

OTTER: Resolución de primer orden

Francisco J. Martín Mateos
José A. Alonso Jiménez

Dpto. Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla

Literales, cláusulas y resolución

- Un **literal** es una fórmula atómica o su negación: $P(x, y)$, $\neg Q(y)$
- Una **cláusula** es una disyunción de literales con variables implícitamente cuantificadas universalmente: $\neg P(x) \vee Q(x)$, $P(a) \vee \neg R(z)$, $\neg Q(z)$, $R(b)$
- **Resolución** es una regla de inferencia que genera una **resolvente** como consecuencia lógica de unas **cláusulas padre**

Literales, cláusulas y resolución

- Un **literal** es una fórmula atómica o su negación: $P(x, y)$, $\neg Q(y)$
- Una **cláusula** es una disyunción de literales con variables implícitamente cuantificadas universalmente: $\neg P(x) \vee Q(x)$, $P(a) \vee \neg R(z)$, $\neg Q(z)$, $R(b)$
- **Resolución** es una regla de inferencia que genera una **resolvente** como consecuencia lógica de unas **cláusulas padre**

$$\frac{\neg P(x) \vee Q(x) \quad P(a) \vee \neg R(z)}{Q(a) \vee \neg R(z)} \quad \{x/a\}$$

Literales, cláusulas y resolución

- Un **literal** es una fórmula atómica o su negación: $P(x, y)$, $\neg Q(y)$
- Una **cláusula** es una disyunción de literales con variables implícitamente cuantificadas universalmente: $\neg P(x) \vee Q(x)$, $P(a) \vee \neg R(z)$, $\neg Q(z)$, $R(b)$
- **Resolución** es una regla de inferencia que genera una **resolvente** como consecuencia lógica de unas **cláusulas padre**

$$\frac{\neg P(x) \vee Q(x) \quad P(a) \vee \neg R(z)}{Q(a) \vee \neg R(z)} \quad \{x/a\}$$
$$\frac{Q(a) \vee \neg R(z) \quad \neg Q(z)}{\neg R(z)} \quad \{z/a\}$$

Literales, cláusulas y resolución

- Un **literal** es una fórmula atómica o su negación: $P(x, y)$, $\neg Q(y)$
- Una **cláusula** es una disyunción de literales con variables implícitamente cuantificadas universalmente: $\neg P(x) \vee Q(x)$, $P(a) \vee \neg R(z)$, $\neg Q(z)$, $R(b)$
- **Resolución** es una regla de inferencia que genera una **resolvente** como consecuencia lógica de unas **cláusulas padre**

$$\begin{array}{rcc} \frac{\neg P(x) \vee Q(x) \quad P(a) \vee \neg R(z)}{Q(a) \vee \neg R(z)} & \{x/a\} \\ \frac{Q(a) \vee \neg R(z) \quad \neg Q(z)}{\neg R(z)} & \{z/a\} \\ \frac{\neg R(z) \quad R(b)}{} & \{z/b\} \end{array}$$

Literales, cláusulas y resolución

- Un **literal** es una fórmula atómica o su negación: $P(x, y)$, $\neg Q(y)$
- Una **cláusula** es una disyunción de literales con variables implícitamente cuantificadas universalmente: $\neg P(x) \vee Q(x)$, $P(a) \vee \neg R(z)$, $\neg Q(z)$, $R(b)$
- **Resolución** es una regla de inferencia que genera una **resolvente** como consecuencia lógica de unas **cláusulas padre**

$$\frac{\frac{\neg P(x_1) \vee Q(x_1) \quad P(a) \vee \neg R(z_1)}{Q(a) \vee \neg R(z_1)} \{x_1/a\}}{\frac{Q(a) \vee \neg R(z_1) \quad \neg Q(z_2)}{\neg R(z_1)} \{z_2/a\}} \{z_1/b\}$$

- Renombrado de variables

Transformación a cláusulas

1. Eliminar las equivalencias usando la relación

$$\mathbf{A} \leftrightarrow \mathbf{B} \equiv (\mathbf{A} \rightarrow \mathbf{B}) \wedge (\mathbf{B} \rightarrow \mathbf{A})$$

2. Eliminar las implicaciones usando la equivalencia

$$\mathbf{A} \rightarrow \mathbf{B} \equiv \neg \mathbf{A} \vee \mathbf{B}$$

3. Interiorizar las negaciones usando las equivalencias

$$\neg(\mathbf{A} \wedge \mathbf{B}) \equiv \neg \mathbf{A} \vee \neg \mathbf{B}$$

$$\neg(\mathbf{A} \vee \mathbf{B}) \equiv \neg \mathbf{A} \wedge \neg \mathbf{B}$$

$$\neg \neg \mathbf{A} \equiv \mathbf{A}$$

$$\neg(\forall x \mathbf{A}) \equiv \exists x \neg \mathbf{A}$$

$$\neg(\exists x \mathbf{A}) \equiv \forall x \neg \mathbf{A}$$

4. Renombrar variables cuantificadas

5. Skolemización: Eliminación de cuantificaciones existenciales

$$\begin{aligned}\forall x_1 \dots x_n \exists y F(x_1, \dots, x_n, y) &\equiv \\ &\equiv \forall x_1 \dots x_n F(x_1, \dots, x_n, f_y(x_1, \dots, x_n))\end{aligned}$$

6. Agrupar cuantificaciones al principio

7. Interiorizar las disyunciones usando las propiedades distributivas

$$\begin{aligned}\mathbf{A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)} \\ \mathbf{(A \wedge B) \vee C \equiv (A \vee C) \wedge (B \vee C)}\end{aligned}$$

Clausificación en OTTER

- El final de una sentencia se indica con un punto: .
- `sos`: Conjunto soporte
- `formula_list(sos)`: Inicio del conjunto de fórmulas
- `end_of_list`: Final del conjunto de formulas
- `set(auto2)`: Activa el modo automático

ejemplo-1.in

```
set(auto2) .  
  
formula_list(sos) .  
  all x (p(x) -> q(x)) .  
  all z exists y (r(z) -> p(y)) .  
  exists x r(x) .  
  all z -q(z) .  
end_of_list.
```

Transformación a cláusulas

-----> sos clausifies to:

```
list(sos).  
1 [] -p(x) | q(x).  
2 [] -r(z) | p($f1(z)).  
3 [] r($c1).  
4 [] -q(z).  
end_of_list.
```

- Constantes de Skolem: \$c1, \$c2, ...
- Funciones de Skolem: \$f1, \$f2, ...

- Regla de resolución binaria

$$\frac{\begin{array}{l} \mathbf{L_1} \vee \dots \vee \mathbf{L_i} \vee \mathbf{A} \vee \mathbf{L_{i+1}} \vee \dots \vee \mathbf{L_n} \\ \mathbf{M_1} \vee \dots \vee \mathbf{M_j} \vee \neg \mathbf{B} \vee \mathbf{M_{j+1}} \vee \dots \vee \mathbf{M_k} \end{array}}{\begin{array}{l} \sigma(\mathbf{L_1} \vee \dots \vee \mathbf{L_i} \vee \mathbf{L_{i+1}} \vee \dots \vee \mathbf{L_n} \vee \\ \mathbf{M_1} \vee \dots \vee \mathbf{M_j} \vee \mathbf{M_{j+1}} \vee \dots \vee \mathbf{M_k}) \end{array}}$$

con $\sigma = \text{umg}(\mathbf{A}, \mathbf{B})$ y $\sigma(\mathbf{A}) = \sigma(\mathbf{B})$

- Separación de variables mediante renombrado
 - Unificación
- La regla de resolución binaria por sí sola forma un cálculo correcto y completo.

