Teórica 7 : Resolución Lógica

Tomás Felipe Melli

June 8, 2025

Índice

1 Introducción a Prolog

Prolog es un lenguaje de programación lógico y declarativo, diseñado principalmente para la inteligencia artificial y el procesamiento de lenguaje natural. Su nombre proviene de "PROgramming in LOGic".
La forma de operar de **Prolog** es con **términos de primer orden**, en este lenguaje, sería cualquier constante, variable o estructura que representa datos u objetos, y se pueden combinar para formar afirmaciones lógicas.

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

Las **fórmulas atómicas** son de la forma $pred(t_1, \ldots, t_n)$ como por ejemplo

$padre(zeus, atenea)$ `esto es, zeus es el padre de atenea`

En prolog, lo anterior se llama **hecho (fact)** y por tanto, podemos **consultar (query)**

$?- \; padre(zeus, atenea)$

$>> \; true$

Otra manera es definir relaciones lógicas con **reglas (rules)** como por ejemplo, cuándo alguien es abuelo de otro

$$\underbrace{abuelo(X,Y)}_{cabeza} \; :- \; \underbrace{padre(X,Z), \; padre(Z,Y)}_{cuerpo}$$

Formalmente decimos que un progrma es un conjunto de **reglas** donde

- Cada **regla** tiene la forma

  $\sigma :- \tau_1, \ldots, \tau_n$

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

  La interpretación lógica de las reglas es

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

  `donde` $\forall X_1 \ldots \forall X_k$ `son todas las variables libres de las fórmulas`

- Las reglas en las que $\mathbf{n = 0}$ se llaman **hechos** y se escriben

  $\sigma.$

- Las **consultas** tienen la forma

  $?- \; \sigma_1, \ldots, \sigma_n$

  Y su interpretación lógica es

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

  `donde` $\exists X_1 \ldots \exists X_k$ `son todas las variables libres de las fórmulas`

El entorno de Prolog **busca demostrar la fórmula τ de la consulta** para ello, **busca refutar** $\neg\tau$ para demostrar que $\neg\tau \Rightarrow \bot$. Esta búsqueda se basa en el **método de resolución**

2 Resolución para lógica proposicional

Consideramos

`entrada : una fórmula` σ `de la lógica proposicional`

`salida : un booleano que indica si` σ `es válida`

El **método de resolución** consta de dos partes

1. Escribir $\neg\sigma$ como un conjunto $\mathcal{C}$ de **cláusula** `(Pasaje a forma clausal)`
2. Buscar una **refutación** de $\mathcal{C}$,o sea, una derivación de forma que $\mathcal{C} \vdash \bot$

Este método desenlaza en :

- **Si encuentra una refutación de $\mathcal{C}$**
  Vale $\neg\sigma \vdash \bot$. Esto quiere decir que $\neg\sigma$ es insatisfactible y como consecuencia podemos afirmar que $\vdash \sigma$ es válida.
- **Si no encuentra una refutación de $\mathcal{C}$**
  No vale $\neg\sigma \vdash \bot$. Esto quiere decir que $\neg\sigma$ es satisfactible y como consecuencia decimos que $\vdash \sigma$ no es válida.

2.1 Pasaje a Forma Clausal

Pasar a forma clausal (**forma normal conjuntiva** en lógica proposicional, o **forma clausal** en lógica de primer orden) es el proceso de transformación de una fórmula lógica a un formato que nos va a permitir aplicar el método de resolución. La idea es expresar una fórmula lógica como una conjunción de cláusulas, y cada cláusula es una disyunción de **literales** (mínima unidad de una fórmula lógica).

1. Paso I : Deshacerse del conectivo $\implies$. Para ello vamos aplicar la **eliminación de la implicación**

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

   Esto nos permite tener una fórmula resultante sólo con los coectivos $\{\neg, \vee, \wedge\}$

2. Paso II : "distribuir" el conectivo $\neg$

   $$\begin{array}{rcll} \neg(\sigma \vee \tau) & \equiv & \neg\sigma \vee \neg\tau & \texttt{DeMorgan} \\ \neg(\sigma \wedge \tau) & \equiv & \neg\sigma \wedge \neg\tau & \texttt{DeMorgan} \\ \neg\neg\sigma & \equiv & \sigma & \texttt{Eliminación de la doble negación} \end{array}$$

