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

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

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

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

$$\frac{Q(a) \vee \neg R(z) \qquad \neg Q(z)}{\neg R(z)} \{z/a\}$$

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

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

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

$$\frac{\neg R(z) \qquad R(b)}{} \quad \{z/b\}$$

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

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

$$\frac{Q(a) \vee \neg R(z_1) \qquad \neg Q(z_2)}{\neg R(z_1)} \{z_2/a\}$$

$$\frac{\neg R(z_1) \qquad R(b)}{} \{z_1/b\}$$

- Renombrado de variables

Transformación a cláusulas

1. Eliminar las equivalencias usando la relación

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

2. Eliminar las implicaciones usando la equivalencia

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

3. Interiorizar las negaciones usando las equivalencias

   $$\neg(\boldsymbol{A} \wedge \boldsymbol{B}) \equiv \neg \boldsymbol{A} \vee \neg \boldsymbol{B}$$
   $$\neg(\boldsymbol{A} \vee \boldsymbol{B}) \equiv \neg \boldsymbol{A} \wedge \neg \boldsymbol{B}$$
   $$\neg\neg \boldsymbol{A} \equiv \boldsymbol{A}$$
   $$\neg(\forall \boldsymbol{x}\ \boldsymbol{A}) \equiv \exists \boldsymbol{x}\ \neg \boldsymbol{A}$$
   $$\neg(\exists \boldsymbol{x}\ \boldsymbol{A}) \equiv \forall \boldsymbol{x}\ \neg \boldsymbol{A}$$

4. Renombrar variables cuantificadas

Transformación a cláusulas

5. Skolemización: Eliminación de cuantificaciones existenciales

$$\forall x_1 \ldots x_n \; \exists y \; F(x_1, \ldots, x_n, y) \equiv$$
$$\equiv \forall x_1 \ldots x_n \; F(x_1, \ldots, x_n, f_y(x_1, \ldots, x_n))$$

6. Agrupar cuantificaciones al principio
7. Interiorizar las disyunciones usando las propiedades distributivas

$$A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$$
$$(A \wedge B) \vee C \equiv (A \vee C) \wedge (B \vee C)$$