Procedimiento de unificación de primer orden

- Datos: Conjunto de ecuaciones a unificar ($\mathbf{R}$) y una unificación parcialmente computada (σ)
- Se procesa cada ecuación a unificar de acuerdo con las siguientes reglas
 - Borrado: $\langle \{t \approx t\} \cup \mathbf{R}; \sigma \rangle \Rightarrow_u \langle \mathbf{R}; \sigma \rangle$
 - Ocurrencia: $\langle \{x \approx t\} \cup \mathbf{R}; \sigma \rangle \Rightarrow_u \perp$ si $x \in \mathcal{V}(t)$ y $x \neq t$
 - Eliminación: $\langle \{x \approx t\} \cup \mathbf{R}; \sigma \rangle \Rightarrow_u \langle \theta(\mathbf{R}); \{x \approx t\} \cup \theta(\sigma) \rangle$ si $x \in \mathbf{X}$, $x \notin \mathcal{V}(t)$ y $\theta = \{x/t\}$
 - Descomposición: $\langle \{f(s_1, \dots, s_n) \approx f(t_1, \dots, t_n)\} \cup \mathbf{R}; \sigma \rangle \Rightarrow_u \langle \{s_1 \approx t_1, \dots, s_n \approx t_n\} \cup \mathbf{R}; \sigma \rangle$
 - Fallo: $\langle \{f(s_1, \dots, s_n) \approx g(t_1, \dots, t_m)\} \cup \mathbf{R}; \sigma \rangle \Rightarrow_u \perp$ si $n \neq m$ o $f \neq g$
 - Orientación: $\langle \{t \approx x\} \cup \mathbf{R}; \sigma \rangle \Rightarrow_u \langle \{x \approx t\} \cup \mathbf{R}; \sigma \rangle$ si $x \in \mathbf{X}$ y $t \notin \mathbf{X}$

Procedimiento de unificación de primer orden

- Aplicación de una sustitución $\sigma = \{X_1/t_1, \dots, X_n/t_n\}$ a una expresión

$$\sigma(\{E_1, \dots, E_n\}) = \{\sigma(E_1), \dots, \sigma(E_n)\}$$

$$\sigma(f(s_1, \dots, s_m)) = f(\sigma(s_1), \dots, \sigma(s_m))$$

$$\sigma(X_i) = t_i$$

$$\sigma(X) = X, \text{ si } X \notin \text{dominio}(\sigma)$$

- Composición de sustituciones

$$\sigma = \{X_1/t_1, \dots, X_n/t_n\}$$

$$\theta = \{Y_1/s_1, \dots, Y_m/s_m\}$$

$$\theta(\sigma) = \{X_i/\theta(t_i) : 1 \leq i \leq n, X_i \neq \theta(t_i)\} \cup \\ \{Y_j/s_j : 1 \leq j \leq m, Y_j \notin \text{dominio}(\sigma)\}$$

Ejemplos de unificación de primer orden

- Unificación de $f(X, X)$ y $f(Y, a)$

$$\begin{aligned}\langle \{f(X, X) \approx f(Y, a)\}; \{\} \rangle &\Rightarrow_u \langle \{X \approx Y, X \approx a\}; \{\} \rangle \\ &\Rightarrow_u \langle \{Y \approx a\}; \{X/Y\} \rangle \\ &\Rightarrow_u \langle \{\}; \{X/a, Y/a\} \rangle\end{aligned}$$

- Unificación de $f(a, X)$ y $f(b, Y)$

$$\begin{aligned}\langle \{f(a, X) \approx f(b, Y)\}; \{\} \rangle &\Rightarrow_u \langle \{a \approx b, X \approx Y\}; \{\} \rangle \\ &\Rightarrow_u \perp\end{aligned}$$

- Unificación de $f(X, X)$ y $f(a, b)$

$$\begin{aligned}\langle \{f(X, X) \approx f(a, b)\}; \{\} \rangle &\Rightarrow_u \langle \{X \approx a, X \approx b\}; \{\} \rangle \\ &\Rightarrow_u \langle \{a \approx b\}; \{X/a\} \rangle \\ &\Rightarrow_u \perp\end{aligned}$$

Ejemplos de unificación de primer orden

- Unificación de $f(Y, X)$ y $f(X, g(Y))$

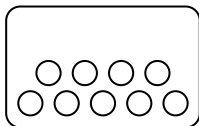
$$\begin{aligned} & \langle \{f(Y, X) \approx f(X, g(Y))\}; \{\} \rangle \\ & \Rightarrow_u \langle \{Y \approx X, X \approx g(Y)\}; \{\} \rangle \\ & \Rightarrow_u \langle \{X \approx g(X)\}; \{Y/X\} \rangle \\ & \Rightarrow_u \perp \end{aligned}$$

- Unificación de $p(U, a, g(U))$ y $p(f(X, Y), X, Z)$

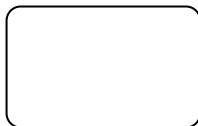
$$\begin{aligned} & \langle \{p(U, a, g(U)) \approx p(f(X, Y), X, Z)\}; \{\} \rangle \\ & \Rightarrow_u \langle \{U \approx f(X, Y), a \approx X, g(U) \approx Z\}; \{\} \rangle \\ & \Rightarrow_u \langle \{a \approx X, g(f(X, Y)) \approx Z\}; \{U/f(X, Y)\} \rangle \\ & \Rightarrow_u \langle \{X \approx a, g(f(X, Y)) \approx Z\}; \{U/f(X, Y)\} \rangle \\ & \Rightarrow_u \langle \{g(f(a, Y)) \approx Z\}; \{X/a, U/f(a, Y)\} \rangle \\ & \Rightarrow_u \langle \{Z \approx g(f(a, Y))\}; \{X/a, U/f(a, Y)\} \rangle \\ & \Rightarrow_u \langle \{\}; \{Z/g(f(a, Y)), X/a, U/f(a, Y)\} \rangle \end{aligned}$$

Refutación por resolución en OTTER

- Generar resolventes hasta que aparezca la cláusula vacía
- Procedimiento **controlado**: Se consideran dos conjuntos de cláusulas, **soporte** y **usables**. Inicialmente todas las cláusulas están en soporte y ninguna en usables. Mientras soporte es no vacío y no se ha encontrado una refutación



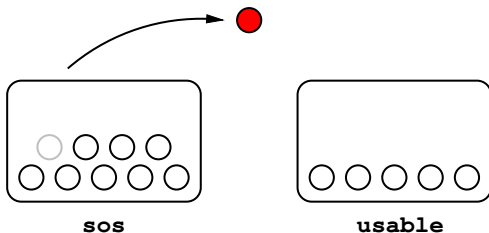
sos



usable

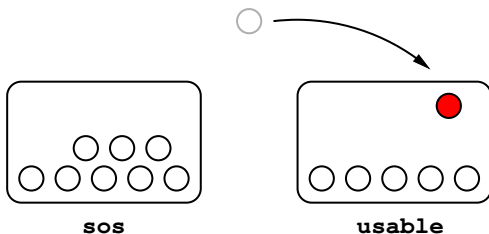
Refutación por resolución en OTTER

- Generar resolventes hasta que aparezca la cláusula vacía
- Procedimiento **controlado**: Se consideran dos conjuntos de cláusulas, **soporte** y **usables**. Inicialmente todas las cláusulas están en soporte y ninguna en usables. Mientras soporte es no vacío y no se ha encontrado una refutación
 - seleccionar una cláusula de soporte;



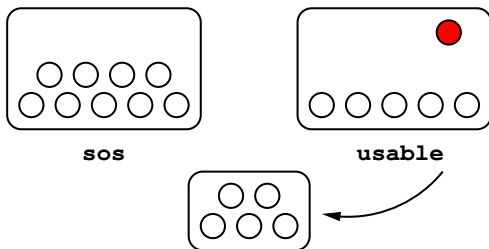
Refutación por resolución en OTTER

- Generar resolventes hasta que aparezca la cláusula vacía
- Procedimiento **controlado**: Se consideran dos conjuntos de cláusulas, **soporte** y **usables**. Inicialmente todas las cláusulas están en soporte y ninguna en usables. Mientras soporte es no vacío y no se ha encontrado una refutación
 - mover dicha cláusula de soporte a usables;



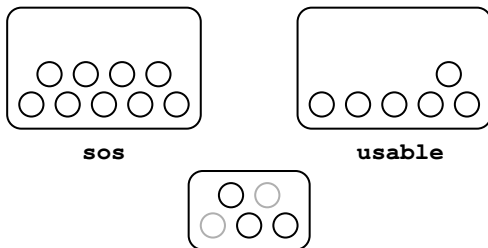
Refutación por resolución en OTTER

- Generar resolventes hasta que aparezca la cláusula vacía
- Procedimiento **controlado**: Se consideran dos conjuntos de cláusulas, **soporte** y **usables**. Inicialmente todas las cláusulas están en soporte y ninguna en usables. Mientras soporte es no vacío y no se ha encontrado una refutación
 - calcular todas las resolventes con usables;



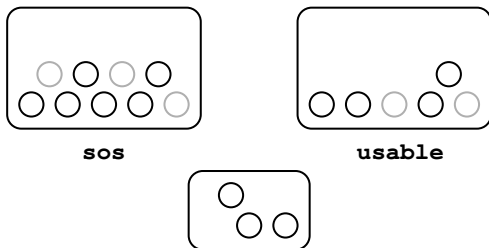
Refutación por resolución en OTTER

- Generar resolventes hasta que aparezca la cláusula vacía
- Procedimiento **controlado**: Se consideran dos conjuntos de cláusulas, **soporte** y **usables**. Inicialmente todas las cláusulas están en soporte y ninguna en usables. Mientras soporte es no vacío y no se ha encontrado una refutación
- simplificar las resolventes e identificar refutaciones;



