Paradigmas de Programación

Resolución lógica

1er cuatrimestre de 2024
Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

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).
>> true.
?- padre(zeus, cronos).
>> false.
?- abuelo(X, atenea).
>> X = cronos.
?- abuelo(X, zeus).
>> false.
```

```
?- abuelo(cronos, X).
>> X = atenea ;
>> X = hefesto ;
>> X = ares.
?- abuelo(X, Y).
>> X = cronos, Y = atenea ;
>> X = cronos, Y = hefesto ;
>> 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 $\texttt{pred}(\texttt{t}_1, \ldots, \texttt{t}_n)$:

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

Introducción a Prolog

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

$$\sigma \;\; \texttt{:-} \;\; \tau_1 \texttt{,} \ldots \texttt{,} \tau_n \texttt{.}$$

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

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

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

σ.   Ej.: `padre(zeus, ares).`

Las reglas tienen la siguiente interpretación lógica:

$$\forall \texttt{X}_1 \ldots \forall \texttt{X}_k.((\tau_1 \wedge \ldots \wedge \tau_n) \Rightarrow \sigma)$$

donde $\texttt{X}_1, \ldots, \texttt{X}_k$ son todas las variables libres de las fórmulas.

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

Introducción a Prolog

Una **consulta** es de la forma:

```
?- σ1 , ... , σn
```

Ej.: `?- abuelo(X, ares).`

Las consultas tienen la siguiente interpretación lógica:

$$\exists \mathtt{X}_1 \dots \exists \mathtt{X}_k.(\sigma_1 \wedge \dots \wedge \sigma_n)$$

donde $\mathtt{X}_1, \dots, \mathtt{X}_k$ son todas las variables libres de las fórmulas.

El entorno de Prolog busca demostrar la fórmula τ de la consulta.
En realidad busca *refutar* $\neg\tau$.
La búsqueda de la refutación se basa en el **método de resolución**.

Resolución para lógica proposicional

Entrada:	una fórmula σ de la lógica proposicional.
Salida:	un booleano que indica si σ 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 $\mathcal{C} \vdash \bot$.

Si se encuentra una refutación de $\mathcal{C}$:
Vale $\neg\sigma \vdash \bot$. Es decir, $\neg\sigma$ es insatisfactible/contradicción.
Luego vale $\vdash \sigma$. Es decir, σ es válida/tautología.
Si no se encuentra una refutación de $\mathcal{C}$:
No vale $\neg\sigma \vdash \bot$. Es decir, σ es satisfactible.
Luego no vale $\vdash \sigma$. Es decir, σ no es válida.

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:

$$\begin{array}{lcl} \neg(\sigma \wedge \tau) & \longrightarrow & \neg\sigma \vee \neg\tau \\ \neg(\sigma \vee \tau) & \longrightarrow & \neg\sigma \wedge \neg\tau \\ \neg\neg\sigma & \longrightarrow & \sigma \end{array}$$

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

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

Pasaje a forma clausal

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

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

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	σ_{cnf}	$::=$	$(\kappa_1 \wedge \kappa_2 \wedge \dots \wedge \kappa_n)$
Cláusulas	κ	$::=$	$(\ell_1 \vee \ell_2 \vee \dots \vee \ell_m)$
Literales	ℓ	$::=$	$\mathbf{P} \mid \neg\mathbf{P}$

Pasaje a forma clausal

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

asociativa	$\sigma \vee (\tau \vee \rho) \iff (\sigma \vee \tau) \vee \rho$
conmutativa	$\sigma \vee \tau \iff \tau \vee \sigma$
idempotente	$\sigma \vee \sigma \iff \sigma$

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

$$(\ell_1 \vee \ell_2 \vee \ldots \vee \ell_n) \quad \text{se nota} \quad \{\ell_1, \ell_2, \ldots, \ell_n\}$$

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

$$(\kappa_1 \wedge \kappa_2 \wedge \ldots \wedge \kappa_n) \quad \text{se nota} \quad \{\kappa_1, \kappa_2, \ldots, \kappa_n\}$$

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:

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

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 $\mathcal{C} = \{\kappa_1, \ldots, \kappa_n\}$,
se busca una **refutación**, es decir, una demostración de $\mathcal{C} \vdash \bot$.

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

Regla de resolución

$$\frac{\mathbf{P} \vee \ell_1 \vee \ldots \vee \ell_n \qquad \neg\mathbf{P} \vee \ell'_1 \vee \ldots \vee \ell'_m}{\ell_1 \vee \ldots \vee \ell_n \vee \ell'_1 \vee \ldots \vee \ell'_m}$$

Escrita con notación de cláusulas:

$$\frac{\{\mathbf{P}, \ell_1, \ldots, \ell_n\} \qquad \{\neg\mathbf{P}, \ell'_1, \ldots, \ell'_m\}}{\{\ell_1, \ldots, \ell_n, \ell'_1, \ldots, \ell'_m\}}$$

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

Refutación

Entrada: un conjunto de cláusulas $\mathcal{C}_0 = \{\kappa_1, \ldots, \kappa_n\}$.
Salida: `SAT`/`INSAT` indicando si $\mathcal{C}_0$ es insatisfactible ($\mathcal{C}_0 \vdash \bot$).