   Luego de este paso tendríamos que tener la fórmula en su **forma normal negada (NNF)**, formalmente :

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

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

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

   Luego de este paso tendríamos que tener la fórmula en su **forma normal conjuntiva (CNF)**, formalmente :

   $$\begin{array}{ll} \texttt{Fórmulas en CNF} & \sigma_{cnf} ::= (\mathrm{k}_1 \wedge \mathrm{k}_2 \wedge \ldots \wedge \mathrm{k}_n) \\ \texttt{Cláusulas} & \mathrm{k} ::= (\mathrm{l}_1 \vee \mathrm{l}_2 \vee \ldots \vee \mathrm{l}_m) \\ \texttt{Literales} & \mathrm{l} ::= \mathbf{P} \mid \neg\mathbf{P} \end{array}$$

Recordemos 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** $\sigma \vee \sigma \iff \sigma$

Con esto en mente, la manera de notar la disyunción de literales (cláusula) como conjunto es :

$$(l_1 \vee l_2 \vee \ldots \vee l_n) \quad \texttt{se representa como} \quad \{l_1, l_2, \ldots, l_n\}$$

Análogamente, como la conjunción también cumple las tres propiedades anteriores, notamos el conjunto de conjunción de cláusulas como

$$(k_1 \wedge k_2 \wedge \ldots \wedge k_n) \quad \texttt{se representa como} \quad \{k_1, k_2, \ldots, k_n\}$$

Ejemplo

$\sigma \equiv (((P \Rightarrow (Q \wedge R)) \wedge P) \Rightarrow Q)$ para demostrar σ la negamos y la pasamos a forma clausal.

$$\begin{array}{l} \neg(((P \Rightarrow (Q \wedge R)) \wedge P) \Rightarrow Q) \\ \overset{\Rightarrow}{\equiv} \neg(\neg((\neg P \vee (Q \wedge R)) \wedge P) \vee Q) \\ \overset{\neg}{\equiv} (\neg\neg((\neg P \vee (Q \wedge R)) \wedge P) \wedge \neg Q) \\ \overset{\neg\neg}{\equiv} (((\neg P \vee (Q \wedge R)) \wedge P) \wedge \neg Q) \\ \overset{\vee}{\equiv} (((\neg P \vee Q) \wedge (\neg P \vee R)) \wedge P) \wedge \neg Q) \\ \overset{\wedge}{\equiv} (\neg P \vee Q) \wedge (\neg P \vee R) \wedge P \wedge \neg Q \end{array}$$

Que en su forma clausal es :

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

2.2 Refutación

Una vez que tenemos el **conjunto de cláusulas** $\mathcal{C} = \{\mathrm{k}_1, \ldots, \mathrm{k}_n\}$ buscamos una **refutación**, es decir, queremos demostrar que $\mathcal{C} \vdash \perp$ el método de refutación se basa en la siguiente regla de deducción :

2.2.1 Regla de Resolución

$$\frac{\mathbf{P} \vee \ l_1 \vee \ldots \vee l_n \qquad \mathbf{P} \ \vee \ l'_1 \vee \ldots \vee l'_m}{l_1 \vee \ldots \vee l_n \vee l'_1 \vee \ldots \vee l'_m}$$

Que si la escribimos con notación de cláusulas :

$$\frac{\{\mathbf{P}, l_1, \ldots, l_n\} \qquad \{\mathbf{P}, l'_1, \ldots, l'_m\}}{\{l_1 \vee \ldots \vee l_n, l'_1 \vee \ldots \vee l'_m\}}$$

Es decir, esta regla de inferencia permite eliminar un par de literales complementarios (uno positivo y uno negativo) entre dos cláusulas, y construir una nueva cláusula: **la resolvente**.
Sigamos con el ejemplo(2.1) anterior. Ya tenemos la forma clausal, de la negación de la fórmula, ahora toca ver si podemos obtener una resolvente $\{\}$ para demostrar que vale.

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

Con **1** y **3** obtenemos la resolvente **5** $= \{\}$ que a continuación con **4** nos permite obtener la resolvente $\{\}$. Lo que nos permite concluir que $\mathcal{C} \vdash \perp$, o sea, $\neg\sigma \vdash \perp$, o sea, $\vdash \sigma$.

