Representación de Conocimiento (2)

Carlos Hurtado L.

Depto de Ciencias de la Computación,
Universidad de Chile

Resolución

- Procedimiento de prueba correcto y completo para Cálculo de Proposiciones
- Se puede generalizar a Cálculo de Predicados de Primer Orden
- Deriva en métodos eficientes para fragmentos como cláusulas de Horn.
- Base de motor de inferencia en Prolog
- Requiere expresar sentencias en forma clausal

Cláusulas

- Un literal es un átomo o la negación de un átomo. Por ejemplo P, $\neg P$, Q, $\neg Q$ son literales
- Una cláusula es una disyunción de literales. Por ejemplo, $(\neg P \vee Q \vee R)$ es una cláusula.
  - Notación abreviada: $(\neg P, Q, R)$
  - Cláusula vacía () denota "Falso"
- Una sentencia en forma CNF (Conjunctive Normal Form) es una conjunción de cláusulas. Ejemplo: $(\neg P \vee Q \vee R) \wedge (\neg Q \vee R) \wedge (\neg R \vee \neg Q)$
  - Notación abreviada: $\{(\neg P, Q, R), (\neg Q, R), (\neg R, \neg Q)\}$.
- Toda sentencia se puede transformar vía procedimiento simple en una sentencia CNF.

Regla de Resolución

$$\frac{(p, q_1, \ldots, q_n) \quad (\neg p, r_1, \ldots, r_n)}{(q_1, \ldots, q_k, r_1, \ldots, r_n)}$$

Resolvente

Otra forma de ver la regla de resolución:

$$\frac{(\neg q_1 \wedge \ldots \wedge \neg q_k \Rightarrow p) \quad (p \Rightarrow r_1 \vee \ldots \vee r_n)}{(\neg q_1 \wedge \ldots \wedge \neg q_k \Rightarrow r_1 \vee \ldots \vee r_n)}$$

Casos Especiales de Regla de Resolución

Modus Ponens

$$\frac{p \Rightarrow q \quad p}{q}$$

$$\frac{\{\neg p, q\} \quad \{p\}}{\{q\}}$$

Modus Tolens

$$\frac{p \Rightarrow q \quad \neg q}{\neg p}$$

$$\frac{\{\neg p, q\} \quad \{\neg q\}}{\{\neg p\}}$$

Encadenamiento

$$\frac{p \Rightarrow q \quad q \Rightarrow r}{p \Rightarrow r}$$

$$\frac{\{\neg p, q\} \quad \{\neg q, r\}}{\{\neg p, r\}}$$

Regla de Resolución: Conclusiones Múltiples

$$\frac{(P, Q, \neg R) \quad (P, W, \neg Q, R)}{(P, Q, W, \neg Q)}$$

$$\frac{(P, Q, \neg R) \quad (P, W, \neg Q, R)}{(P, \neg R, W, R)}$$

En este caso tenemos dos resolventes, pero ambos son "Verdadero"

Regla de Resolución

- Es correcta
  - Ejercicio: demostrarlo.
- No es completa
  - Ejemplo: no podemos derivar ($P \vee R$) a partir de ($P \wedge R$) usando la regla de resolución.
- Refutación mediante Resolución
  - Idea: usar regla de resolución para generar una refutación.
  - Refutación: demostración de "Falso" a partir de un conjunto de sentencias.

Refutación mediante resolución

Recordar Teorema de Deducción:

$KB \models \alpha$ ssi $(KB \wedge \neg \alpha) \models F$

Luego el problema se reduce a encontrar una refutación para $(KB \wedge \neg \alpha)$

Refutación Mediate Resolución

¿ KB $\models \alpha$?

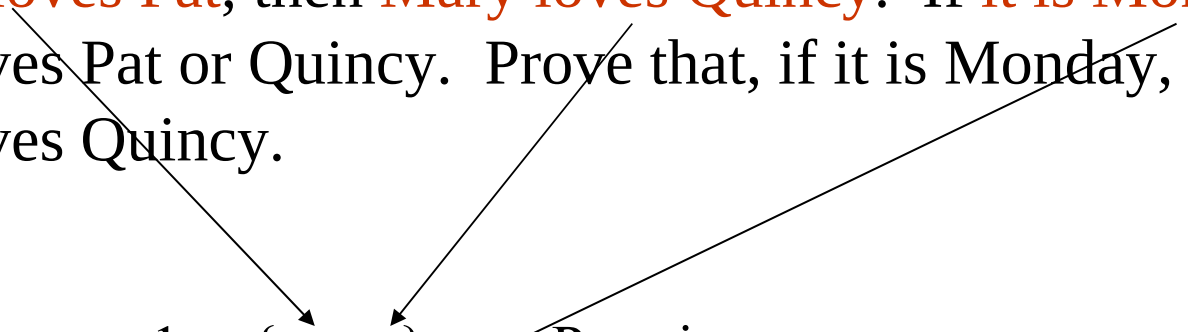
2. Agregar $\neg \alpha$ to KB
3. Convertir KB en CNF
4. En forma iterativa, aplicar regla de resolución a cláusulas de KB e ir añadiendo los resultados a KB, hasta que:
   - (A) no tengamos nuevos resolventes, o
   - (B) se generó la clásula vacía.
5. Si (A) return(false), si (B) return(true)

Ejercicio

If Mary loves Pat, then Mary loves Quincy. If it is Monday, Mary loves Pat or Quincy. Prove that, if it is Monday, then Mary loves Quincy.

Ejemplo

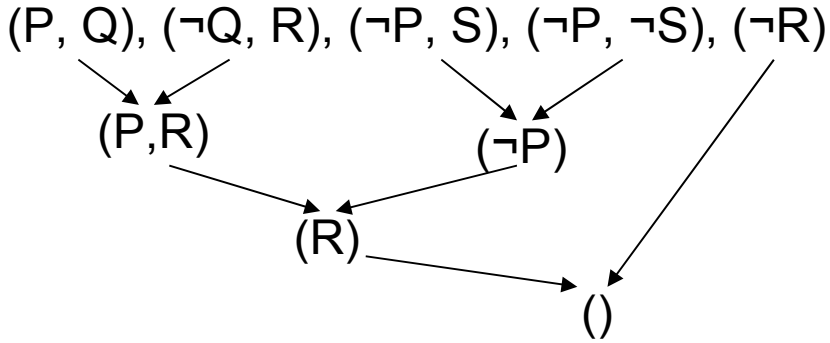
If Mary loves Pat, then Mary loves Quincy. If it is Monday, Mary loves Pat or Quincy. Prove that, if it is Monday, then Mary loves Quincy.



1.	$\{\neg p, q\}$	Premisa
2.	$\{\neg m, p, q\}$	Premisa
3.	$\{m\}$	Objetivo Negado
4.	$\{\neg q\}$	Objetivo Negado
5.	$\{p, q\}$	3,2
6.	$\{q\}$	5,1
7.	$\{\}$	6,4

Ejemplo

- KB={ (P, Q), (¬Q, R), (¬P, S), (¬P, ¬S)}, α=R.
- Agregamos (¬α) a KB
- Aplicamos regla de resolución para construir una refutación:



- Lo que genera la siguiente refutación:
  (P, Q), (¬Q, R), (P,R), (¬P, S), (¬P, ¬S), (¬P), (R), (¬R),()
- Por lo tanto, KB $\models$ α.