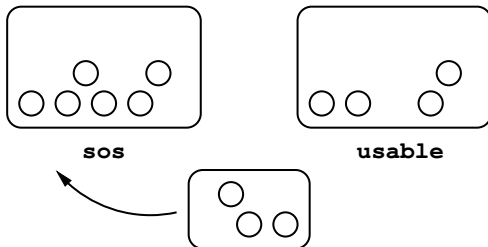
Refutación por resolución en OTTER

- Generar resolventes hasta que aparezca la cláusula vacía
- Procedimiento **controlado**: Se consideran dos conjuntos de cláusulas, **soporte** y **usables**. Inicialmente todas las cláusulas están en soporte y ninguna en usables. Mientras soporte es no vacío y no se ha encontrado una refutación
 - simplificar soporte y usables;



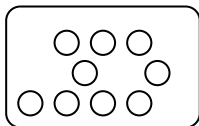
Refutación por resolución en OTTER

- Generar resolventes hasta que aparezca la cláusula vacía
- Procedimiento **controlado**: Se consideran dos conjuntos de cláusulas, **soporte** y **usables**. Inicialmente todas las cláusulas están en soporte y ninguna en usables. Mientras soporte es no vacío y no se ha encontrado una refutación
 - añadir las resolventes simplificadas a soporte;

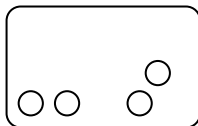


Refutación por resolución en OTTER

- Generar resolventes hasta que aparezca la cláusula vacía
- Procedimiento **controlado**: Se consideran dos conjuntos de cláusulas, **soporte** y **usables**. Inicialmente todas las cláusulas están en soporte y ninguna en usables. Mientras soporte es no vacío y no se ha encontrado una refutación



sos



usable

Refutación por resolución en OTTER

- Generar resolventes hasta que aparezca la cláusula vacía
- Procedimiento **controlado**: Se consideran dos conjuntos de cláusulas, **soporte** y **usables**. Inicialmente todas las cláusulas están en soporte y ninguna en usables. Mientras soporte es no vacío y no se ha encontrado una refutación
 - seleccionar una cláusula de soporte;
 - mover dicha cláusula de soporte a usables;
 - calcular todas las resolventes entre la cláusula seleccionada y las de usables;
 - simplificar las resolventes e identificar refutaciones;
 - simplificar soporte y usables;
 - añadir las resolventes simplificadas a soporte

- `set(binary_res)`: Activa la regla de resolución binaria
- `set(very_verbos)`: Información sobre generación de cláusulas

ejemplo-2.in

```
set(binary_res) .
set(very_verbos) .

formula_list(sos) .
  all x (p(x) -> q(x)) .
  all z exists y (r(z) -> p(y)) .
  exists x r(x) .
  all z -q(z) .
end_of_list.
```

Proceso de búsqueda

```
===== start of search =====  
given clause #1: (wt=2) 3 [] r($c1).  
given clause #2: (wt=2) 4 [] -q(z).  
given clause #3: (wt=4) 1 [] -p(x) | q(x).  
    0 [binary,1.2,4.1] -p(x).  
** KEPT (pick-wt=2): 5 [binary,1.2,4.1] -p(x).  
5 back subsumes 1.  
given clause #4: (wt=2) 5 [binary,1.2,4.1] -p(x).  
given clause #5: (wt=5) 2 [] -r(z) | p($f1(z)).  
    0 [binary,2.1,3.1] p($f1($c1)).  
** KEPT (pick-wt=3): 6 [binary,2.1,3.1] p($f1($c1)).  
----> UNIT CONFLICT at 0.00 sec ----> 7 [binary,6.1,5.1] $F.
```

Prueba encontrada

```
----- PROOF -----  
1 [] -p(x) | q(x) .  
2 [] -r(z) | p($f1(z)) .  
3 [] r($c1) .  
4 [] -q(z) .  
5 [binary,1.2,4.1] -p(x) .  
6 [binary,2.1,3.1] p($f1($c1)) .  
7 [binary,6.1,5.1] $F .  
----- end of proof -----
```

- Explicación de las cláusulas generadas

5 [binary, 1.2, 4.1] $\neg p(x)$

1.2 $\neg p(x) \mid q(x)$

4.1 $\neg q(z)$

Unificador $\sigma = \{x/z\}$

Resolvente $\neg p(z)$

Renombrado $\neg p(x)$

6 [binary, 2.1, 3.1] $p(f1(c1))$

2.1 $\neg r(x) \mid p(f1(c1))$

3.1 $r(c1)$

Unificador $\sigma = \{x/c1\}$

Resolvente $p(f1(c1))$

- Disyunción de literales: $\neg p(x) \mid q(x), \neg r(z) \mid p(a)$
- Cuantificación implícita de variables
 - Variables: Secuencias alfanuméricas que comienzan por **u, v, w, x, y** o **z**
 - Constantes, predicados y funciones: Secuencias alfanuméricas que no comienzan por **u, v, w, x, y** ni **z**

- `list(sos)`: Inicio de un conjunto de cláusulas
- `end_of_list`: Final de un conjunto de cláusulas

ejemplo-3.in

```
set(binary_res) .  
set(very_verbose) .  
  
list(sos) .  
  -p(x) | q(x) .  
  -r(z) | p(f(z)) .  
  r(c) .  
  -q(z) .  
end_of_list .
```

- Demostrar la validez de la siguiente argumentación: *Los caballos son más rápidos que los perros. Algunos galgos son más rápidos que los conejos. Lucero es un caballo y Orejón es un conejo. Por tanto, Lucero es más rápido que Orejón*
- Elementos de la resolución de un problema
 - Representación del conocimiento
 - Explicitación del conocimiento implícito
- Lenguaje del problema

Lucero

Orejon

CABALLO (x)

CONEJO (x)

GALGO (x)

PERRO (x)

MAS_RAPIDO (x, y)

Lucero

Orejón

x es un caballo

x es un conejo

x es un galgo

x es un perro

x es más rápido que y

argumentacion-1a.in

```
set(binary_res) .

formula_list(sos) .
% Los caballos son más rápidos que los perros.
all x y (CABALLO(x) & PERRO(y) -> MAS_RAPIDO(x,y)) .
% Algunos galgos son más rápidos que los conejos
exists x (GALGO(x) &
          (all y (CONEJO(y) -> MAS_RAPIDO(x,y)))) .
% Lucero es un caballo
CABALLO(Lucero) .
% Orejón es un conejo.
CONEJO(Orejon) .
% Lucero no es más rápido que Orejón
-MAS_RAPIDO(Lucero,Orejon) .
end_of_list.
```


- OTTER no es capaz de encontrar una prueba

Respuesta de OTTER

```
===== start of search =====  
given clause #1: (wt=2) 2 [] GALGO($c1).  
given clause #2: (wt=2) 4 [] CABALLO(Lucero).  
given clause #3: (wt=2) 5 [] CONEJO(Orejon).  
given clause #4: (wt=3) 6 [] -MAS_RAPIDO(Lucero,Orejon).  
...  
Search stopped because sos empty.  
===== end of search =====
```

- Información implícita: *Los galgos son perros*

argumentacion-1b.in

```
include("argumentacion-1a.in").  
  
formula_list(sos).  
  % Los galgos son perros  
  all x (GALGO(x) -> PERRO(x)).  
end_of_list.
```

- `include("fichero.in")`: Inclusión de un fichero desde otro
- Varias listas de fórmulas asociadas al conjunto soporte
- ¿Qué dice ahora OTTER?

- Información implícita: *Conejos, perros y caballos son razas distintas*

argumentacion-1c.in

```
include("argumentacion-1b.in").  
  
formula_list(sos).  
  % Los conejos no son perros  
  all x (CONEJO(x) -> -PERRO(x)).  
  % Los conejos no son caballos  
  all x (CONEJO(x) -> -CABALLO(x)).  
  % Los perros no son caballos  
  all x (PERRO(x) -> -CABALLO(x)).  
end_of_list.
```

- Fórmulas simétricas
- ¿Qué dice ahora OTTER?