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.
```

Clausificación en OTTER

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`, ...

Resolución binaria

- Regla de resolución binaria

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

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

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

Procedimiento de unificación de primer orden

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

$$\begin{aligned}
\sigma(\{E_1, \ldots, E_n\}) &= \{\sigma(E_1), \ldots, \sigma(E_n)\} \\
\sigma(f(s_1, \ldots, s_m)) &= f(\sigma(s_1), \ldots, \sigma(s_m)) \\
\sigma(X_i) &= t_i \\
\sigma(X) &= X, \text{ si } X \notin \text{dominio}(\sigma)
\end{aligned}$$

- Composición de sustituciones

$$\sigma = \{X_1/t_1, \ldots, X_n/t_n\}$$
$$\theta = \{Y_1/s_1, \ldots, Y_m/s_m\}$$

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

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 $f(b, Y)$

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

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

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

Ejemplos de unificación de primer orden

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

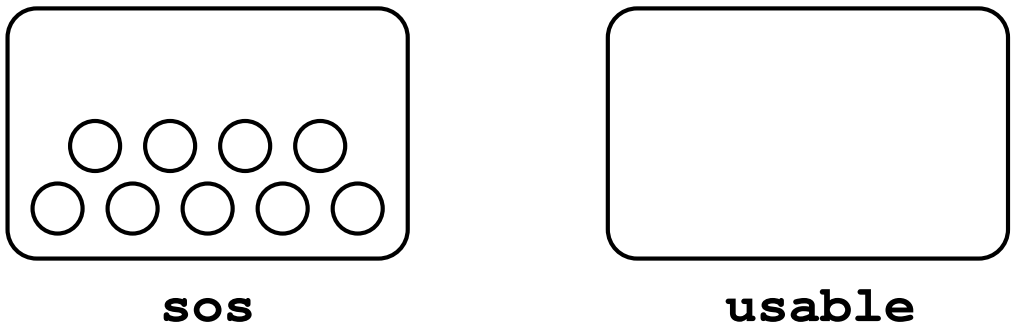
$$\begin{aligned}
&\langle \{f(Y,X) \approx f(X,g(Y))\}; \{\}\rangle \\
&\quad \Rightarrow_u \langle \{Y \approx X, X \approx g(Y)\}; \{\}\rangle \\
&\quad \Rightarrow_u \langle \{X \approx g(X)\}; \{Y/X\}\rangle \\
&\quad \Rightarrow_u \bot
\end{aligned}$$

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

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

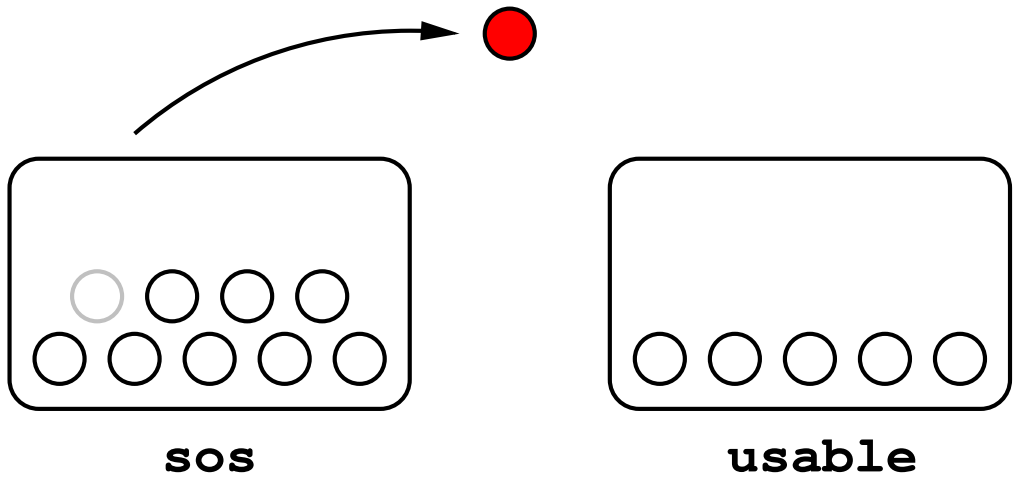
Refutación por resolución en OTTER

- 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



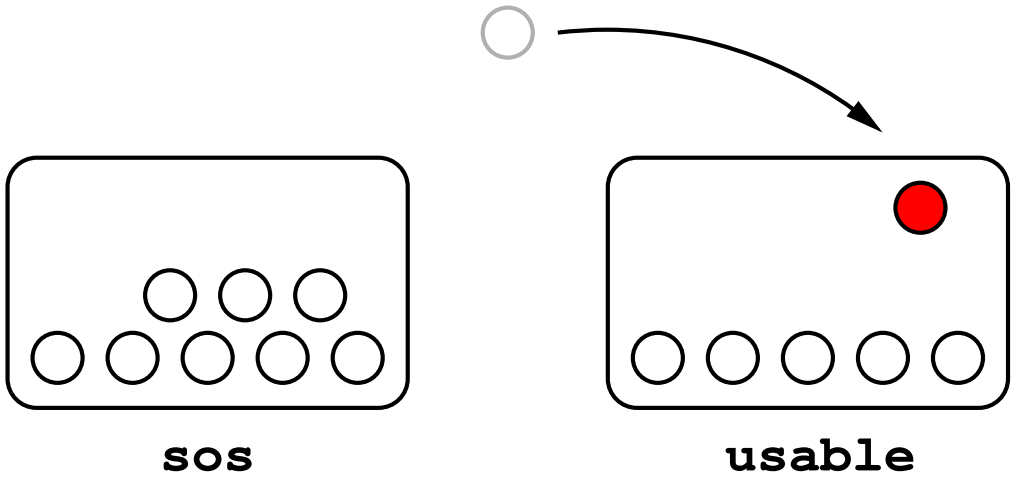
Refutación por resolución en OTTER

- 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;