Propiedades de Refutación Mediante Resolución

- Teorema: Regla de resolución es completa y correcta para generar refutaciones

  Resolución de $(KB \wedge \neg\alpha)$ genera la cláusula vacía ssi $KB \models \alpha$

¿Por qué funciona?

- La regla de resolución no siempre genera α:

  KB = { {a,b}, {$\neg$a,b}, {b,c} }

  α = b $\vee \neg$c = {b,$\neg$c}

- **Teorema**: La regla de resolución genera una cláusula contenida en α ssi KB $\models \alpha$
- Ejemplo: Al aplicar regla de resolución a KB anterior obtenemos b

Resolución Estrategias de Búsqueda

- Proceso de resolución: resolver sucesivamente hasta condición de término.
- Estrategias de ordenamiento:
  - Definen el orden en que se realizan las resoluciones
    - Primero en Anchura
    - Primero en Profundidad
- Estrategias de refinamiento:
  - Imponen restricciones en las resoluciones a realizar:
    - Conjunto Soporte
    - Resolución Lineal
    - Resolución Unitaria

Estrategia Primero en Anchura

- Res(a,b):= cláusula que se obtienen de resolver a y b.
- Resolventes nivel 0,
  - R(0):= KB
- Resolventes nivel 1,
  - R(1):={Res(a,b) | a,b in R(0)} – R(0)
- …
- Resolventes nivel i+1,
  - R(i+1):={Res(a,b) | a en R(i) y b en R(k), k <= i} – (Union$_{j<i}$ R(j))

Estrategia Primero en Anchura (cont.)

- R(0):=KB, i:=1;
- Loop
  - Calcular R(i),
  - Si (() en R(i)) return(true)
  - Si (R(i)={}) return(false)
  - i:=i+1

Ejemplo: Primero en Anchura

$$(P,Q),(\neg Q,R),(\neg P,S),(\neg P,\neg S),(\neg R)$$

$$(P,R),(Q,S),(Q,\neg S),(\neg Q),(\neg P)$$

$$(R),(Q),(S),(\neg S),(R,S)(R,\neg S),(P),(Q,\neg P)$$

$$....()$$

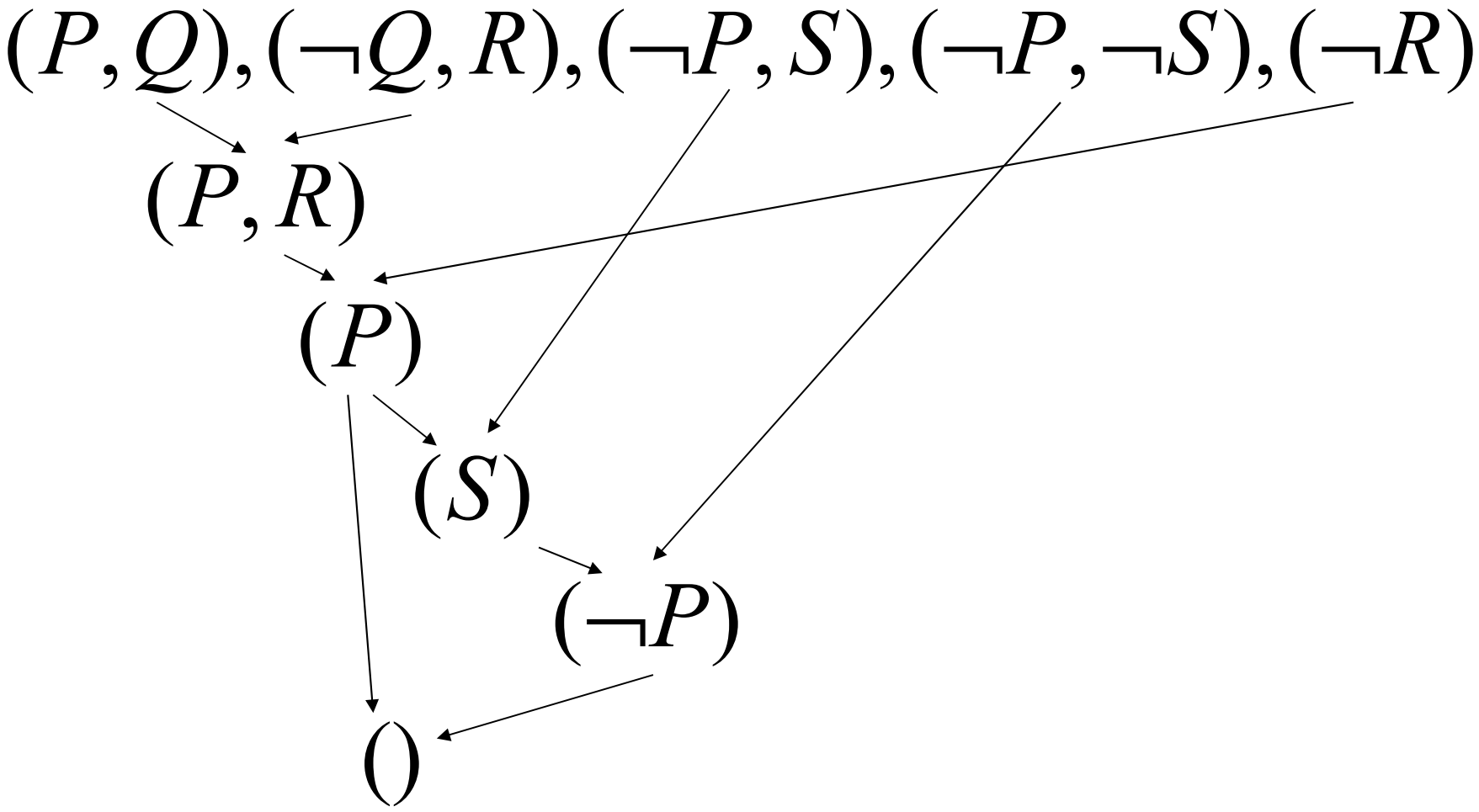
Complejidad de Resolución Primero en Anchura

- Dado un conjunto de m cláusulas con n átomos:
  - Aplicar regla de resolución sobre dos cláusulas: $O(n^2)$
  - Encontrar dos cláusulas a resolver: $O(n)$
  - Tiempo total del algoritmo: $O(3^n n^2)$

Estrategia Primero en Profundidad

- Almacenamos la demostración como una lista de cláusulas .
- En cada paso resolvemos usando el resolvente generado en el paso anterior y una cláusula de la lista.
- Es necesario incluir mecanismo para detectar loops.
- Usualmente se pone una cota de profundidad.