- Información implícita: *La relación “ser más rápido que” es transitiva*

argumentacion-1d.in

```
include("argumentacion-1c.in").  
  
formula_list(sos).  
% "más rápido que" es transitiva.  
all x y z  
  (MAS_RAPIDO(x,y) & MAS_RAPIDO(y,z) -> MAS_RAPIDO(x,z)).  
end_of_list.
```

Prueba obtenida

```
1 [] -CABALLO(x) | -PERRO(y) | MAS_RAPIDO(x,y) .
2 [] GALGO($c1) .
3 [] -CONEJO(y) | MAS_RAPIDO($c1,y) .
4 [] CABALLO(Lucero) .
5 [] CONEJO(Orejon) .
6 [] -MAS_RAPIDO(Lucero,Orejon) .
7 [] -GALGO(x) | PERRO(x) .
11 [] -MAS_RAPIDO(x,y) | -MAS_RAPIDO(y,z) | MAS_RAPIDO(x,z) .
12 [binary,7.1,2.1] PERRO($c1) .
23 [binary,3.1,5.1] MAS_RAPIDO($c1,Orejon) .
24 [binary,1.1,4.1] -PERRO(x) | MAS_RAPIDO(Lucero,x) .
27 [binary,24.1,12.1] MAS_RAPIDO(Lucero,$c1) .
30 [binary,11.1,27.1]
    -MAS_RAPIDO($c1,x) | MAS_RAPIDO(Lucero,x) .
47 [binary,30.1,23.1] MAS_RAPIDO(Lucero,Orejon) .
48 [binary,47.1,6.1] $F.
```

La estrategia del conjunto soporte

- Procedimiento **controlado**: Se consideran dos conjuntos de cláusulas, **soporte** y **usables**. Inicialmente todas las cláusulas están en soporte y ninguna en usables. Mientras soporte es no vacío y no se ha encontrado una refutación
 - seleccionar una cláusula de soporte;
 - mover dicha cláusula de soporte a usables;
 - calcular todas las resolventes entre la cláusula seleccionada y las de usables;
 - simplificar las resolventes e identificar refutaciones;
 - simplificar soporte y usables;
 - añadir las resolventes simplificadas a soporte
- Estrategia del conjunto soporte: Colocar en usables un conjunto consistente de fórmulas

La estrategia del conjunto soporte

- **usable**: Conjunto usable

argumentacion-2.in

```
set(binary_res) .

formula_list(sos) .
  -MAS_RAPIDO(Lucero,Orejon) .
end_of_list.

formula_list(usable) .
  all x y (CABALLO(x) & PERRO(y) -> MAS_RAPIDO(x,y)) .
  exists x (GALGO(x) &
    (all y (CONEJO(y) -> MAS_RAPIDO(x,y)))) .
  CABALLO(Lucero) .
  CONEJO(Orejon) .
  all x (GALGO(x) -> PERRO(x)) .
  all x (CONEJO(x) -> -PERRO(x)) .
  all x (CONEJO(x) -> -CABALLO(x)) .
  all x (PERRO(x) -> -CABALLO(x)) .
  all x y z
    (MAS_RAPIDO(x,y) & MAS_RAPIDO(y,z) -> MAS_RAPIDO(x,z)) .
end_of_list.
```

Prueba obtenida

```
1 [] -MAS_RAPIDO(Lucero,Oregon) .
2 [] -CABALLO(x) | -PERRO(y) | MAS_RAPIDO(x,y) .
3 [] GALGO($c1) .
4 [] -CONEJO(y) | MAS_RAPIDO($c1,y) .
5 [] CABALLO(Lucero) .
6 [] CONEJO(Oregon) .
7 [] -GALGO(x) | PERRO(x) .
11 [] -MAS_RAPIDO(x,y) | -MAS_RAPIDO(y,z) | MAS_RAPIDO(x,z) .
12 [binary,1.1,11.3] -MAS_RAPIDO(Lucero,x) |
    -MAS_RAPIDO(x,Oregon) .
19 [binary,12.2,4.2] -MAS_RAPIDO(Lucero,$c1) |
    -CONEJO(Oregon) .
22 [binary,19.2,6.1] -MAS_RAPIDO(Lucero,$c1) .
24 [binary,22.1,2.3] -CABALLO(Lucero) | -PERRO($c1) .
25 [binary,24.1,5.1] -PERRO($c1) .
27 [binary,25.1,7.2] -GALGO($c1) .
28 [binary,27.1,3.1] $F.
```


Resolución Unitaria (UR)

- Resolución Unitaria (UR)

$$\mathbf{L}_1 \vee \dots \vee \mathbf{L}_n$$

$$\mathbf{M}_1$$

$$\vdots$$

$$\mathbf{M}_{i-1}$$

$$\mathbf{M}_{i+1}$$

$$\vdots$$

$$\mathbf{M}_n$$

$$\sigma(\mathbf{L}_i)$$

donde $\sigma = \text{umg}(\mathbf{L}_j, \overline{\mathbf{M}_j})$, $j \in \{1, \dots, i-1, i+1, \dots, n\}$

- La resolución unitaria por sí sola no forma un cálculo completo:
 $\{\mathbf{p} \vee \mathbf{q}, \mathbf{p} \vee \neg \mathbf{q}, \neg \mathbf{p} \vee \mathbf{q}, \neg \mathbf{p} \vee \neg \mathbf{q}\}$

Resolución Unitaria (UR)

- `set(ur_res)`: Activa la regla de resolución unitaria

argumentacion-3.in

```
set(ur_res).

formula_list(sos).
  all x y (CABALLO(x) & PERRO(y) -> MAS_RAPIDO(x,y)).
  exists x (GALGO(x) &
    (all y (CONEJO(y) -> MAS_RAPIDO(x,y)))).
  CABALLO(Lucero).
  CONEJO(Orejon).
  all x (GALGO(x) -> PERRO(x)).
  all x (CONEJO(x) -> -PERRO(x)).
  all x (CONEJO(x) -> -CABALLO(x)).
  all x (PERRO(x) -> -CABALLO(x)).
  all x y z
    (MAS_RAPIDO(x,y) & MAS_RAPIDO(y,z) -> MAS_RAPIDO(x,z)).
  -MAS_RAPIDO(Lucero,Orejon).
end_of_list.
```

Prueba obtenida

```
1 [] -CABALLO(x) | -PERRO(y) | MAS_RAPIDO(x,y) .
2 [] GALGO($c1) .
3 [] -CONEJO(y) | MAS_RAPIDO($c1,y) .
4 [] CABALLO(Lucero) .
5 [] CONEJO(Orejon) .
6 [] -GALGO(x) | PERRO(x) .
10 [] -MAS_RAPIDO(x,y) | -MAS_RAPIDO(y,z) | MAS_RAPIDO(x,z) .
11 [] -MAS_RAPIDO(Lucero,Orejon) .
12 [ur,6,2] PERRO($c1) .
21 [ur,3,5] MAS_RAPIDO($c1,Orejon) .
22 [ur,1,4,12] MAS_RAPIDO(Lucero,$c1) .
23 [ur,10,21,11] -MAS_RAPIDO(Lucero,$c1) .
24 [binary,23.1,22.1] $F.
```

Resolución Unitaria (UR)

- Explicación de las cláusulas generadas

22 [ur, 1, 4, 12] MAS_RAPIDO(Lucero, \$c1)

1	$\neg \text{CABALLO}(x) \mid \neg \text{PERRO}(y) \mid \text{MAS_RAPIDO}(x, y)$
4	$\text{CABALLO}(\text{Lucero})$
12	$\text{PERRO}(\$c1)$
Unificador	$\sigma = \{x/\text{Lucero}, y/\$c1\}$
Resolvente	$\text{MAS_RAPIDO}(\text{Lucero}, \$c1)$

23 [ur, 10, 21, 11] $\neg \text{MAS_RAPIDO}(\text{Lucero}, \$c1)$

10	$\neg \text{MAS_RAPIDO}(x, y) \mid \neg \text{MAS_RAPIDO}(y, z) \mid \text{MAS_RAPIDO}(x, z)$
21	$\text{MAS_RAPIDO}(\$c1, \text{Orejon})$
11	$\neg \text{MAS_RAPIDO}(\text{Lucero}, \text{Orejon})$
Unificador	$\sigma = \{x/\text{Lucero}, y/\$c1, z/\text{Orejon}\}$
Resolvente	$\neg \text{MAS_RAPIDO}(\text{Lucero}, \$c1)$