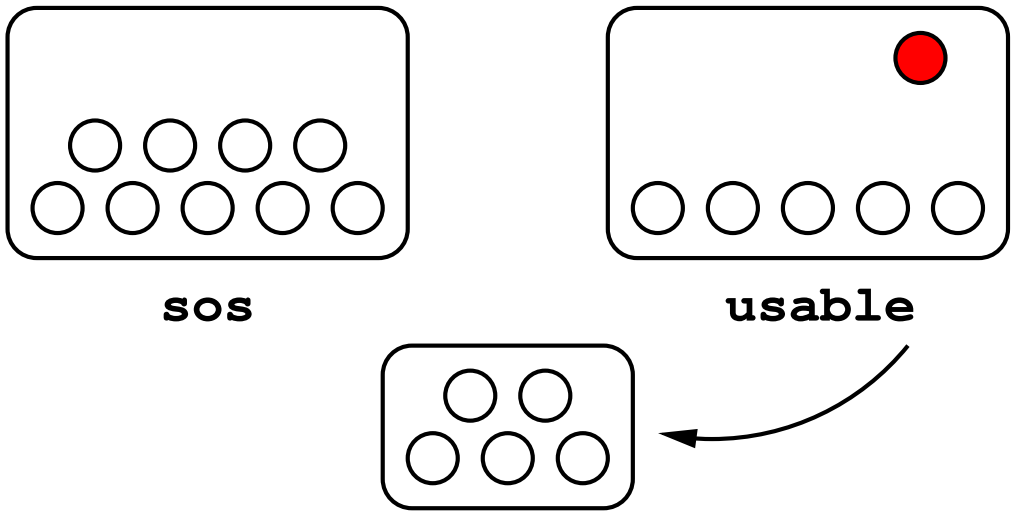
Refutación por resolución en OTTER

- 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;



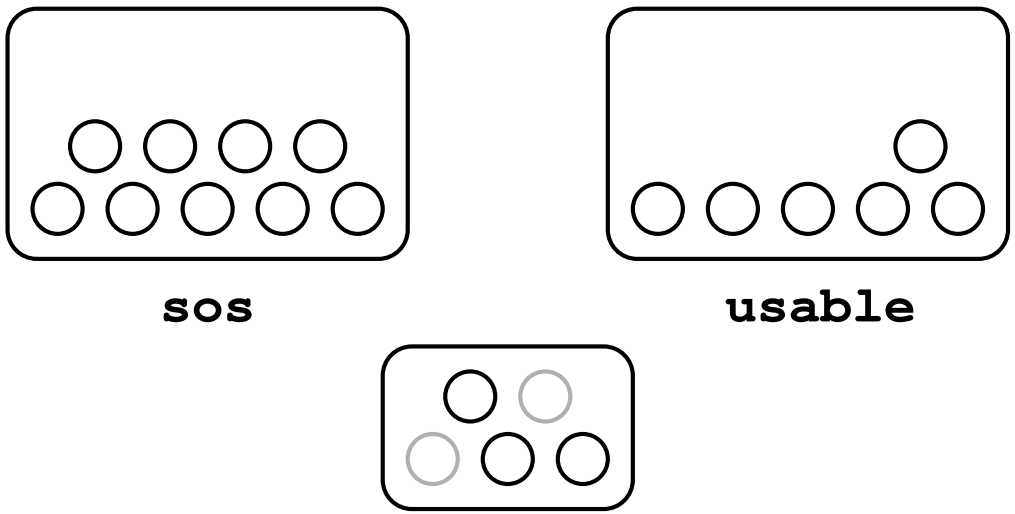
Refutación por resolución en OTTER

- 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;



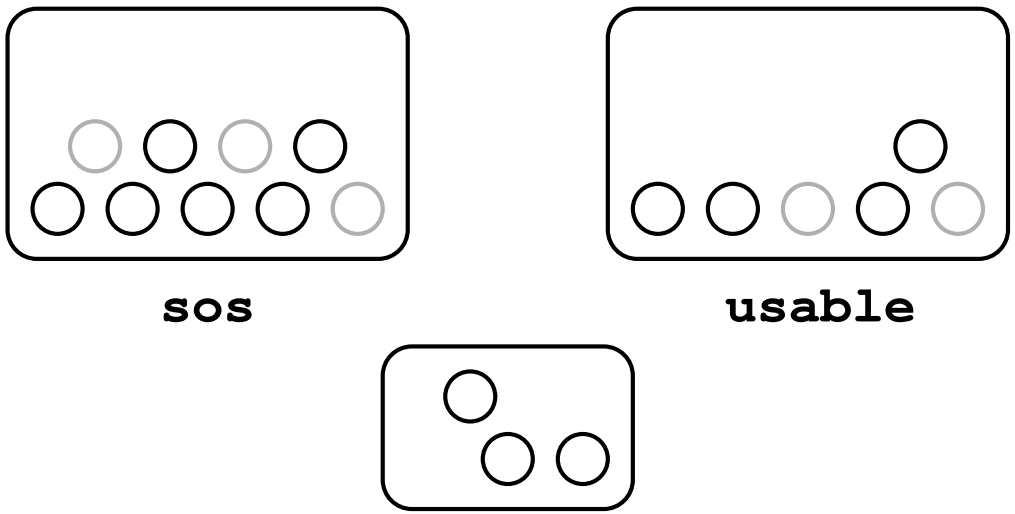
Refutación por resolución en OTTER

- 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;



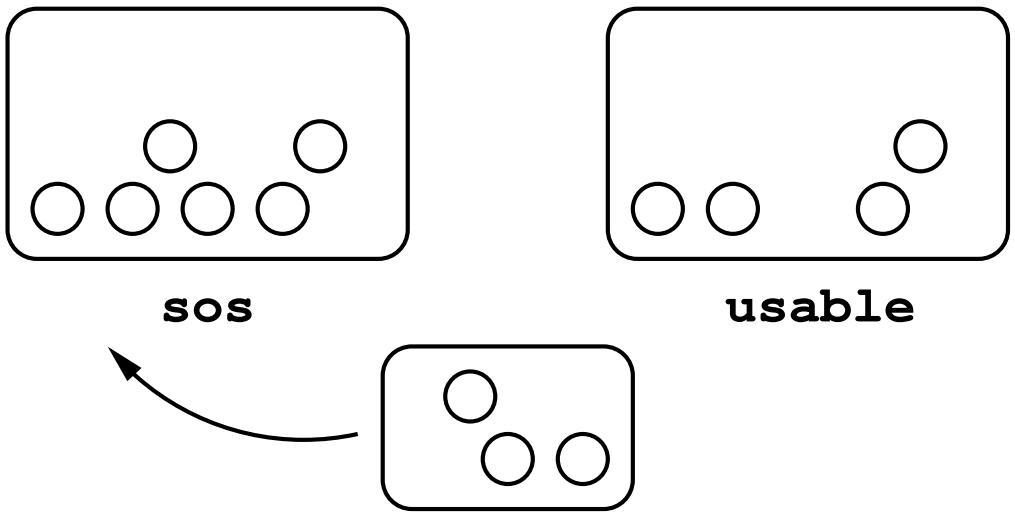
Refutación por resolución en OTTER

- 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;



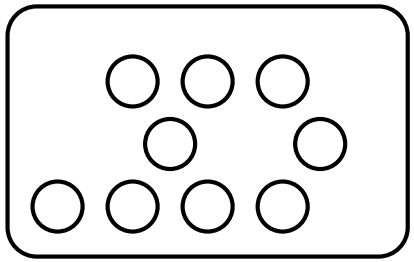
Refutación por resolución en OTTER

- 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;

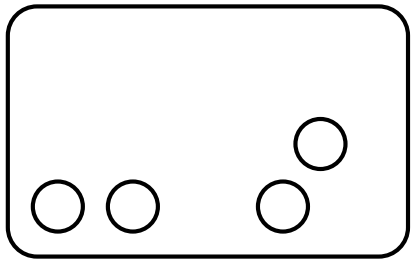


Refutación por resolución en OTTER

- 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

Refutación por resolución en OTTER

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

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.
```