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:

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

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

   La resolvente $\rho = \{\ell_1, \ldots, \ell_n, \ell'_1, \ldots, \ell'_m\}$ 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 $\neg\sigma \vdash \bot$.
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 $\mathcal{C} \vdash \bot$.
  Luego $\neg\sigma \vdash \bot$.
  Luego $\vdash \sigma$.

Corrección del método de resolución proposicional

Teorema (corrección del pasaje a forma clausal)

Dada una fórmula σ:

1. El pasaje a forma clausal termina.
2. El conjunto de cláusulas $\mathcal{C}$ obtenido es equivalente a σ.
   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 $\mathcal{C}_0 \vdash \bot$.

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 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 ρ 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.

Resolución para lógica de primer orden

Entrada: una fórmula σ de la lógica de primer orden.
Salida: un booleano indicando si σ es válida.

Si σ es válida, el método siempre termina.

Si σ 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 "$\Rightarrow$":

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

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

Paso 2. Empujar el conectivo "$\neg$" hacia adentro:

$$\begin{array}{lcl}
\neg(\sigma \wedge \tau) & \longrightarrow & \neg\sigma \vee \neg\tau \\
\neg(\sigma \vee \tau) & \longrightarrow & \neg\sigma \wedge \neg\tau \\
\neg\neg\sigma & \longrightarrow & \sigma \\
\neg\forall \mathtt{X}.\, \sigma & \longrightarrow & \exists \mathtt{X}.\, \neg\sigma \\
\neg\exists \mathtt{X}.\, \sigma & \longrightarrow & \forall \mathtt{X}.\, \neg\sigma
\end{array}$$

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

$$\begin{array}{rcl}
\sigma_{\mathrm{nnf}} & ::= & \mathbf{P}(t_1, \dots t_n) \mid \neg\mathbf{P}(t_1, \dots t_n) \mid \sigma_{\mathrm{nnf}} \wedge \sigma_{\mathrm{nnf}} \mid \sigma_{\mathrm{nnf}} \vee \sigma_{\mathrm{nnf}} \\
 & \mid & \forall \mathtt{X}.\, \sigma_{\mathrm{nnf}} \mid \exists \mathtt{X}.\, \sigma_{\mathrm{nnf}}
\end{array}$$

Pasaje a forma clausal en lógica de primer orden

Paso 3. Extraer los cuantificadores ("$\forall/\exists$") hacia afuera.
Se asume siempre $\mathtt{X} \notin \mathsf{fv}(\tau)$:

$$\begin{array}{lcl@{\qquad}lcl}
(\forall \mathtt{X}.\, \sigma) \wedge \tau & \longrightarrow & \forall \mathtt{X}.\, (\sigma \wedge \tau) & \tau \wedge (\forall \mathtt{X}.\, \sigma) & \longrightarrow & \forall \mathtt{X}.\, (\tau \wedge \sigma) \\
(\forall \mathtt{X}.\, \sigma) \vee \tau & \longrightarrow & \forall \mathtt{X}.\, (\sigma \vee \tau) & \tau \vee (\forall \mathtt{X}.\, \sigma) & \longrightarrow & \forall \mathtt{X}.\, (\tau \vee \sigma) \\
(\exists \mathtt{X}.\, \sigma) \wedge \tau & \longrightarrow & \exists \mathtt{X}.\, (\sigma \wedge \tau) & \tau \wedge (\exists \mathtt{X}.\, \sigma) & \longrightarrow & \exists \mathtt{X}.\, (\tau \wedge \sigma) \\
(\exists \mathtt{X}.\, \sigma) \vee \tau & \longrightarrow & \exists \mathtt{X}.\, (\sigma \vee \tau) & \tau \vee (\exists \mathtt{X}.\, \sigma) & \longrightarrow & \exists \mathtt{X}.\, (\tau \vee \sigma)
\end{array}$$

Todas las reglas transforman la fórmula en otra equivalente.