- Dado un conjunto de cláusulas cerradas $\mathbf{S}$ que no contiene la cláusula vacía y una estructura $\mathcal{I}$ cualquiera. El conjunto $\mathbf{S}$ se divide en dos subconjuntos:
 - El conjunto $\mathbf{S}_0$ formado por todas las fórmulas de $\mathbf{S}$ que son falsas en $\mathcal{I}$.
 - El conjunto $\mathbf{S}_1$ formado por todas las fórmulas de $\mathbf{S}$ que son verdaderas en $\mathcal{I}$.
- Sólo podemos generar una contradicción al hacer resolución entre cláusulas de distintos conjuntos
- La regla de resolución semántica por sí sola forma un cálculo correcto y completo

Hiper-resolución

- Resolución semántica: Eliminación de literales negativos resolviendo con cláusulas positivas
- Regla de hiper-resolución positiva

$$\frac{\begin{array}{l} \neg \mathbf{A}_1 \vee \dots \vee \neg \mathbf{A}_n \vee \mathbf{B}_1 \vee \dots \vee \mathbf{B}_m \\ \mathbf{M}_1 \vee \mathbf{M}_{1,1} \vee \dots \vee \mathbf{M}_{1,h_1} \\ \vdots \\ \mathbf{M}_n \vee \mathbf{M}_{n,1} \vee \dots \vee \mathbf{M}_{n,h_n} \end{array}}{\sigma(\mathbf{B}_1 \vee \dots \vee \mathbf{B}_m \vee (\bigvee_{i,j} \mathbf{M}_{i,j}))}$$

donde $\mathbf{A}_1, \dots, \mathbf{A}_n, \mathbf{B}_1, \dots, \mathbf{B}_m, \mathbf{M}_i, \mathbf{M}_{i,j}$ son átomos y
 $\sigma = \text{umg}(\mathbf{A}_i, \mathbf{M}_i), i \in \{1, \dots, n\}$

- Regla de hiper-resolución negativa

- `set(hyper_res)`: Activa la regla de hiper-resolución positiva

argumentacion-4.in

```
set(hyper_res).

formula_list(sos).
  all x y (CABALLO(x) & PERRO(y) -> MAS_RAPIDO(x,y)).
  exists x (GALGO(x) &
            (all y (CONEJO(y) -> MAS_RAPIDO(x,y)))).
  CABALLO(Lucero).
  CONEJO(Orejon).
  all x (GALGO(x) -> PERRO(x)).
  all x (CONEJO(x) -> -PERRO(x)).
  all x (CONEJO(x) -> -CABALLO(x)).
  all x (PERRO(x) -> -CABALLO(x)).
  all x y z
    (MAS_RAPIDO(x,y) & MAS_RAPIDO(y,z) -> MAS_RAPIDO(x,z)).
  -MAS_RAPIDO(Lucero,Orejon).
end_of_list.
```

Prueba obtenida

```
1 [] -CABALLO(x) | -PERRO(y) | MAS_RAPIDO(x,y) .
2 [] GALGO($c1) .
3 [] -CONEJO(y) | MAS_RAPIDO($c1,y) .
4 [] CABALLO(Lucero) .
5 [] CONEJO(Orejon) .
6 [] -GALGO(x) | PERRO(x) .
10 [] -MAS_RAPIDO(x,y) | -MAS_RAPIDO(y,z) | MAS_RAPIDO(x,z) .
11 [] -MAS_RAPIDO(Lucero,Orejon) .
12 [hyper,6,2] PERRO($c1) .
13 [hyper,3,5] MAS_RAPIDO($c1,Orejon) .
14 [hyper,1,4,12] MAS_RAPIDO(Lucero,$c1) .
15 [hyper,10,14,13] MAS_RAPIDO(Lucero,Orejon) .
16 [binary,15.1,11.1] $F.
```


- Explicación de las cláusulas generadas

14 [hyper, 1, 4, 12] MAS_RAPIDO(Lucero, \$c1)

1	$\neg \text{CABALLO}(x) \mid \neg \text{PERRO}(y) \mid \text{MAS_RAPIDO}(x, y)$
4	$\text{CABALLO}(\text{Lucero})$
12	$\text{PERRO}(\$c1)$
Unificador	$\sigma = \{x/\text{Lucero}, y/\$c1\}$
Resolvente	$\text{MAS_RAPIDO}(\text{Lucero}, \$c1)$

15 [hyper, 10, 14, 13] MAS_RAPIDO(Lucero, Orejon)

10	$\neg \text{MAS_RAPIDO}(x, y) \mid \neg \text{MAS_RAPIDO}(y, z) \mid \text{MAS_RAPIDO}(x, z)$
14	$\text{MAS_RAPIDO}(\text{Lucero}, \$c1)$
13	$\text{MAS_RAPIDO}(\$c1, \text{Orejon})$
Unificador	$\sigma = \{x/\text{Lucero}, y/\$c1, z/\text{Orejon}\}$
Resolvente	$\text{MAS_RAPIDO}(\text{Lucero}, \text{Orejon})$

- `set(neg_hyper_res)`: Activa la regla de hiper-resolución negativa

argumentacion-5.in

```
set(neg_hyper_res).  
  
formula_list(sos).  
  all x y (CABALLO(x) & PERRO(y) -> MAS_RAPIDO(x,y)).  
  exists x (GALGO(x) &  
            (all y (CONEJO(y) -> MAS_RAPIDO(x,y)))).  
  CABALLO(Lucero).  
  CONEJO(Orejon).  
  all x (GALGO(x) -> PERRO(x)).  
  all x (CONEJO(x) -> -PERRO(x)).  
  all x (CONEJO(x) -> -CABALLO(x)).  
  all x (PERRO(x) -> -CABALLO(x)).  
  all x y z  
    (MAS_RAPIDO(x,y) & MAS_RAPIDO(y,z) -> MAS_RAPIDO(x,z)).  
  -MAS_RAPIDO(Lucero,Orejon).  
end_of_list.
```

Prueba obtenida

```
1 [] -CABALLO(x) | -PERRO(y) | MAS_RAPIDO(x,y) .
2 [] GALGO($c1) .
3 [] -CONEJO(y) | MAS_RAPIDO($c1,y) .
4 [] CABALLO(Lucero) .
5 [] CONEJO(Orejon) .
6 [] -GALGO(x) | PERRO(x) .
10 [] -MAS_RAPIDO(x,y) | -MAS_RAPIDO(y,z) | MAS_RAPIDO(x,z) .
11 [] -MAS_RAPIDO(Lucero,Orejon) .
19 [neg_hyper,10,11] -MAS_RAPIDO(Lucero,x) |
                    -MAS_RAPIDO(x,Orejon) .
22 [neg_hyper,19,3] -MAS_RAPIDO(Lucero,$c1) |
                    -CONEJO(Orejon) .
24 [neg_hyper,22,5] -MAS_RAPIDO(Lucero,$c1) .
26 [neg_hyper,24,1] -CABALLO(Lucero) | -PERRO($c1) .
27 [neg_hyper,26,6] -CABALLO(Lucero) | -GALGO($c1) .
28 [neg_hyper,27,2] -CABALLO(Lucero) .
29 [binary,28.1,4.1] $F.
```

- Explicación de las cláusulas generadas

19 [neg_hyper, 10, 11] $\neg \text{MAS_RAPIDO}(\text{Lucero}, x) \mid$
 $\neg \text{MAS_RAPIDO}(x, \text{Orejon}) .$

10 $\neg \text{MAS_RAPIDO}(x, y) \mid \neg \text{MAS_RAPIDO}(y, z) \mid$
 $\text{MAS_RAPIDO}(x, z)$

11 $\neg \text{MAS_RAPIDO}(\text{Lucero}, \text{Orejon}) .$

Unificador $\sigma = \{x/\text{Lucero}, z/\text{Orejon}\}$

Resolvente $\neg \text{MAS_RAPIDO}(\text{Lucero}, y) \mid \neg \text{MAS_RAPIDO}(y, \text{Orejon}) .$

Renombrado $\neg \text{MAS_RAPIDO}(\text{Lucero}, x) \mid \neg \text{MAS_RAPIDO}(x, \text{Orejon}) .$

22 [neg_hyper, 19, 3] $\neg \text{MAS_RAPIDO}(\text{Lucero}, \$c1) \mid \neg \text{CONEJO}(\text{Orejon}) .$

19 $\neg \text{MAS_RAPIDO}(\text{Lucero}, x) \mid \neg \text{MAS_RAPIDO}(x, \text{Orejon}) \mid$
3 $\neg \text{CONEJO}(y) \mid \text{MAS_RAPIDO}(\$c1, y)$

Unificador $\sigma = \{x/\$c1, y/\text{Orejon}\}$

Resolvente $\neg \text{MAS_RAPIDO}(\text{Lucero}, \$c1) \mid \neg \text{CONEJO}(\text{Orejon})$