Refutación por resolución en OTTER

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.
```

Pruebas en OTTER

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 -------------
```

Argumentaciones

- Explicación de las cláusulas generadas

```
5 [binary,1.2,4.1] -p(x)

   1.2            -p(x) | q(x)
   4.1            -q(z)
   Unificador     σ = {x/z}
   Resolvente     -p(z)
   Renombrado     -p(x)


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

   2.1            -r(x) | p($f1($c1))
   3.1            r($c1)
   Unificador     σ = {x/$c1}
   Resolvente     p($f1($c1))
```

Cláusulas en OTTER

- Disyunción de literales: `-p(x) | q(x)`, `-r(z) | 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`

Cláusulas en OTTER

- `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.
```

Argumentaciones

- 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`	Lucero
`Orejon`	Orejón
`CABALLO(x)`	`x` es un caballo
`CONEJO(x)`	`x` es un conejo
`GALGO(x)`	`x` es un galgo
`PERRO(x)`	`x` es un perro
`MAS_RAPIDO(x,y)`	`x` es más rápido que `y`

Argumentaciones

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.
```

Argumentaciones

- 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 ============
```

Argumentaciones

- 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?

Argumentaciones

- 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?

Argumentaciones

- Información implícita: *La relación "ser más rapido 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.
```

La estrategia del conjunto soporte

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).
19 [binary,12.2,4.2] -MAS_RAPIDO(Lucero,$c1) |
                     -CONEJO(Orejon).
22 [binary,19.2,6.1] -MAS_RAPIDO(Lucero,$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)

$$\frac{\begin{array}{l} L_1 \vee \ldots \vee L_n \\ M_1 \\ \vdots \\ M_{i-1} \\ M_{i+1} \\ \vdots \\ M_n \end{array}}{\sigma(L_i)}$$

  donde $\sigma = umg(L_j, \overline{M_j})$, $j \in \{1, \ldots, i-1, i+1, \ldots, n\}$

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

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.
```

Resolución Unitaria (UR)

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            -CABALLO(x) | -PERRO(y) | MAS_RAPIDO(x,y)
   4            CABALLO(Lucero)
   12           PERRO($c1)
   Unificador   σ = {x/Lucero,y/$c1}
   Resolvente   MAS_RAPIDO(Lucero,$c1)

23 [ur,10,21,11] -MAS_RAPIDO(Lucero,$c1)

   10           -MAS_RAPIDO(x,y) | -MAS_RAPIDO(y,z) |
                                    MAS_RAPIDO(x,z)
   21           MAS_RAPIDO($c1,Orejon)
   11           -MAS_RAPIDO(Lucero,Orejon)
   Unificador   σ = {x/Lucero,y/$c1,z/Orejon}
   Resolvente   -MAS_RAPIDO(Lucero,$c1)
```