Ejemplo: Primero en Profundidad



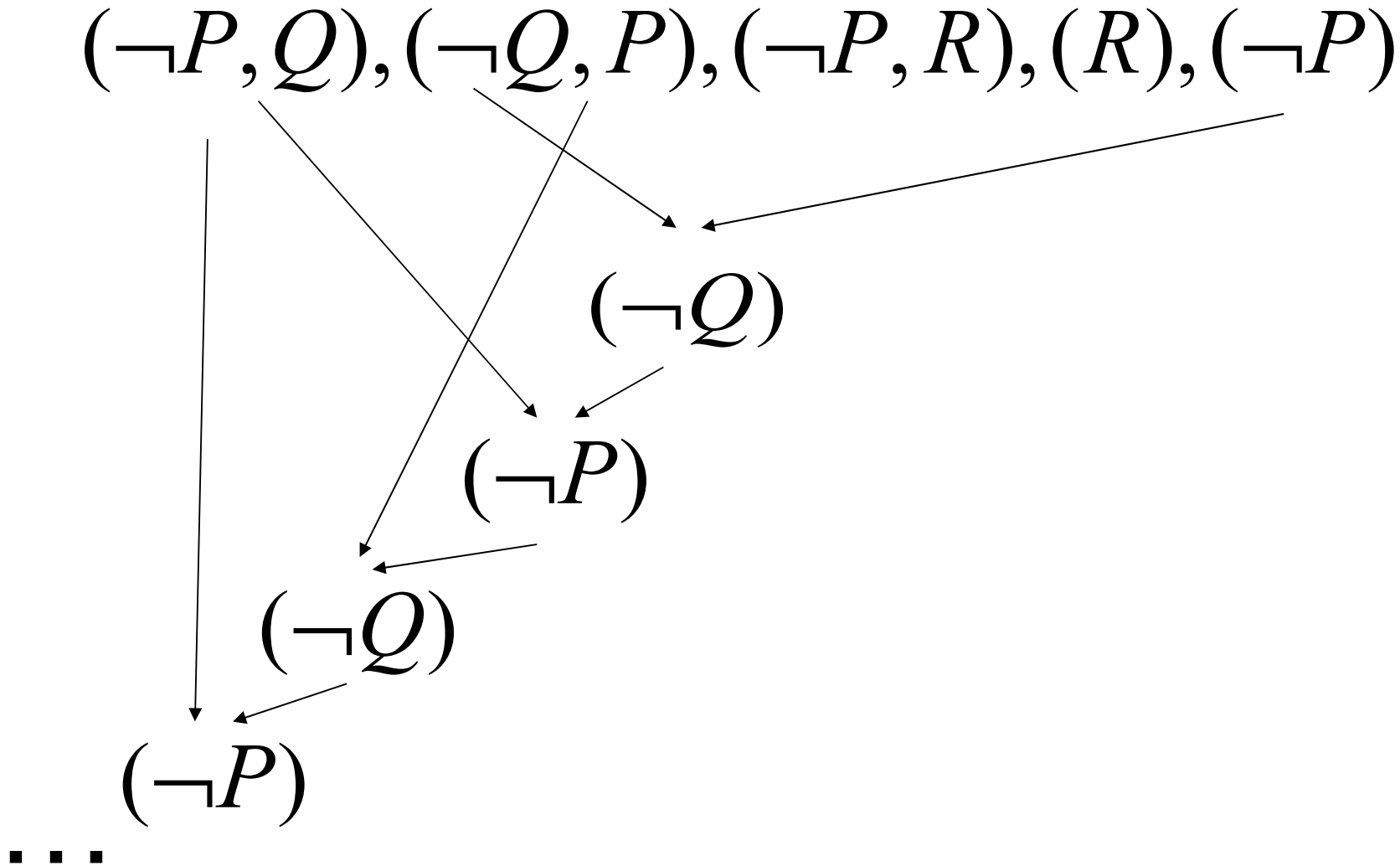
Estrategia Primero en Profundidad (sin detección de loops)

- Resp:=Falso
- Para toda c en KB
  - RPP(<c>);
- return(Resp)

RPP(L: lista de resolventes)

- Sea c cláusula inicial de L
- Para toda cláusula c' en KB o en L que resuelva con c
  - c'':=Res(c',c)
  - Si (c''==())
    - Resp:=Verdadero
    - return() // aborta todos los llamados a RPP
  - Else
    - agregar c'' al inicio de L
    - RPP(L,KB)

Ejemplo: Loop



Primero en Profundidad vs. Primero en Anchura

- Ambas estrategias son completas
- Ventajas Primero en Profundidad:
  - Requiere menos memoria: sólo almacenamos resoluciones en lista L
- Desventaja Primero en Profundidad:
  - Podemos generar listas (o demostraciones parciales) muy largas.
  - Podemos llegar varias veces a la misma cláusula (redundancia).

Ejercicio

Encuentre una refutación para las siguientes cláusulas usando las dos estrategias de ordenación vistas.

$$A \vee C$$

$$A \vee \neg C \vee F$$

$$\neg A \vee E \vee F$$

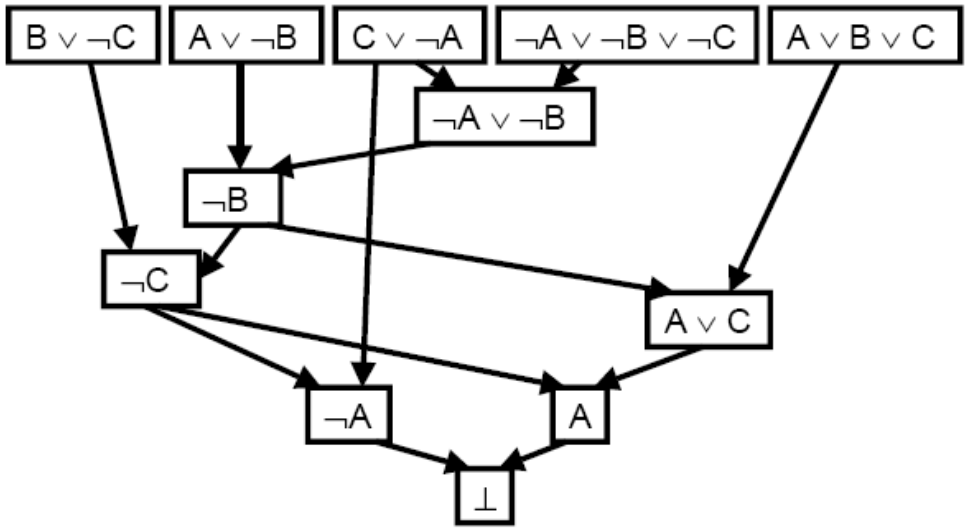
$$\neg C \vee \neg E \vee F$$

$$\neg A \vee \neg F$$

$$C \vee F$$

Eliminación de Cláusulas

- En general no podemos eliminar cláusulas generadas ya que pueden ser usadas más adelante.
- Es decir, el grafo generado no siempre es un árbol: Ejemplo:



Eliminación de Cláusulas

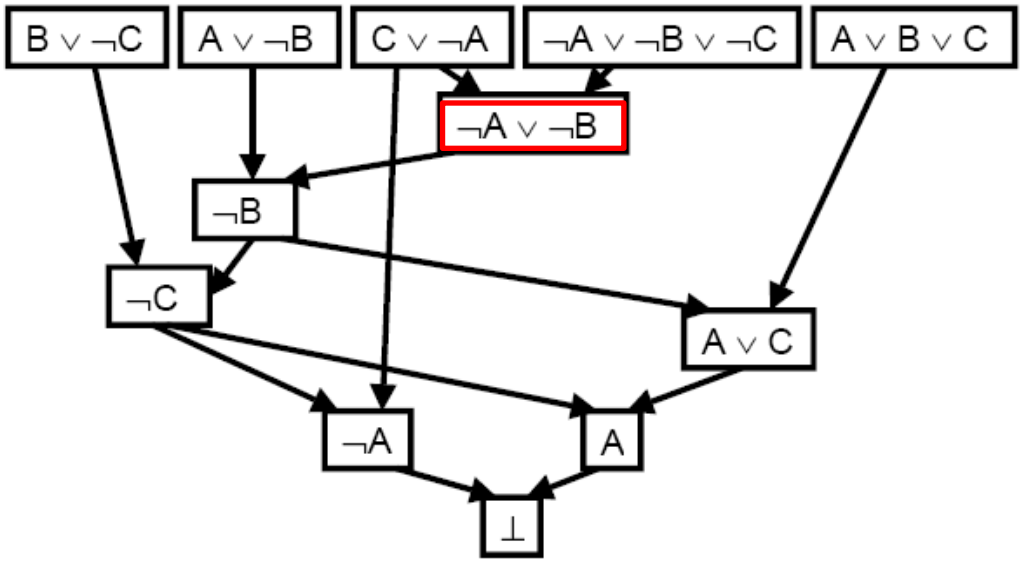
- Recordar que vemos las cláusulas como conjunto de literales
- Dadas dos cláusulas $C1$, $C2$ tal que $C1 \subseteq C2$, es fácil verificar que $C1 \models C2$
- Esto debido a que $Mod(C1) \subseteq Mod(C2)$

Eliminación de Cláusulas

- Dado un conjunto de cláusulas S, sea Base(S) el conjunto de cláusulas en S minimales (con respecto a inclusión).
- Ejemplo, si S={{a,b}, {¬c,b,a}, {b,c},{b,c,d}}, Base(S)={{a,b}, {b,c}},
- Es fácil verificar que Mod(S)=Mod(Base(S)).
- Luego Base(S) y S son lógicamente equivalentes y por lo tanto podemos aplicar el proceso de resolución sobre Base(S) en lugar de S.

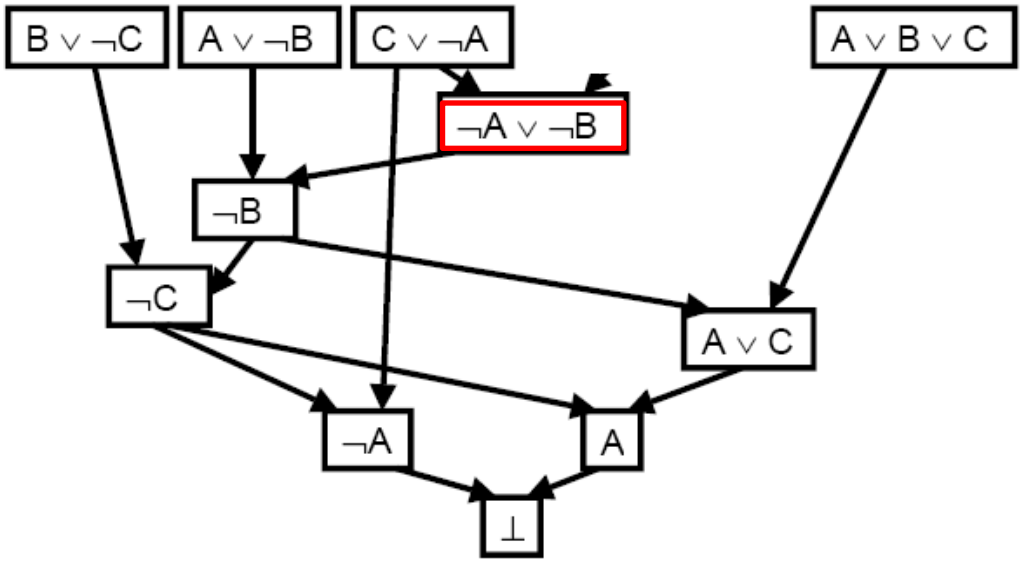
Eliminación de Cláusulas

- El resultado del proceso de resolución no cambia si eliminamos cláusulas no minimales. Ejemplo:



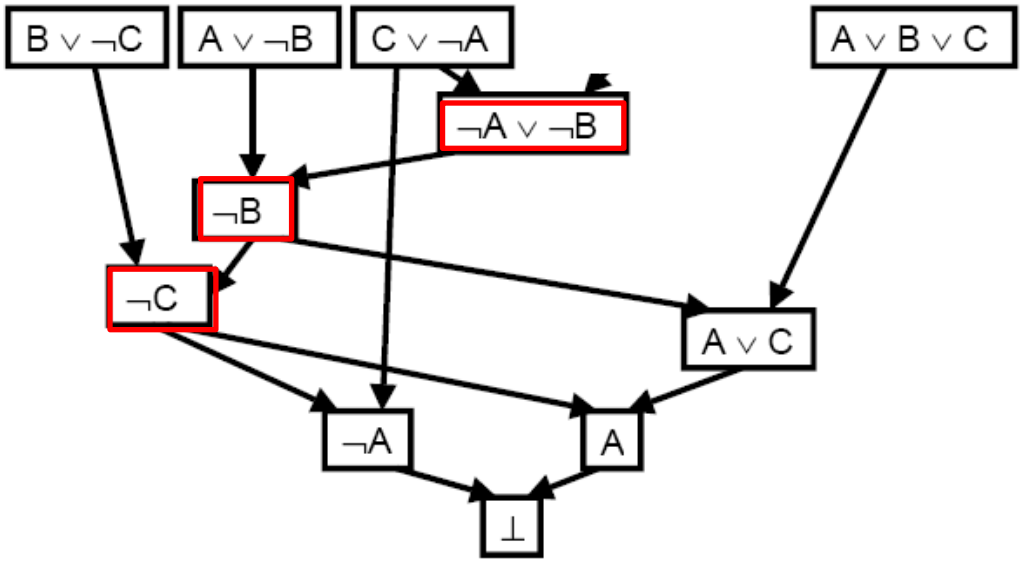
Eliminación de Cláusulas

- El resultado del proceso de resolución no cambia si eliminamos cláusulas no minimales. Ejemplo:



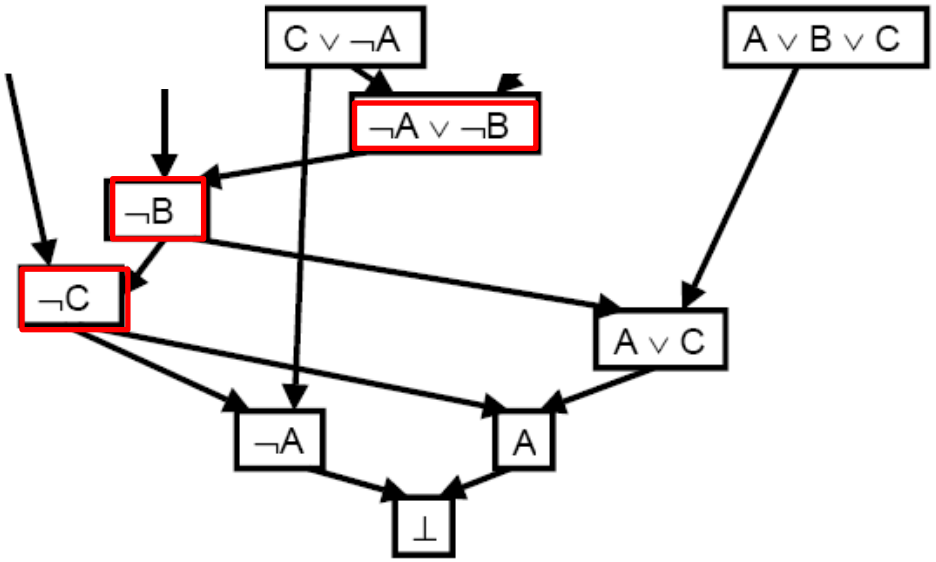
Eliminación de Cláusulas

- El resultado del proceso de resolución no cambia si eliminamos cláusulas no minimales. Ejemplo:



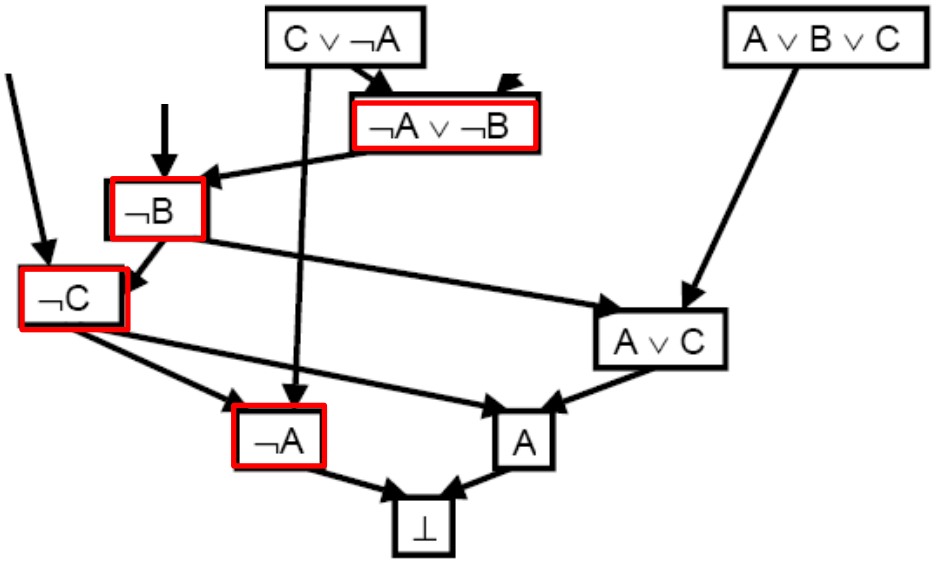
Eliminación de Cláusulas

- El resultado del proceso de resolución no cambia si eliminamos cláusulas no minimales. Ejemplo:



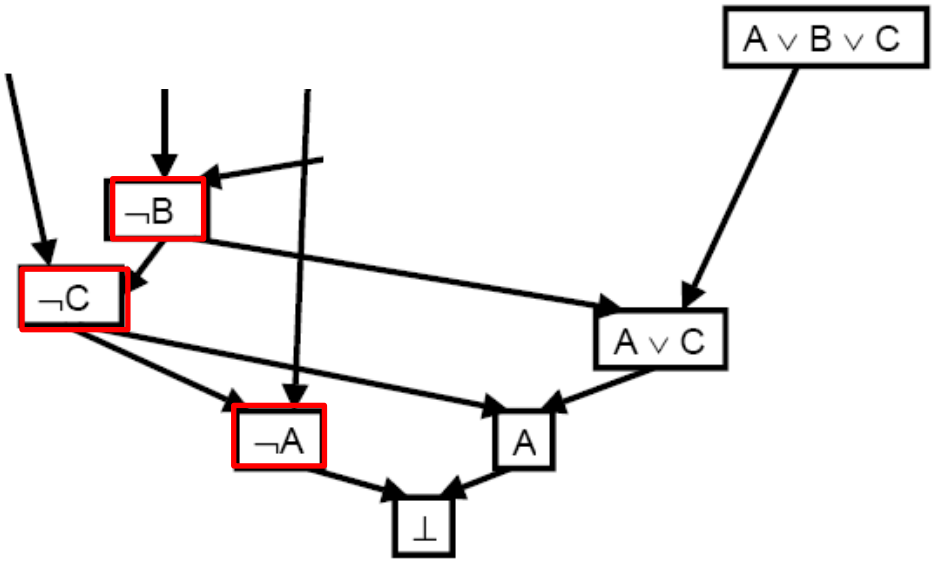
Eliminación de Cláusulas

- El resultado del proceso de resolución no cambia si eliminamos cláusulas no minimales. Ejemplo:



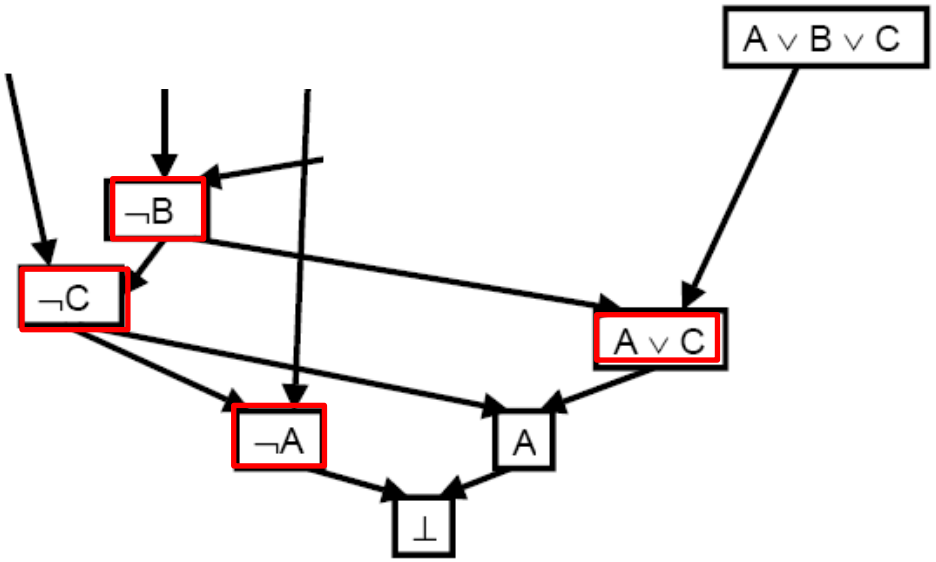
Eliminación de Cláusulas

- El resultado del proceso de resolución no cambia si eliminamos cláusulas no minimales. Ejemplo:



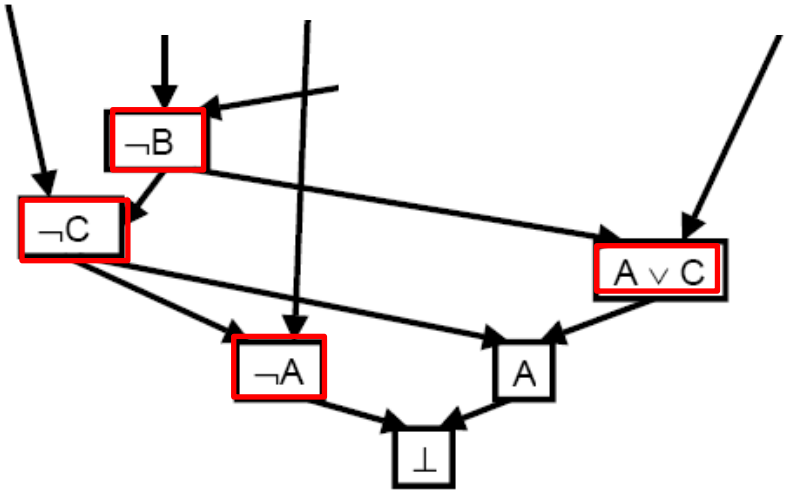
Eliminación de Cláusulas

- El resultado del proceso de resolución no cambia si eliminamos cláusulas no minimales. Ejemplo:



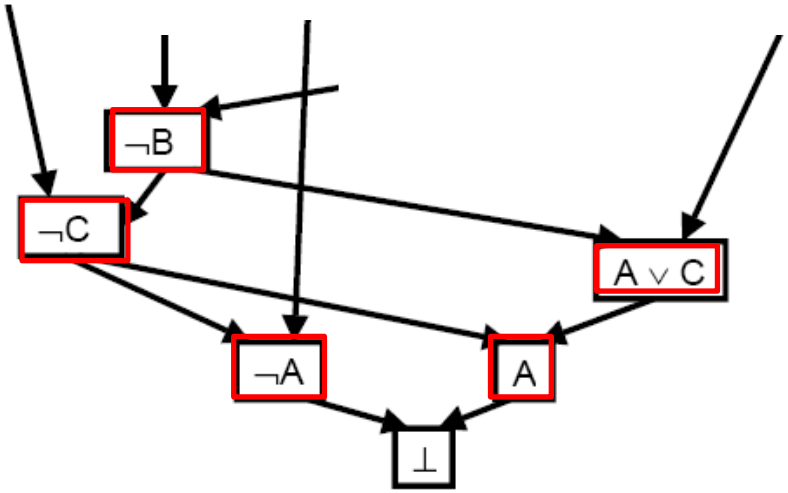
Eliminación de Cláusulas

- El resultado del proceso de resolución no cambia si eliminamos cláusulas no minimales. Ejemplo:



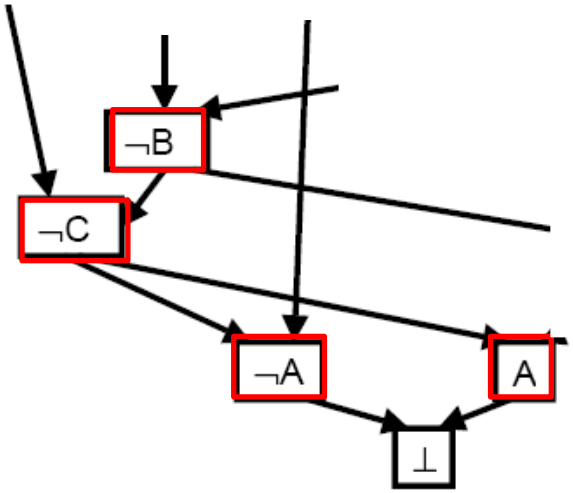
Eliminación de Cláusulas

- El resultado del proceso de resolución no cambia si eliminamos cláusulas no minimales. Ejemplo:



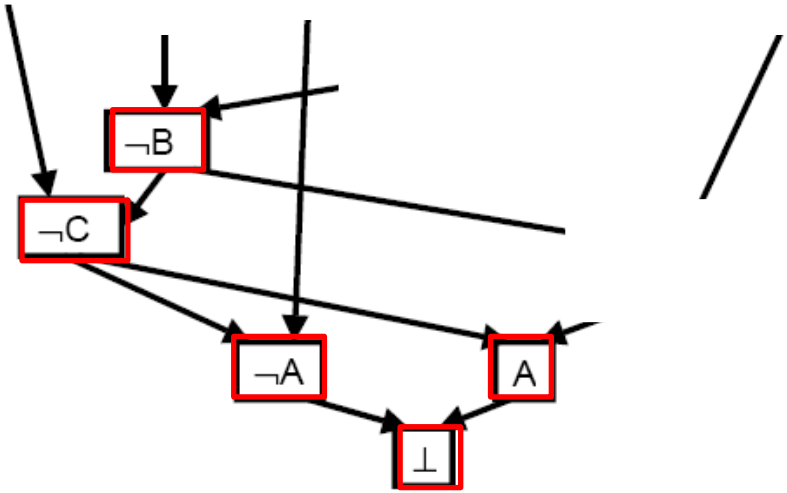
Eliminación de Cláusulas

- El resultado del proceso de resolución no cambia si eliminamos cláusulas no minimales. Ejemplo:



Eliminación de Cláusulas

- El resultado del proceso de resolución no cambia si eliminamos cláusulas no minimales. Ejemplo:



Estrategias de refinamiento

- Imponen restricciones en las resoluciones a realizar.
  - Conjunto Soporte
  - Resolución Lineal
  - Resolución Unitaria
- Podan el espacio de posibles resoluciones

Conjunto Soporte

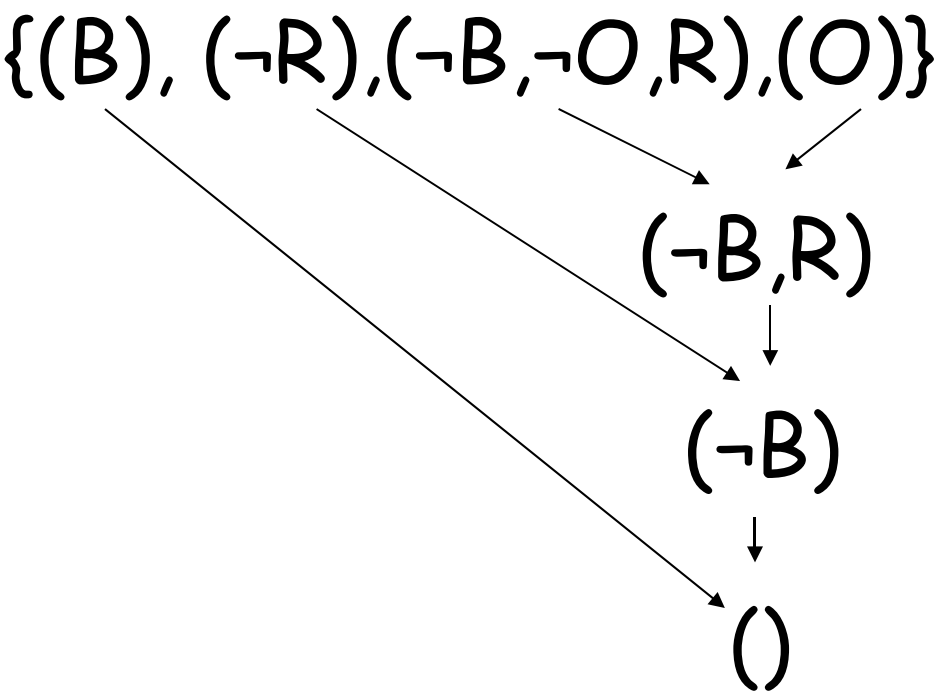
Debemos verificar si KB$\models \alpha$:

- Si KB es satisfacible entonces toda refutación para (KB $\wedge \neg \alpha$) debe contener alguna cláusula de $\neg\alpha$
- Idea: guiar la búsqueda de la cláusula vacía usando ($\neg\alpha$).

- Conjunto soporte:
  - Una cláusula U es descendiente de una cláusula W si (i) U es resolvente de W y otra clásula o (ii) U es resolvente de una cláusula descendiente de W y otra cláusula
  - Conjunto soporte: cláusulas en ($\neg\alpha$) o descendientes de cláusulas en ($\neg\alpha$).
- Estrategia Conjunto Soporte: al resolver tomar en cuenta al menos una cláusula en conjunto soporte.