Simplificación de cláusulas en OTTER

- Procedimiento controlado: Se consideran dos conjuntos de cláusulas, soporte y usables. Inicialmente todas las cláusulas están en soporte y ninguna en usables. Mientras soporte es no vacío y no se ha encontrado una refutación
 - seleccionar la cláusula **menos pesada** del soporte;
 - mover dicha cláusula de soporte a usables;
 - calcular todas las resolventes entre la cláusula seleccionada y las de usables;
 - **simplificar** las resolventes e **identificar refutaciones**;
 - **simplificar** soporte y usables;
 - añadir las resolventes simplificadas a soporte

Simplificación de cláusulas en OTTER

- Simplificación de resolventes
 - aplicar a la resolvente **eliminación unitaria**;
 - aplicar a la resolvente **factorización** y **simplificación de factores**;
 - si la resolvente es una **tautología** se descarta;
 - si la resolvente es **subsumida** por alguna cláusula de usable o del soporte (subsumción hacia adelante) se descarta;
- Identificación de refutaciones
 - si la resolvente tiene 0 literales, se ha encontrado una refutación;
 - si la resolvente tiene 1 literal, entonces buscar su complementaria (**conflicto unitario**) en usable y soporte;
- Simplificación de soporte y usables
 - Descartar las cláusulas **subsumidas** por alguna resolvente (subsumción hacia atrás)

Peso de una cláusula

- El peso de una cláusula es la suma de los pesos de sus literales
 - El peso de un literal es el peso del átomo que contiene
 - El peso de un átomo es 1 más el peso de todos sus términos
 - El peso de un término es 1 más el peso de todos sus subtérminos
 - El peso de una constante es 1
 - El peso de una variable es 1

- Ejemplos

Cláusula	Peso
$\{p(x), q(x), \neg r(y)\}$	6
$\{\neg p(f(x), g(f(y)))\}$	6

Pesos de cláusulas

```
given clause #1: (wt=2) 2 [] GALGO($c1).
given clause #2: (wt=2) 4 [] CABALLO(Lucero).
given clause #3: (wt=2) 5 [] CONEJO(Orejon).
given clause #4: (wt=3) 6 [] -MAS_RAPIDO(Lucero,Orejon).
given clause #5: (wt=4) 7 [] -GALGO(x) | PERRO(x).
** KEPT (pick-wt=2): 12 [binary,7.1,2.1] PERRO($c1).
given clause #6: (wt=2) 12 [binary,7.1,2.1] PERRO($c1).
given clause #7: (wt=4) 8 [] -CONEJO(x) | -PERRO(x).
...
given clause #22:
    (wt=7) 1 [] -CABALLO(x) | -PERRO(y) | MAS_RAPIDO(x,y).
** KEPT (pick-wt=5):
    24 [binary,1.1,4.1] -PERRO(x) | MAS_RAPIDO(Lucero,x).
** KEPT (pick-wt=5):
    25 [binary,1.2,12.1] -CABALLO(x) | MAS_RAPIDO(x,$c1).
...
```


- Una cláusula C_1 subsume a otra C_2 si existe una sustitución σ tal que $\sigma(C_1)$ es un subconjunto de C_2
- Ejemplos
 - $\{q(x)\}$ subsume a $\{p(x), q(y)\}$ con la sustitución $\{x/y\}$
 - $\{p(x), q(y)\}$ subsume a $\{p(a), q(g(a)), \neg r(x, b)\}$ con la sustitución $\{x/a, y/g(a)\}$
 - $\{p(a, x), p(y, b)\}$ subsume a $\{p(a, b)\}$ con la sustitución $\{x/b, y/a\}$
 - $\{p(x, y)\}$ subsume a $\{p(a, u), p(w, b)\}$ con la sustitución $\{x/a, y/u\}$ o $\{x/w, y/b\}$
- Se puede prescindir de la cláusula subsumida

Subsumción de cláusulas

```
...
given clause #13: (wt=6) 94 [hyper,11,10] p(5,E) | p(5,A) .
...
given clause #56: (wt=13) 88 [] -palabra(x1,x2,x3,x4,x5,1) |
    -p(2,x2) | -p(5,x5) .
    0 [hyper,88,2,95,94] p(2,R) | p(5,A) .
** KEPT (pick-wt=6): 96 [hyper,88,2,95,94] p(2,R) | p(5,A) .
    0 [hyper,88,1,95,94] p(2,A) | p(5,A) .
** KEPT (pick-wt=6): 97 [hyper,88,1,95,94] p(2,A) | p(5,A) .
given clause #57: (wt=6) 96 [hyper,88,2,95,94] p(2,R) | p(5,A) .
    0 [hyper,96,88,1,94] p(5,A) | p(5,A) .
** KEPT (pick-wt=3): 98 [hyper,96,88,1,94] p(5,A) .
    0 [hyper,96,12,7,95] p(5,A) | p(2,R) .
    Subsumed by 98.
    0 [hyper,96,12,10,94] p(2,R) | p(5,A) .
    Subsumed by 96.
98 back subsumes 97.
98 back subsumes 96.
98 back subsumes 94.
```

Factorización y simplificación de factores

- Dada una cláusula $\mathbf{C}$ con dos literales $\mathbf{L}_1, \mathbf{L}_2 \in \mathbf{C}$ tales que existe una sustitución σ tal que $\sigma(\mathbf{L}_1) = \sigma(\mathbf{L}_2)$, entonces se genera la cláusula $\sigma(\mathbf{C})$ (**factor** de $\mathbf{C}$).
- Ejemplos. Sea $\mathbf{C} = \{\mathbf{p}(\mathbf{x}, \mathbf{y}), \mathbf{p}(\mathbf{a}, \mathbf{z}), \mathbf{p}(\mathbf{x}, \mathbf{b})\}$
 - $\{\mathbf{p}(\mathbf{a}, \mathbf{z}), \mathbf{p}(\mathbf{a}, \mathbf{b})\}$ es un factor de $\mathbf{C}$ que se obtiene combinando los literales $\mathbf{p}(\mathbf{x}, \mathbf{y})$ y $\mathbf{p}(\mathbf{a}, \mathbf{z})$ con la sustitución $\sigma = \{\mathbf{x}/\mathbf{a}, \mathbf{y}/\mathbf{z}\}$
 - $\{\mathbf{p}(\mathbf{x}, \mathbf{b}), \mathbf{p}(\mathbf{a}, \mathbf{z})\}$ es un factor de $\mathbf{C}$ que se obtiene combinando los literales $\mathbf{p}(\mathbf{x}, \mathbf{y})$ y $\mathbf{p}(\mathbf{x}, \mathbf{b})$ con la sustitución $\sigma = \{\mathbf{y}/\mathbf{b}\}$
 - $\{\mathbf{p}(\mathbf{a}, \mathbf{y}), \mathbf{p}(\mathbf{a}, \mathbf{b})\}$ es un factor de $\mathbf{C}$ que se obtiene combinando los literales $\mathbf{p}(\mathbf{a}, \mathbf{z})$ y $\mathbf{p}(\mathbf{x}, \mathbf{b})$ con la sustitución $\sigma = \{\mathbf{x}/\mathbf{a}, \mathbf{z}/\mathbf{b}\}$

Factorización y simplificación de factores

- La cláusula C_1 se reduce a la cláusula C_2 por simplificación de factores si existen dos literales $L_1, L_2 \in C_1$ y una sustitución σ tales que $\sigma(L_1) = L_2$ y $C_2 = \sigma(C_1)$ es un subconjunto de C_1
- Ejemplos
 - La cláusula $\{q(z), p(x, y), p(a, y)\}$ se reduce a la cláusula $\{q(z), p(a, y)\}$ por simplificación de factores con la sustitución $\{x/a\}$
 - La cláusula $\{q(x), p(x, y), p(a, y)\}$ no se puede reducir por simplificación de factores

Factorización y simplificación de factores

factorizacion.in

```
set(auto2) .  
set(very_verbose) .  
  
list(sos) .  
  q(z) | p(x,y) | p(a,y) .  
  p(x,y) | p(a,z) | p(x,b) | -q(a,y,x) .  
end_of_list.
```