2.3 Algoritmo de Refutación

`entrada : un conjunto de cláusulas` $\mathcal{C}_0 = \{k_1, \ldots, k_n\}$
`salida : SAT/INSAT que indica si` $\mathcal{C}_0$ `es insatisfactible` $(\mathcal{C}_0 \vdash \perp)$

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

1. **Si** $\{\} \in \mathcal{C}$, devolver INSAT. (Si derivamos la claúsula vacía gol)
2. Elegir **dos cláusulas** $k, k' \in \mathcal{C}$, tales que :

$$\begin{aligned} k =& \{\mathbf{P}, l_1, \ldots, l_n\} \\ k' =& \{\neg\mathbf{P}, l'_1, \ldots, l'_m\} \end{aligned}$$

   Es decir, tenemos dos cláusulas que contienen literales complementarios, y por ello, podemos usar la regla que vimos. Como consecuencia, la resolvente es $\rho = \{l_1, \ldots, l_n, l'_1, \ldots, l'_m\}$. Si $\rho \notin \mathcal{C}$, la agregamos. Si no pudimos encontrar ninguna resolvente nueva, **devolvemos SAT**
3. Tomamos $\mathcal{C} := \mathcal{C} \cup \{\rho\}$ y volvemos a empezar (paso 1)

2.4 Corrección del método de resolución proposicional

2.4.1 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, $\vdash \sigma \iff \mathcal{C}$