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

$$\sigma_{\mathrm{pre}} ::= \mathcal{Q}_1 \mathtt{X}_1.\, \mathcal{Q}_2 \mathtt{X}_2.\, \dots \mathcal{Q}_n \mathtt{X}_n.\, \tau$$

donde cada $\mathcal{Q}_i$ es un cuantificador $\{\forall, \exists\}$
y τ 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

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

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 $x_1, \ldots, x_n$ 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 $B\{X := f_X(y_1, \ldots, y_m)\}$ donde:
  - $fv(B) = \{x, y_1, \ldots, y_m\}$.
  - 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

$$\forall x.\Big(P(a) \vee \exists y.(Q(y) \wedge \forall z.(P(y, z) \vee \exists u.Q(x, u)))\Big) \vee \exists w.Q(w)$$

La forma normal de Skolem es:

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

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.

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

Ejemplo — la Skolemización no preserva la validez

$$\underbrace{\exists \mathtt{X}.\,(\mathbf{P}(0) \Rightarrow \mathbf{P}(\mathtt{X}))}_{\text{válida}} \qquad \underbrace{\mathbf{P}(0) \Rightarrow \mathbf{P}(\mathtt{c})}_{\text{inválida}}$$

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

$$\forall \mathtt{X}_1.\ \ldots \forall \mathtt{X}_n.\, \exists \mathtt{Y}.\, \sigma \quad \longrightarrow \quad \forall \mathtt{X}_1.\ \ldots \forall \mathtt{X}_n.\, \sigma\{\mathtt{Y} := \mathtt{f}(\mathtt{X}_1, \ldots, \mathtt{X}_n)\}$$

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

Forma normal de Skolem:

$$\sigma_{\mathrm{Sk}} ::= \forall \mathtt{X}_1 \mathtt{X}_2 \ldots \mathtt{X}_n.\, \tau$$

donde τ representa una fórmula en NNF libre de cuantificadores.

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

$$\forall \mathtt{X}_1 \mathtt{X}_2 \ldots \mathtt{X}_n.\, \tau \qquad (\tau \text{ libre de cuantificadores})$$

se pasa τ a forma normal conjuntiva usando las reglas ya vistas:

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

El resultado es una fórmula de la forma:

$$\forall \mathtt{X}_1 \ldots \mathtt{X}_n. \left(\begin{array}{ll} & (\ell_1^{(1)} \vee \ldots \vee \ell_{m_1}^{(1)}) \\ \wedge & (\ell_1^{(2)} \vee \ldots \vee \ell_{m_2}^{(2)}) \\ \ldots & \\ \wedge & (\ell_1^{(k)} \vee \ldots \vee \ell_{m_k}^{(k)}) \end{array} \right)$$

Paso 6. Empujar los cuantificadores universales hacia adentro:

$$\forall \mathtt{X}_1 \ldots \mathtt{X}_n. \left(\begin{array}{ll} & (\ell_1^{(1)} \vee \ldots \vee \ell_{m_1}^{(1)}) \\ \wedge & (\ell_1^{(2)} \vee \ldots \vee \ell_{m_2}^{(2)}) \\ \ldots & \\ \wedge & (\ell_1^{(k)} \vee \ldots \vee \ell_{m_k}^{(k)}) \end{array} \right) \longrightarrow \left(\begin{array}{ll} & \forall \mathtt{X}_1 \ldots \mathtt{X}_n.(\ell_1^{(1)} \vee \ldots \vee \ell_{m_1}^{(1)}) \\ \wedge & \forall \mathtt{X}_1 \ldots \mathtt{X}_n.(\ell_1^{(2)} \vee \ldots \vee \ell_{m_2}^{(2)}) \\ \ldots & \\ \wedge & \forall \mathtt{X}_1 \ldots \mathtt{X}_n.(\ell_1^{(k)} \vee \ldots \vee \ell_{m_k}^{(k)}) \end{array} \right)$$

Por último la **forma clausal** es:

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

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 $\sigma \equiv \exists \mathtt{X}. (\forall \mathtt{Y}. \mathbf{P}(\mathtt{X}, \mathtt{Y}) \Rightarrow \forall \mathtt{Y}. \mathbf{P}(\mathtt{Y}, \mathtt{X}))$ es válida.
Primero la negamos: $\neg\sigma \equiv \neg\exists \mathtt{X}. (\forall \mathtt{Y}. \mathbf{P}(\mathtt{X}, \mathtt{Y}) \Rightarrow \forall \mathtt{Y}. \mathbf{P}(\mathtt{Y}, \mathtt{X}))$.
Pasamos $\neg\sigma$ a forma clausal:

$$
\begin{array}{ll}
 & \neg\exists \mathtt{X}. (\forall \mathtt{Y}. \mathbf{P}(\mathtt{X}, \mathtt{Y}) \Rightarrow \forall \mathtt{Y}. \mathbf{P}(\mathtt{Y}, \mathtt{X})) \\
\to & \neg\exists \mathtt{X}. (\neg\forall \mathtt{Y}. \mathbf{P}(\mathtt{X}, \mathtt{Y}) \vee \forall \mathtt{Y}. \mathbf{P}(\mathtt{Y}, \mathtt{X})) \\
\to & \forall \mathtt{X}. \neg(\neg\forall \mathtt{Y}. \mathbf{P}(\mathtt{X}, \mathtt{Y}) \vee \forall \mathtt{Y}. \mathbf{P}(\mathtt{Y}, \mathtt{X})) \\
\to & \forall \mathtt{X}. (\neg\neg\forall \mathtt{Y}. \mathbf{P}(\mathtt{X}, \mathtt{Y}) \wedge \neg\forall \mathtt{Y}. \mathbf{P}(\mathtt{Y}, \mathtt{X})) \\
\to & \forall \mathtt{X}. (\forall \mathtt{Y}. \mathbf{P}(\mathtt{X}, \mathtt{Y}) \wedge \exists \mathtt{Y}. \neg\mathbf{P}(\mathtt{Y}, \mathtt{X})) \\
\to & \forall \mathtt{X}. \exists \mathtt{Y}. (\forall \mathtt{Y}. \mathbf{P}(\mathtt{X}, \mathtt{Y}) \wedge \neg\mathbf{P}(\mathtt{Y}, \mathtt{X})) \\
\to & \forall \mathtt{X}. \exists \mathtt{Y}. \forall \mathtt{Z}. (\mathbf{P}(\mathtt{X}, \mathtt{Z}) \wedge \neg\mathbf{P}(\mathtt{Y}, \mathtt{X})) \\
\to & \forall \mathtt{X}. \forall \mathtt{Z}. (\mathbf{P}(\mathtt{X}, \mathtt{Z}) \wedge \neg\mathbf{P}(\mathtt{f}(\mathtt{X}), \mathtt{X})) \\
\to & \forall \mathtt{X}. \forall \mathtt{Z}. \mathbf{P}(\mathtt{X}, \mathtt{Z}) \wedge \forall \mathtt{X}. \forall \mathtt{Z}. \neg\mathbf{P}(\mathtt{f}(\mathtt{X}), \mathtt{X})
\end{array}
$$

La forma clausal es:

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

Refutación en lógica de primer orden

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

Recordemos la regla de resolución proposicional:

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

Queremos adaptarla a lógica de primer orden.

En lugar de una variable proposicional $\mathbf{P}$ vamos a tener una fórmula atómica $\mathbf{P}(t_1, \ldots, t_n)$.
¿Podemos escribir la regla así?:

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

Refutación en lógica de primer orden

Consideremos la fórmula:

$$\forall \mathtt{X}. \mathbf{P}(\mathtt{X}) \wedge \neg\mathbf{P}(0)$$

Debería ser refutable, pues es insatisfactible.
Su forma clausal consta de dos cláusulas:

$$\{\mathbf{P}(\mathtt{X})\} \quad \{\neg\mathbf{P}(0)\}$$

La regla de resolución propuesta no aplica pues $\mathbf{P}(\mathtt{X}) \neq \mathbf{P}(0)$.

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:

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

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 $\sigma \equiv \exists \mathtt{X}. (\forall \mathtt{Y}. \mathbf{P}(\mathtt{X},\mathtt{Y}) \Rightarrow \forall \mathtt{Y}. \mathbf{P}(\mathtt{Y},\mathtt{X}))$.
Equivalentemente, veamos que $\neg\sigma \vdash \bot$.
La forma clausal de $\neg\sigma$ era:

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

- De $\boxed{1}$ y $\boxed{2}$ calculamos
  $\mathbf{mgu}(\mathbf{P}(\mathtt{X},\mathtt{Y}) \stackrel{?}{=} \mathbf{P}(\mathtt{f}(\mathtt{Z}),\mathtt{Z})) = \{\mathtt{X} := \mathtt{f}(\mathtt{Z}), \mathtt{Y} := \mathtt{Z}\}$
  y se obtiene la resolvente $\{\,\}$.

Refutación en lógica de primer orden

Resolución binaria

Considerar la siguiente variante de la regla de resolución:

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

No es completa.

Ejemplo

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

1. El pasaje a forma clausal termina.
2. El conjunto de cláusulas $\mathcal{C}$ obtenido es equisatisfactible a σ.
   Es decir, σ 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 $\mathcal{C}_0 \vdash \bot$, existe una manera de elegir las cláusulas tal que el algoritmo de refutación termina.
2. El algoritmo retorna `INSAT` si y sólo si $\mathcal{C}_0 \vdash \bot$.

Si $\mathcal{C}_0 \nvdash \bot$, no hay garantía de terminación.

Resolución de primer orden

Ejemplo — no terminación

La siguiente fórmula σ no es válida:

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

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

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

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

  ...