Resolución semántica

- Dado un conjunto de cláusulas cerradas $\boldsymbol{S}$ que no contiene la cláusula vacía y una estructura $\mathcal{I}$ cualquiera. El conjunto $\boldsymbol{S}$ se divide en dos subconjuntos:
  - El conjunto $\boldsymbol{S_0}$ formado por todas las fórmulas de $\boldsymbol{S}$ que son falsas en $\mathcal{I}$.
  - El conjunto $\boldsymbol{S_1}$ formado por todas las fórmulas de $\boldsymbol{S}$ 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

$$\frac{\begin{array}{l}\neg A_1 \vee \ldots \vee \neg A_n \vee B_1 \vee \ldots \vee B_m \\ M_1 \vee M_{1,1} \vee \ldots \vee M_{1,h_1} \\ \vdots \\ M_n \vee M_{n,1} \vee \ldots \vee M_{n,h_n}\end{array}}{\sigma(B_1 \vee \ldots \vee B_m \vee (\bigvee_{i,j} M_{i,j}))}$$

donde $A_1, \ldots, A_n, B_1, \ldots, B_m, M_i, M_{i,j}$ son átomos y $\sigma = umg(A_i, M_i)$, $i \in \{1, \ldots, n\}$

- Regla de hiper–resolución negativa

Hiper–resolución

- `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.
```

Hiper–resolución

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.
```

Hiper–resolución

- Explicación de las cláusulas generadas

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

   1              -CABALLO(x) | -PERRO(y) | MAS_RAPIDO(x,y)
   4              CABALLO(Lucero)
   12             PERRO($c1)
   Unificador     σ = {x/Lucero,y/$c1}
   Resolvente     MAS_RAPIDO(Lucero,$c1)

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

   10             -MAS_RAPIDO(x,y) | -MAS_RAPIDO(y,z) |
                                     MAS_RAPIDO(x,z)
   14             MAS_RAPIDO(Lucero,$c1)
   13             MAS_RAPIDO($c1,Orejon)
   Unificador     σ = {x/Lucero,y/$c1,z/Orejon}
   Resolvente     MAS_RAPIDO(Lucero,Orejon)
```

Hiper–resolución

- `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.
```

Hiper–resolución

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.
```

Hiper–resolución

- Explicación de las cláusulas generadas

```
19 [neg_hyper,10,11] -MAS_RAPIDO(Lucero,x) |
                     -MAS_RAPIDO(x,Orejon).

   10              -MAS_RAPIDO(x,y) | -MAS_RAPIDO(y,z) |
                                      MAS_RAPIDO(x,z)
   11              -MAS_RAPIDO(Lucero,Orejon).
   Unificador      σ = {x/Lucero,z/Orejon}
   Resolvente      -MAS_RAPIDO(Lucero,y) | -MAS_RAPIDO(y,Orejon).
   Renombrado      -MAS_RAPIDO(Lucero,x) | -MAS_RAPIDO(x,Orejon).


22 [neg_hyper,19,3] -MAS_RAPIDO(Lucero,$c1) | -CONEJO(Orejon).

   19              -MAS_RAPIDO(Lucero,x) | -MAS_RAPIDO(x,Orejon) |
   3               -CONEJO(y) | MAS_RAPIDO($c1,y)
   Unificador      σ = {x/$c1,y/Orejon}
   Resolvente      -MAS_RAPIDO(Lucero,$c1) | -CONEJO(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

Peso de una cláusula

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).
...
```

Subsumción

- 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 C con dos literales $L_1, L_2 \in C$ tales que esiste una sustitución σ tal que $\sigma(L_1) = \sigma(L_2)$, entonces se genera la cláusula $\sigma(C)$ (factor de C).
- Ejemplos. Sea $C = \{p(x, y), p(a, z), p(x, b)\}$
  - $\{p(a, z), p(a, b)\}$ es un factor de C que se obtiene combinando los literales $p(x, y)$ y $p(a, z)$ con la sustitución $\sigma = \{x/a, y/z\}$
  - $\{p(x, b), p(a, z)\}$ es un factor de C que se obtiene combinando los literales $p(x, y)$ y $p(x, b)$ con la sustitución $\sigma = \{y/b\}$
  - $\{p(a, y), p(a, b)\}$ es un factor de C que se obtiene combinando los literales $p(a, z)$ y $p(x, b)$ con la sustitución $\sigma = \{x/a, z/b\}$

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) | p(a,y)

   Copy 1            q(x) | p(y,z) | p(a,z)
   Unificador        σ = {y/a}
   Factor_simp       q(x) | p(a,z)
   Renombrado        q(x) | p(a,y)

7 [factor,3.1.2] p(a,x) | p(a,b) | -q(a,x,a)

   3.1.2             p(x,y) | p(a,z) | p(x,b) | -q(a,y,x)
   Unificador        σ = {x/a,y/z}
   Factor            p(a,z) | p(a,b) | -q(a,z,a)
   Renombrado        p(a,x) | p(a,b) | -q(a,x,a)
```

