(x))\}$$

es conveniente renombrar las variables

LPO - **Método de Resolución**

$$C = \{P(x), \neg P(f(x))\}$$

es conveniente renombrar las variables… **para encontrar unificaciones**

LPO - **Método de Resolución**

$$C = \{P(x), \neg P(f(x))\}$$

es conveniente renombrar las variables… **para encontrar unificaciones**

$$C = \{P(x), \neg P(f(\mathbf{y}))\}$$

LPO - **Método de Resolución**

$$C = \{P(x), \neg P(f(x))\}$$

es conveniente renombrar las variables… **para encontrar unificaciones**

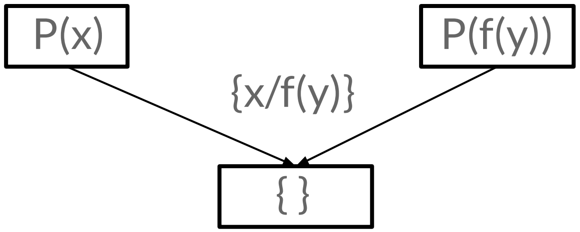
$$C = \{P(x), \neg P(f(\mathbf{y}))\}$$

una aplicación de la regla de resolución permite obtener:

$$\frac{\begin{array}{c} P(x)\ \mathbf{\{x/f(y)\}} \\ \neg P(f(y)) \end{array}}{\square}$$

LPO - **Método de Resolución**

Otra forma de representar la aplicación de la regla de resolución es mostrarlo como un **Grafo Acíclico Dirigido** (GAD)

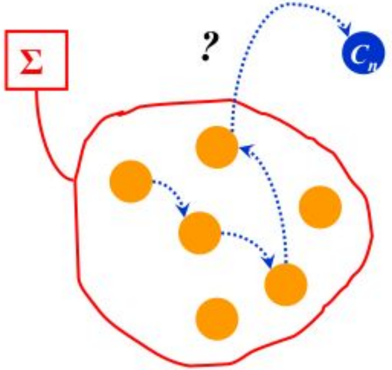


LPO - **Un sistema completo**

El sistema basado en resolución sólo nos permite usar cláusulas.

Sólo podemos usar resolución para demostrar que ¬C es inconsistente.

LPO - **Método de Resolución**



- Dados:
  - un conjunto de cláusulas Σ
  - una cláusula C

Una demostración por resolución de C desde Σ es una secuencia de cláusulas $C_1, C_2, \ldots, C_n$ tal que:

- Para cada $i \leq n$:
  - C_i pertenece a Σ o
  - C_i es una tautología o
  - C_i es obtenido por aplicación de regla de resolución a partir de C_j y C_k
- $C_n = C$

$$\Sigma \vdash_{\text{Res.}} C$$

LPO - **Método de Resolución**

- **Teorema**: (Completitud de Resolución) Dado un conjunto de cláusulas $\Sigma \cup \{C\}$

$$\text{si } \Sigma \models C \text{ entonces } \Sigma \vdash_{Res.} C$$

- **Teorema**: (Correctitud de Resolución) Si C se puede deducir desde Σ usando el conjunto de reglas, entonces C es consecuencia lógica de Σ. En símbolos:

$$\text{si } \Sigma \vdash_{Res.} C \text{ entonces } \Sigma \models C$$

LPO - **Método de Resolución**

- **Teorema**: (Completitud de Resolución) Dado un conjunto de cláusulas $\Sigma \cup \{C\}$

$$\text{si } \Sigma \models C \text{ entonces } \Sigma \vdash_{\text{Res.}} C$$

- **Teorema**: (Correctitud de Resolución) Si C se puede deducir desde Σ usando el conjunto de reglas, entonces C es consecuencia lógica de Σ. En símbolos:

$$\text{si } \Sigma \vdash_{\text{Res.}} C \text{ entonces } \Sigma \models C$$

Finalmente:

$$\boldsymbol{\Sigma \vdash_{\text{Res.}} C \iff \Sigma \models C}$$

LPO - **Un sistema completo**

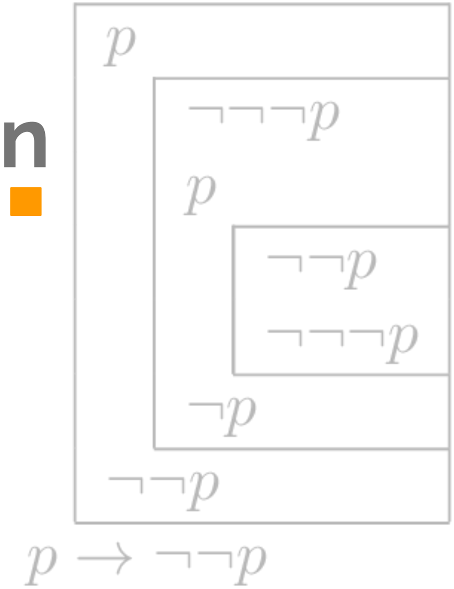
El sistema basado en resolución sólo nos permite usar cláusulas.

¿Cómo generar un sistema completo para cualquier tipo de fórmula?

Departamento de Ingeniería Informática
Universidad de Santiago de Chile

Lógica y Teoría de la Computación

Primer semestre 2022



Santiago, 2022