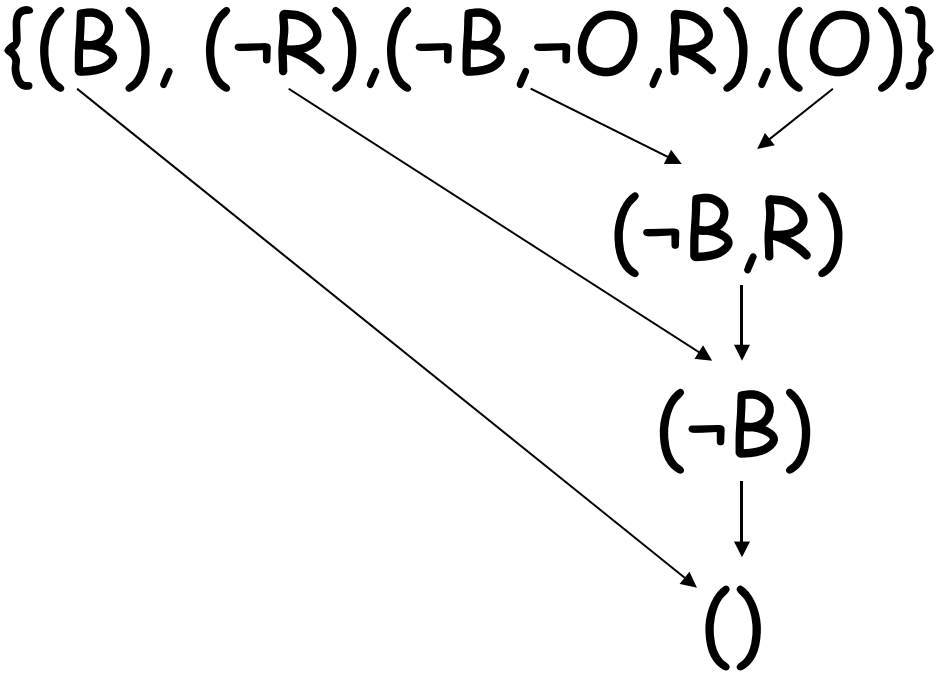
Conjunto Soporte: Ejemplo

Sea KB= {(B), (¬R),(¬B,¬O,R)}; y α = ¬O.



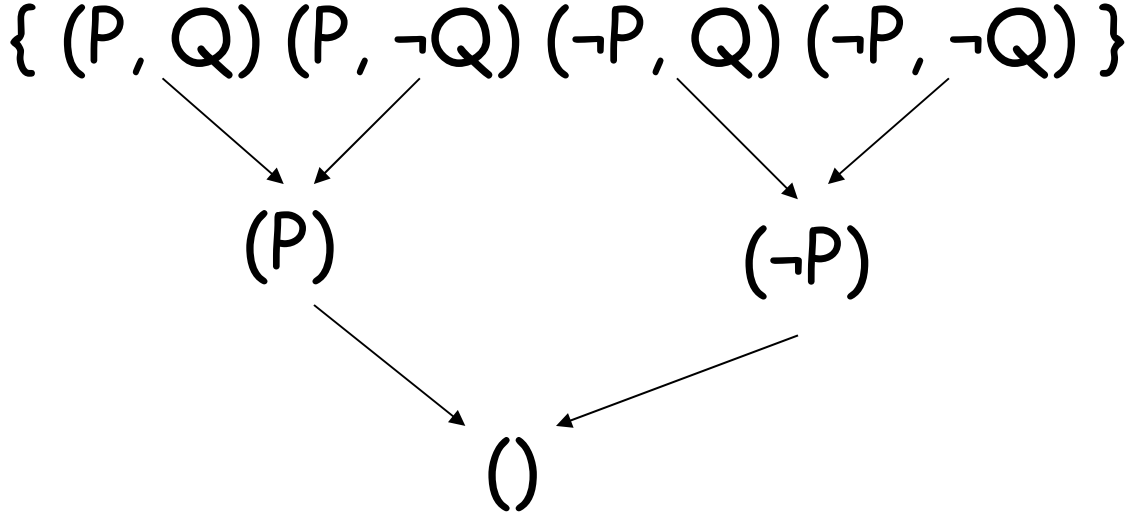
Resolución Lineal

- Sólo permite resoluciones que usan al menos una cláusula en $(KB \wedge \neg\alpha)$
- Ejemplo anterior es resolución lineal:



Resolución Lineal

- Problema: refutación basada en resolución lineal no es completa



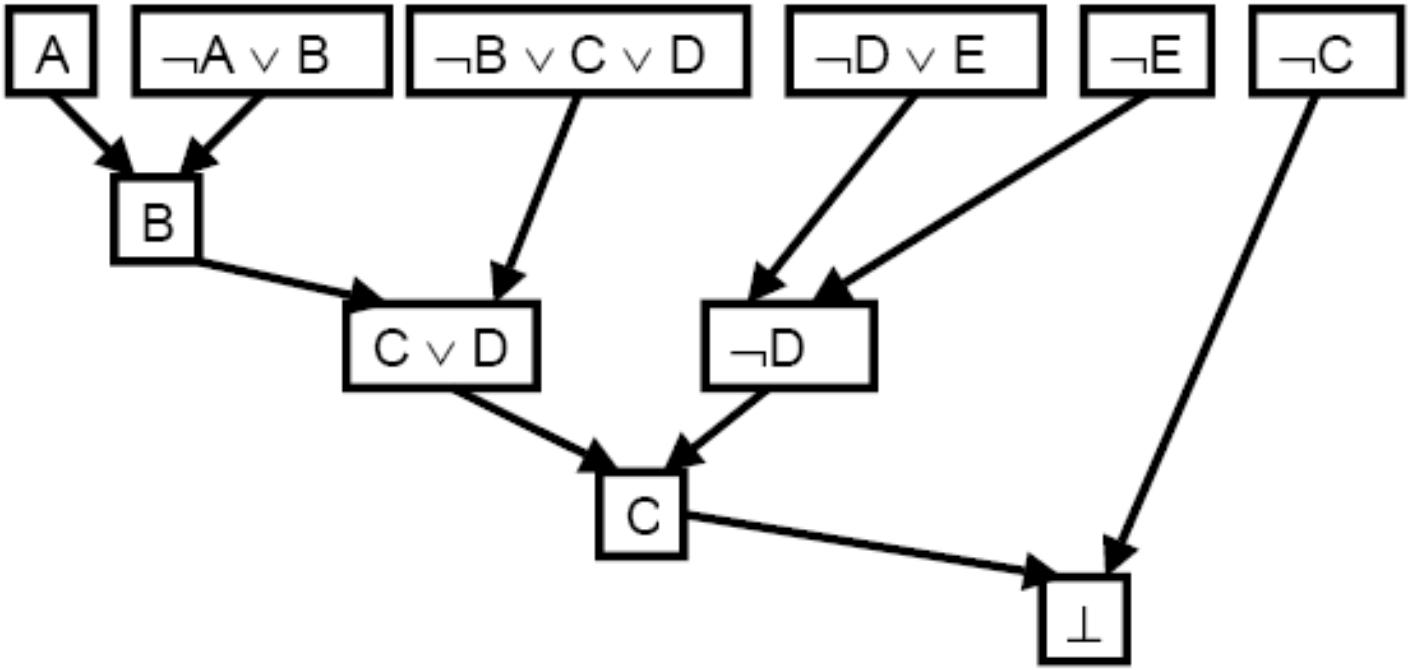
Ejercicio: complejidad de resolución lineal

- Resolución lineal es más eficiente que resolución.
- Investigar la complejidad de resolución lineal (en muchos artículos resolución lineal se denomina "input resolution").