Factorización y simplificación de factores

```
0 [] q(x) | p(y,z) | p(a,z) .
** KEPT (pick-wt=5): 2 [copy,1,factor_simp] q(x) | p(a,y) .

0 [] p(x,y) | p(a,z) | p(x,b) | -q(a,y,x) .
** KEPT (pick-wt=13): 3 [] p(x,y) | p(a,z) | p(x,b) | -q(a,y,x) .

0 [factor,3.1.2] p(a,x) | p(a,b) | -q(a,x,a) .
** KEPT (pick-wt=10): 4 [factor,3.1.2] p(a,x) | p(a,b) | -q(a,x,a) .

0 [factor,3.1.3] p(x,b) | p(a,y) | -q(a,b,x) .
** KEPT (pick-wt=10): 5 [factor,3.1.3] p(x,b) | p(a,y) | -q(a,b,x) .

0 [factor,3.2.3] p(a,x) | p(a,b) | -q(a,x,a) .
Subsumed by 4.

0 [factor,4.1.2] p(a,b) | -q(a,b,a) .
** KEPT (pick-wt=7): 6 [factor,4.1.2] p(a,b) | -q(a,b,a) .

0 [factor,5.1.2] p(a,b) | -q(a,b,a) .
Subsumed by 6.
```

Factorización y simplificación de factores

- Explicación de las cláusulas generadas

2 [copy,1,factor_simp] $q(x) \mid p(a,y)$

Copy 1	$q(x) \mid p(y,z) \mid p(a,z)$
Unificador	$\sigma = \{y/a\}$
Factor_simp	$q(x) \mid p(a,z)$
Renombrado	$q(x) \mid p(a,y)$

7 [factor,3.1.2] $p(a,x) \mid p(a,b) \mid \neg q(a,x,a)$

3.1.2	$p(x,y) \mid p(a,z) \mid p(x,b) \mid \neg q(a,y,x)$
Unificador	$\sigma = \{x/a, y/z\}$
Factor	$p(a,z) \mid p(a,b) \mid \neg q(a,z,a)$
Renombrado	$p(a,x) \mid p(a,b) \mid \neg q(a,x,a)$

- Una resolvente $\mathbf{C}$ se simplifica por eliminación unitaria con respecto al literal $\mathbf{L}_1 \in \mathbf{C}$ si en soporte o usables existe una cláusula unitaria formada por el literal $\mathbf{L}_2$ y una sustitución σ tal que $\sigma(\overline{\mathbf{L}_2}) = \mathbf{L}_1$. El resultado de la reducción es $\mathbf{C} - \{\mathbf{L}_1\}$
- Ejemplos
 - La cláusula $\{\mathbf{p(a, x), q(a, x)}\}$ se reduce por eliminación unitaria con respecto al literal $\mathbf{q(a, x)}$, la cláusula $\{\neg \mathbf{q(u, v)}\}$ y la sustitución $\{\mathbf{u/a, v/x}\}$. El resultado de la reducción es $\{\mathbf{p(a, x)}\}$
 - La cláusula $\{\mathbf{p(a, x), q(a, x)}\}$ NO se reduce por eliminación unitaria con respecto al literal $\mathbf{q(a, x)}$ y la cláusula $\{\neg \mathbf{q(u, b)}\}$ (para que los literales fuesen iguales habría que asignar un valor a $\mathbf{x}$)

argumentacion-6.in

```
set(binary_res).

formula_list(sos).
  -MAS_RAPIDO(Lucero,Orejon).
end_of_list.

formula_list(usable).
  all x y (CABALLO(x) & PERRO(y) -> MAS_RAPIDO(x,y)).
  exists x (GALGO(x) &
    (all y (CONEJO(y) -> MAS_RAPIDO(x,y)))).
  CABALLO(Lucero).
  CONEJO(Orejon).
  all x (GALGO(x) -> PERRO(x)).
  all x (CONEJO(x) -> -PERRO(x)).
  all x (CONEJO(x) -> -CABALLO(x)).
  all x (PERRO(x) -> -CABALLO(x)).
  all x y z
    (MAS_RAPIDO(x,y) & MAS_RAPIDO(y,z) -> MAS_RAPIDO(x,z)).
end_of_list.
```

Prueba obtenida

```
1 [] -MAS_RAPIDO(Lucero,Orejon) .
2 [] -CABALLO(x) | -PERRO(y) | MAS_RAPIDO(x,y) .
3 [] GALGO($c1) .
4 [] -CONEJO(y) | MAS_RAPIDO($c1,y) .
5 [] CABALLO(Lucero) .
6 [] CONEJO(Orejon) .
7 [] -GALGO(x) | PERRO(x) .
11 [] -MAS_RAPIDO(x,y) | -MAS_RAPIDO(y,z) | MAS_RAPIDO(x,z) .
12 [binary,1.1,11.3] -MAS_RAPIDO(Lucero,x) |
    -MAS_RAPIDO(x,Orejon) .
17 [binary,12.2,4.2,unit_del,6] -MAS_RAPIDO(Lucero,$c1) .
19 [binary,17.1,2.3,unit_del,5] -PERRO($c1) .
20 [binary,19.1,7.2] -GALGO($c1) .
21 [binary,20.1,3.1] $F.
```

- Explicación de las cláusulas generadas

17 [binary,12.2,4.2,unit_del,6] -MAS_RAPIDO(Lucero,\$c1)

12.2	-MAS_RAPIDO(Lucero,x) -MAS_RAPIDO(x,Orejon)
4.2	-CONEJO(y) MAS_RAPIDO(\$c1,y)
Unificador	$\sigma = \{x/\$c1, y/Orejon\}$
Resolvente	-MAS_RAPIDO(Lucero,\$c1) -CONEJO(Orejon)
6	CONEJO(Orejon)
unit_del,6	-MAS_RAPIDO(Lucero,\$c1)

19 [binary,17.1,2.3,unit_del,5] -PERRO(\$c1)

17.1	-MAS_RAPIDO(Lucero,\$c1)
2.3	-CABALLO(x) -PERRO(y) MAS_RAPIDO(x,y)
Unificador	$\sigma = \{x/Lucero, y/\$c1\}$
Resolvente	-CABALLO(Lucero) -PERRO(\$c1)
5	CABALLO(Lucero)
unit_del,5	-PERRO(\$c1)

Eliminación de tautologías

- La cláusula C_1 es una tautología si contiene literales complementarios
- Ejemplos
 - La cláusula $\{p(x), \neg p(x), q(z, a)\}$ es una tautología
 - La cláusula $\{p(x), \neg p(y)\}$ no es una tautología

Eliminación de tautologías

tautologias.in

```
set(binary_res) .  
set(very_verbose) .  
  
list(sos) .  
  p(x) | -p(y) | q(x,y) .  
  p(x) | -p(y) | q(x,z) .  
  -q(x,x) .  
end_of_list.
```

Eliminación de tautologías

```
===== start of search =====  
given clause #1: (wt=3) 3 [] -q(x,x).  
given clause #2: (wt=7) 1 [] p(x) | -p(y) | q(x,y).  
  0 [binary,1.1,1.2] -p(x) | q(y,x) | p(z) | q(z,y).  
    Subsumed by 2.  
  0 [binary,1.2,1.1] p(x) | q(x,y) | -p(z) | q(y,z).  
    Subsumed by 2.  
  0 [binary,1.3,3.1] p(x) | -p(x).  
given clause #3: (wt=7) 2 [] p(x) | -p(y) | q(x,z).  
  0 [binary,2.1,2.2] -p(x) | q(y,z) | p(u) | q(u,v).  
    Subsumed by 2.  
...  
  0 [binary,2.2,1.1] p(x) | q(x,y) | -p(z) | q(u,z).  
    Subsumed by 2.  
  0 [binary,2.3,3.1] p(x) | -p(y).  
** KEPT (pick-wt=4): 4 [binary,2.3,3.1] p(x) | -p(y).  
4 back subsumes 2.  
4 back subsumes 1.
```

Conflicto unitario

- Una cláusula unitaria $\{L_1\}$ se reduce por conflicto unitario, si existe una cláusula unitaria $\{L_2\}$ y una sustitución σ tal que $\sigma(\overline{L_2}) = \sigma(L_1)$. Es decir, si L_1 y $\overline{L_2}$ son unificables
- Ejemplos
 - La cláusula $\{q(a, a)\}$ se reduce por conflicto unitario con respecto a la cláusula $\{\neg q(x, x)\}$ y la sustitución $\{x/a\}$
 - La cláusula $\{\neg q(a, a)\}$ se reduce por conflicto unitario con respecto a la cláusula $\{q(x, y)\}$ y la sustitución $\{x/a, y/a\}$
 - La cláusula $\{\neg q(a, y)\}$ se reduce por conflicto unitario con respecto a la cláusula $\{q(x, x)\}$ y la sustitución $\{x/a, y/a\}$

ejemplo-2.in

```
set(binary_res).  
set(very_verbose).  
  
formula_list(sos).  
  all x (p(x) -> q(x)).  
  all z exists y (r(z) -> p(y)).  
  exists x r(x).  
  all z -q(z).  
end_of_list.
```