2.4.2 Teorema : corrección del algoritmo de refutación

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

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

3 Resolución para lógica de primer orden

```
entrada : una fórmula σ de la lógica de primer orden
 salida : un booleano que indica 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. Escribimos $\neg\sigma$ como un conjunto $\mathcal{C}$ de **claúsulas**.

2. Buscamos una **refutación** de $\mathcal{C}$
   Si existe, encontramos alguna.
   Sino, puede "colgarse".

3.1 Pasaje a forma clausal en LPO

Para pasar una fórmula de la LPO a forma clausal tenemos que seguir los siguientes pasos :

1. Paso I : Deshacernos del conectivo $\implies$ como ya vimos, con la regla de eliminación.

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

   Esto nos permite tener una fórmula resultante sólo con los conectivos $\{\neg, \vee, \wedge, \forall, \exists\}$

2. Paso II : "distribuir" el conectivo $\neg$

$\neg(\sigma \vee \tau)$	$\equiv$	$\neg\sigma \vee \neg\tau$	`DeMorgan`
$\neg(\sigma \wedge \tau)$	$\equiv$	$\neg\sigma \wedge \neg\tau$	`DeMorgan`
$\neg\neg\sigma$	$\equiv$	σ	`Eliminación de la doble negación`
$\neg\forall X.\sigma$	$\equiv$	$\exists X.\neg\sigma$	`DeMorgan`
$\neg\exists X.\sigma$	$\equiv$	$\forall X.\neg\sigma$	`DeMorgan`

   Luego de este paso tendremos al fórmula en su **forma normal negada (NNF)** :

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

3. Paso III : Extracción de los cuantificadores (existenciales y universales) hacia afuera. **Se asume siempre** $X \notin fv(\tau)$

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

   Con esta extracción logramos obtener la fórmula en su **forma normal prenexa** :

   $$\sigma_{pre} ::= \mathcal{Q}_1 X_1.\mathcal{Q}_2 X_2.\dots\mathcal{Q}_n X_n.\ \tau$$

   donde cada $\mathcal{Q}_i$ es un cuantificador $\{\forall, \exists\}$ y τ representa una fórmula en su forma normal negada libre de cuantificadores.

4. Paso IV : Nos deshacemos de los cuantificadores existenciales. Para lograrlo utilizaremos la técnica de **Herbrand** y **Skolem** que llamaremos **Skolemización**. La idea es transformar una fórmula de esta pinta : $\exists x.B$ en una sin el cuantificador existencial de manera que $B(x) \Rightarrow B\{x := f(x_1, \dots, x_n)\}$ donde :

   - Sustituimos todas las ocurrencias libres de una variable en una expresión / fórmula o término, por otra.
   - 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)

- 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. Es decir $\exists X.B$ se reemplaza por $B\{X := c_x\}$

Es mejor skolemizar de afuera hacia adentro ! Este proceso no es determinístico

Por qué skolemizar preserva la satisfactibilidad ?

La skolemización preserva la satisfactibilidad porque sustituye cada cuantificador existencial por un testigo concreto (una constante o función) que representa algún valor válido de x.

Por qué skolemizar no necesariamente preserva la validez ?

Porque al introducir una función específica f, estás limitando los modelos posibles. (Reemplaza un "hay algún" por un "hay un específico (función o constante)", lo cual reduce el conjunto de modelos válidos.) Por ejemplo :

$$\underbrace{\exists X.(P(0) \Rightarrow P(X))}_{válida} \overset{Skolemización}{\longrightarrow} (P(0) \Rightarrow P(c))$$

Acá lo vemos claramente, antes, la fórmula era menos fuerte, pero luego, exige que ese c específico haga cumplir la implicación, no cualquier posible testigo.
Dada una fórmula en forma normal prenexa y **cerrada**(sin variables libres), aplicamos la regla :

$$\forall X_1. \ldots \forall X_n.\exists Y.\sigma \overset{Skolemización}{\longrightarrow} \forall X_1. \ldots \forall X_n.\sigma\{Y := f(X_1, \ldots, X_n)\} \texttt{ donde f es un símbolo de aridad nuevo } n \geq 0$$

Logramos entonces tener la fórmula en su **forma normal de Skolem**

$$\sigma_{Sk} ::= \forall X_1 X_2 \ldots X_n.\tau \texttt{ donde } \tau \texttt{ representa una fórmula en NNF libre de cuantificadores}$$

5. Paso V : Dada una forma norma de Skolem queremos pasarla a la **forma normal conjuntiva**. O sea, sea ψ libre de cuantificadores :

$$\forall X_1 X_2 \ldots X_n.\ \psi$$

con las reglas de **distribución del disyunto sobre el conjunto**, tenemos que obtener la fórmula de esta pinta

$$\forall X_1 \ldots X_n. \begin{pmatrix} (\ell_1^{(1)} \vee \ldots \vee \ell_{m_1}^{(1)}) \\ \wedge\ (\ell_1^{(2)} \vee \ldots \vee \ell_{m_2}^{(2)}) \\ \vdots \\ \wedge\ (\ell_1^{(k)} \vee \ldots \vee \ell_{m_k}^{(k)}) \end{pmatrix}$$