Eliminación unitaria

- Una resolvente $\boldsymbol{C}$ se simplifica por eliminación unitaria con respecto al literal $\boldsymbol{L_1} \in \boldsymbol{C}$ si en soporte o usables existe una cláusula unitaria formada por el literal $\boldsymbol{L_2}$ y una sustitución $\boldsymbol{\sigma}$ tal que $\boldsymbol{\sigma}(\overline{\boldsymbol{L_2}}) = \boldsymbol{L_1}$. El resultado de la reducción es $\boldsymbol{C} - \{\boldsymbol{L_1}\}$
- Ejemplos
  - La cláusula $\{\boldsymbol{p}(\boldsymbol{a}, \boldsymbol{x}), \boldsymbol{q}(\boldsymbol{a}, \boldsymbol{x})\}$ se reduce por eliminación unitaria con respecto al literal $\boldsymbol{q}(\boldsymbol{a}, \boldsymbol{x})$, la cláusula $\{\neg \boldsymbol{q}(\boldsymbol{u}, \boldsymbol{v})\}$ y la sustitución $\{\boldsymbol{u}/\boldsymbol{a}, \boldsymbol{v}/\boldsymbol{x}\}$. El resultado de la reducción es $\{\boldsymbol{p}(\boldsymbol{a}, \boldsymbol{x})\}$
  - La cláusula $\{\boldsymbol{p}(\boldsymbol{a}, \boldsymbol{x}), \boldsymbol{q}(\boldsymbol{a}, \boldsymbol{x})\}$ NO se reduce por eliminación unitaria con respecto al literal $\boldsymbol{q}(\boldsymbol{a}, \boldsymbol{x})$ y la cláusula $\{\neg \boldsymbol{q}(\boldsymbol{u}, \boldsymbol{b})\}$ (para que los literales fuesen iguales habría que asignar un valor a $\boldsymbol{x}$)

Eliminación unitaria

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.
```

Eliminación unitaria

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.
```

Eliminación unitaria

- 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($c1,y)
   Unificador     σ = {x/$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,$c1)
   2.3            -CABALLO(x) | -PERRO(y) | MAS_RAPIDO(x,y)
   Unificador     σ = {x/Lucero,y/$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 $\{x/a, y/a\}$

Conflicto unitario

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 $F(x_1, \ldots, x_n)$, cuyas variables libres son $x_1, \ldots, x_n$, encontrar términos $t_1, \ldots, t_n$ tales que $F(t_1, \ldots, t_n)$ sea consecuencia de S
- Procedimiento de solución
  - Introducir un nuevo predicado $\$ANS$
  - Considerar el conjunto de las cláusulas correspondientes a las fórmulas de
    $S \cup \{\forall x_1 \ldots x_n \ (F(x_1, \ldots, x_n) \rightarrow \$ANS(x_1, \ldots, x_n))\}$
  - 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).
```

Obtención de respuestas

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.
```

Obtención de respuestas

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).
```

Obtención de respuestas

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).
```

Obtención de respuestas

- 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.
```

Obtención de respuestas

- 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.
```

Bibliografía

- 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)