Conflicto unitario

```
===== start of search =====
given clause #1: (wt=2) 3 [] r($c1).
given clause #2: (wt=2) 4 [] -q(z).
given clause #3: (wt=4) 1 [] -p(x) | q(x).
    0 [binary,1.2,4.1] -p(x).
** KEPT (pick-wt=2): 5 [binary,1.2,4.1] -p(x).
5 back subsumes 1.
given clause #4: (wt=2) 5 [binary,1.2,4.1] -p(x).
given clause #5: (wt=5) 2 [] -r(z) | p($f1(z)).
    0 [binary,2.1,3.1] p($f1($c1)).
** KEPT (pick-wt=3): 6 [binary,2.1,3.1] p($f1($c1)).
----> UNIT CONFLICT at 0.00 sec ----> 7 [binary,6.1,5.1] $F.
```

Obtención de respuestas

- Dado un conjunto de fórmulas S y una fórmula $F(x_1, \dots, x_n)$, cuyas variables libres son $x_1, \dots, x_n$, encontrar términos $t_1, \dots, t_n$ tales que $F(t_1, \dots, t_n)$ sea consecuencia de S
- Procedimiento de solución
 - Introducir un nuevo predicado **\$ANS**
 - Considerar el conjunto de las cláusulas correspondientes a las fórmulas de
$$S \cup \{\forall x_1 \dots x_n (F(x_1, \dots, x_n) \rightarrow \text{\textbf{\$ANS}}(x_1, \dots, x_n))\}$$
 - Aplicar el procedimiento de resolución hasta encontrar una cláusula cuyo único literal contenga el predicado **\$ANS**
 - Los términos que aparecen en dicho literal forman una respuesta a la cuestión planteada

Obtención de respuestas

- Dado $\{\forall x (P(x) \rightarrow Q(x)), P(a)\}$ determinar un z tal que $Q(z)$ sea consecuencia del conjunto

respuestas-1.in

```
set(binary_res) .  
  
formula_list(sos) .  
  all x (P(x) -> Q(x)) .  
  P(a) .  
  all z (Q(z) -> $ANS(z)) .  
end_of_list.
```

Obtención de varias respuestas

- Dado $\{\forall x(P(x) \rightarrow Q(x)), P(a) \wedge P(b)\}$ determinar un z tal que $Q(z)$ sea consecuencia del conjunto
- `assign(max_proofs, N)`: Número máximo de pruebas encontradas

respuestas-2a.in

```
set(binary_res) .
assign(max_proofs, 2) .

formula_list(sos) .
  all x (P(x) -> Q(x)) .
  P(a) & P(b) .
  all z (Q(z) -> $ANS(z)) .
end_of_list.
```

Obtención de todas las pruebas posibles

- `assign(max_proofs,-1)`: Obtención de todas las pruebas posibles

respuestas-2b.in

```
set(binary_res) .
assign(max_proofs,-1) .

formula_list(sos) .
  all x (P(x) -> Q(x)) .
  P(a) & P(b) .
  all z (Q(z) -> $ANS(z)) .
end_of_list.
```

Obtención de todas las respuestas posibles

- `formula_list(passive)`: Fórmulas a utilizar en último lugar

respuestas-2c.in

```
set(binary_res) .
assign(max_proofs, -1) .

formula_list(sos) .
  all x (P(x) -> Q(x)) .
  P(a) & P(b) .
end_of_list .

formula_list(passive) .
  all z (Q(z) -> $ANS(z)) .
end_of_list .
```

Obtención de respuestas

- De las siguientes personas Juan, Jorge, Víctor, María, Agata y Carla se sabe que María, Jorge y Victor son ricos y que María y Juan se aman, que Víctor ama a María y que Jorge y Víctor se aman. Admitiendo que dos personas pueden casarse si son de distintos sexo y se aman o una es rica y ama a la otra, determinar las parejas que pueden casarse

respuestas-3.in

```
set(ur_res) .
assign(max_proofs, -1) .

list(usable) .
  Hombre(Juan) .   Hombre(Jorge) .   Hombre(Victor) .
  Mujer(Maria) .   Mujer(Agata) .     Mujer(Carla) .
  Rico(Maria) .    Rico(Jorge) .      Rico(Victor) .
  Ama(Maria, Juan) .      Ama(Juan, Maria) .
  Ama(Victor, Maria) .    Ama(Jorge, Victor) .
  Ama(Victor, Jorge) .
```

respuestas-3.in

```
% Los hombres y las mujeres son de sexos distintos:
-Mujer(x) | -Hombre(y) | Distinto_sexo(x,y) .
-Mujer(x) | -Hombre(y) | Distinto_sexo(y,x) .
% Dos personas de distintos sexo pueden casarse si
% se aman o una es rica y ama a la otra:
-Distinto_sexo(x,y) | -Ama(x,y) |
                        -Ama(y,x) | Pueden_casarse(x,y) .
-Distinto_sexo(x,y) | -Ama(x,y) |
                        -Rico(x)   | Pueden_casarse(x,y) .
end_of_list.

list(sos) .
-Pueden_casarse(x,y) | $ANS(x,y) .
end_of_list.
```


Primera respuesta

```
3 [] Hombre(Victor) .
4 [] Mujer(Maria) .
9 [] Rico(Victor) .
12 [] Ama(Victor,Maria) .
16 [] -Mujer(x) | -Hombre(y) | Distinto_sexo(y,x) .
18 [] -Distinto_sexo(x,y) | -Ama(x,y) | -Rico(x) |
    Pueden_casarse(x,y) .
19 [] -Pueden_casarse(x,y) | $ANS(x,y) .
22 [ur,19,18,12,9] $ANS(Victor,Maria) |
    -Distinto_sexo(Victor,Maria) .
27 [ur,22,16,3] $ANS(Victor,Maria) | -Mujer(Maria) .
28 [binary,27.1,4.1] $ANS(Victor,Maria) .
```

Segunda respuesta

```
1 [] Hombre(Juan) .
4 [] Mujer(Maria) .
7 [] Rico(Maria) .
10 [] Ama(Maria,Juan) .
15 [] -Mujer(x) | -Hombre(y) | Distinto_sexo(x,y) .
18 [] -Distinto_sexo(x,y) | -Ama(x,y) | -Rico(x) |
    Pueden_casarse(x,y) .
19 [] -Pueden_casarse(x,y) | $ANS(x,y) .
23 [ur,19,18,10,7] $ANS(Maria,Juan) |
    -Distinto_sexo(Maria,Juan) .
31 [ur,23,15,4] $ANS(Maria,Juan) | -Hombre(Juan) .
32 [binary,31.1,1.1] $ANS(Maria,Juan) .
```

- Si toda persona es hijo de su padre y los hijos de los hijos son nietos entonces, si Luis es nieto de X, ¿quién es X?

respuestas-4.in

```
set(ur_res) .

formula_list(usable) .
  % Toda persona es hijo de su padre:
  all x exists y Hijo(x,y) .
  % Los hijos de los hijos son nietos:
  all x y z (Hijo(x,y) & Hijo(y,z) -> Nieto(x,z)) .
end_of_list.

formula_list(sos) .
  % Si Luis es nieto de x, ¿quién es x?:
  all x (Nieto(Luis,x) -> $ANS(x)) .
end_of_list.
```

- Si Luna es una persona y todas las personas están solteras o casadas, ¿cuál es el estado civil de Luna?

respuestas-5.in

```
set(ur_res) .

formula_list(usable) .
  Persona(Luna) .
  all x (Persona(x) -> Estado(x,Soltero) |
        Estado(x,Casado)) .
end_of_list.

formula_list(sos) .
  all x (Estado(Luna, x) -> $ANS(x)) .
end_of_list.
```

- Alonso, J.A., Fernández, A. y Pérez, M.J. *Razonamiento automático*. (en *Lógica formal (Orígenes, métodos y aplicaciones)*, Ed. Kronos, 1995)
- Chang, C.L. y Lee, R.C.T. *Symbolic logic and mechanical theorem proving*. (Academic Press, 1973)
- Genesereth, M.R. *Computational Logic*
 - Cap. 9 “Relational resolution”
- Genesereth, M.R. y Nilsson, N.J. *Logical foundations of Artificial Intelligence* (Morgan Kaufmann, 1987)
 - Cap. 4 “Resolution”
 - Cap. 5 “Resolution strategies”
- Wos, L., Overbeek, R., Lusk, E. y Boyle, J. *Automated Reasoning: Introduction and Applications*, (2nd ed.) (McGraw-Hill, 1992)