6. Paso VI : Hora de meter dentro los cuantificadores universales

$$\forall X_1 \ldots X_n. \begin{pmatrix} (\ell_1^{(1)} \vee \ldots \vee \ell_{m_1}^{(1)}) \\ \wedge\ (\ell_1^{(2)} \vee \ldots \vee \ell_{m_2}^{(2)}) \\ \vdots \\ \wedge\ (\ell_1^{(k)} \vee \ldots \vee \ell_{m_k}^{(k)}) \end{pmatrix} \quad \Rightarrow \quad \begin{pmatrix} \forall X_1 \ldots X_n.\ (\ell_1^{(1)} \vee \ldots \vee \ell_{m_1}^{(1)}) \\ \wedge\ \forall X_1 \ldots X_n.\ (\ell_1^{(2)} \vee \ldots \vee \ell_{m_2}^{(2)}) \\ \vdots \\ \wedge\ \forall X_1 \ldots X_n.\ (\ell_1^{(k)} \vee \ldots \vee \ell_{m_k}^{(k)}) \end{pmatrix}$$

Con esto, pasamos a la **forma clausal**:

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

Ejemplo LPO

Supongamos que tenemos $\sigma \equiv \exists X.(\forall Y.P(X,Y) \Rightarrow \forall Y.P(Y,X))$, la negamos y entonces ...

$$\neg\exists X.(\forall Y.P(X,Y) \Rightarrow \forall Y.P(Y,X))$$
$$\stackrel{\Rightarrow}{\equiv} \neg\exists X.(\neg\forall Y.P(X,Y) \vee \forall Y.P(Y,X))$$
$$\stackrel{\neg\exists}{\equiv} \forall X.\neg(\neg\forall Y.P(X,Y) \vee \forall Y.P(Y,X))$$
$$\stackrel{\neg\vee}{\equiv} \forall X.(\neg\neg\forall Y.P(X,Y) \wedge \neg\forall Y.P(Y,X))$$
$$\stackrel{\neg\neg}{\equiv} \forall X.(\forall Y.P(X,Y) \wedge \neg\forall Y.P(Y,X))$$
$$\stackrel{\neg\forall}{\equiv} \forall X.(\forall Y.P(X,Y) \wedge \exists Y.\neg P(Y,X))$$
$$\stackrel{\exists\ \texttt{,renombre}}{\equiv} \forall X.\exists Y.\forall Z.(P(X,Z) \wedge \neg P(Y,X))$$
$$\stackrel{\texttt{Skolemización}}{\equiv} \forall X.\forall Z.(P(X,Z) \wedge \neg P(f(X),X))$$
$$\stackrel{\texttt{Distributiva}}{\equiv} \forall X.\forall Z.(P(X,Z) \wedge \forall X.\forall Z.\neg P(f(X),X))$$

Con esto, la forma clausal es :

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

3.2 Refutación en LPO

Una vez que tenemos el conjunto de cláusulas $\mathcal{C} = \{k_1,\dots,k_n\}$, se busca **refutar**, es decir, demostrar que $\mathcal{C} \vdash \bot$. Lo que vamos a hacer es adaptar la regla de la lógica proposicional para la LPO. En vez de la variable proposicional **P** vamos a tener una **fórmula atómica** $\mathbf{P}(t_1,\dots,t_n)$. Tenemos que **relajar la regla** para permitir que los términos no necesariamente sean iguales y **permitir que sean unificables**. Como consecuencia,

3.2.1 Regla de resolución de LPO

$$\frac{S = mgu(\{\sigma_1 \stackrel{?}{=} \sigma_2 \stackrel{?}{=} \dots \sigma_p \stackrel{?}{=} \tau_1 \stackrel{?}{=} \tau_2 \stackrel{?}{=} \dots \stackrel{?}{=} \tau_q\})}{S(\{l_1,\dots,l_n,l'_1,\dots,l'_m\})}$$

con $p > 0$ y $q > 0$. Se asume implícitamente que las cláusulas están renombradas de tal modo que $\{\sigma_1,\dots\sigma_p,\ l_1,\dots,l_n\}$ y $\{\neg\tau_1,\dots,\neg\tau_q,l'_1,\dots,l'_m\}$ no tienen variables en común.
Retomamos el ejemplo en LPO (3.1) donde

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

Y de **1** y **2** calculamos el $mgu(P(X,Y) \stackrel{?}{=} P(f(Z),Z)) = \{X := f(Z), Y := Z\}$ y esto nos permite obtener la resolvente $\{\}$

3.3 Resolución Binaria

El ejemplo anterior nos permite introducir una variante de la regla de resolución, la resolución binaria.

$$\frac{\{\sigma,l_1,\dots,l_n\} \qquad \{\neg\tau,l'_1,\dots,l'_m\} \qquad S = mgu(\{\sigma \stackrel{?}{=} \tau\})}{S(\{l_1,\dots,l_n,l'_1,\dots,l'_m\})}$$

Pero **no es completa** . Por ejemplo si tuviésemos $\{\{P(X),P(Y)\},\{\neg P(Z),\neg P(W)\}\}$ es insatisfactible pero no es posible alcanzar la cláusula vacía con resolución binaria.

3.4 Corrección del método de resolución en LPO

3.4.1 Teorema : Corrección del pasaje a forma clausal

Dada una fórmula σ:

1. El pasaje a forma clausal termina.
2. El conjunto de cláusulas C obtenido es equisatisfactible con σ.

Es decir, σ es satisfacible si y sólo si C es satisfacible.

3.4.2 Teorema : Corrección del algortimo de refutación

Dado un conjunto de cláusulas C_0:

1. Si $C_0 \vdash \bot$, existe una manera de elegir las cláusulas tal que el algoritmo de refutación termina.
2. El algoritmo retorna `INSAT` si y sólo si $C_0 \vdash \bot$.

Si $C_0 \nvdash \bot$, no hay garantía de terminación.

Ejemplo - no terminación

$\forall X. (P(succ(X)) \Rightarrow P(X)) \Rightarrow P(0)$ Cada vez que hallamos una resolvente que no está en $\mathcal{C}$ hacemos una sustitución que la hace crecer ad infinitum.