Resolución Unitaria

- Sólo permite resoluciones donde al menos una de las cláusulas tiene un literal (es unitaria)
- En este caso decimos que la refutación (si la hay) es unitaria

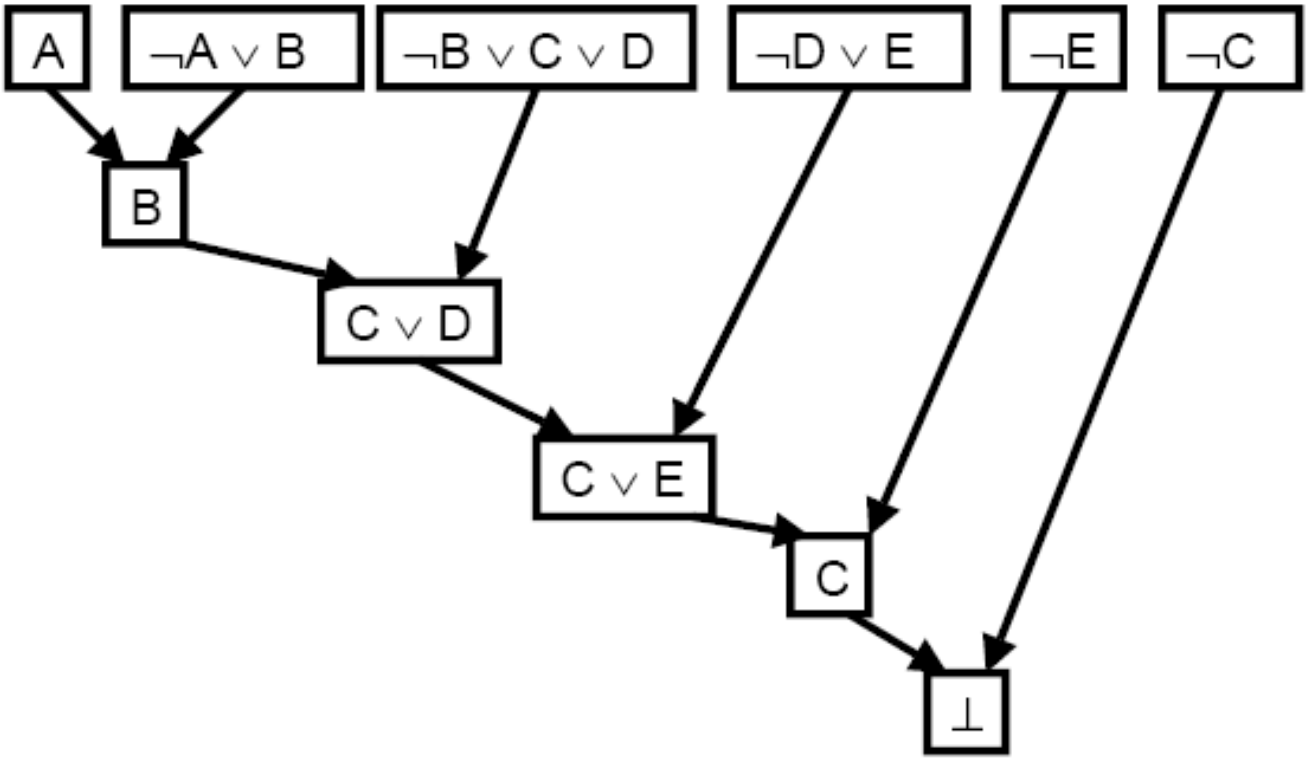
Ejemplo: Resolución Unitaria



Resolución Unitaria vs. Resolución Lineal

- Teorema: Un conjunto de cláusulas tiene una refutación unitaria ssi tiene una refutación lineal
- Por lo tanto refutación basada en resolución unitaria no es completa

Ejemplo: Refutación Lineal para el ejemplo anterior



Complejidad de Resolución Unitaria

- En una resolución C1,U→C3, dode U es la cláusula unitaria, siempre podemos eliminar C1 del conjunto de cláusulas (¿Por qué?)
- Dado un conjunto de cláusulas S, sea Largo(S) su número de literales (incluyendo repetidos).
- Ejemplo, si S es { (P, Q) (P, ¬Q) (¬P, Q) (¬P, ¬Q) }, Largo(S)=8.
- Luego en un proceso de resolución unitaria + eliminación de cláusulas no tenemos más de Largo(S) resoluciones.
- Recordar que en un proceso de resolución podemos generar del orden de $2^{Largo(S)}$ resoluciones.

Consecuencia Lógica en Cálculo Proposicional: Complejidad

- El mejor algoritmo conocido para determinar satisfacibilidad (i.e., implicancia lógica) de una sentencia proposicional tiene complejidad peor-caso $\Theta(2^n)$, donde n es el tamaño de la sentencia.
  - Ejemplo, Refutación Mediante Resolución.
- Esto debido a que Satisfabilidad de sentencias arbitrarias y CNF es NP-completo (también denominados "problemas intratables").

Consecuencia Lógica en Cálculo Proposicional: Complejidad

- Verificar validez (tautología) en CNF requiere tiempo lineal en n.
- Es decir para decidir si $KB \models \alpha$:
  - Transformar $\neg(KB \wedge \neg\alpha)$ en CNF
  - Verificar si la versión CNF es válida
- La siguiente tabla muestra que esto no ayuda significativamente:

	CNF	DNF	General
Satisfiability	NPC	$\Theta(n)$	NPC
Tautology	$\Theta(n)$	NPC	NPC
Conversion of an arbitrary wff to ..	$\Theta(2^n)$	$\Theta(2^n)$	–

Transformación en CNF

Pasos:

2. Eliminar $\Leftrightarrow$
3. Eliminar $\Rightarrow$
4. Usar leyes de DeMorgan para mover los ¬'s a los átomos
5. Eliminar doble negaciones
6. Distribuir $\vee$ sobre $\wedge$

Ejemplo

$$\neg((A_1 \equiv A_2) \wedge \neg(A_3 \vee \neg\ (A_4 \rightarrow A_1)))$$

Pasos 1 y 2:

$$\neg(((A_1 \wedge A_2) \vee (\neg A_1 \wedge \neg A_2))) \wedge \neg(A_3 \vee \neg(\neg A_4 \vee A_1)))$$

Pasos 3 y 4:

$$\neg((A_1 \wedge A_2) \vee (\neg A_1 \wedge \neg A_2)) \vee (\neg A_3 \wedge (\neg A_4 \vee A_1))$$

$$(\neg(A_1 \wedge A_2) \wedge \neg(\neg A_1 \wedge \neg A_2)) \vee (A_3 \vee\ (A_4 \wedge \neg A_1))$$

$$((\neg A_1 \vee \neg A_2) \wedge (A_1 \vee A_2) \vee (A_3 \vee\ (A_4 \wedge \neg A_1))$$

Ejemplo (cont.)

$$((\neg A_1 \vee \neg A_2) \wedge (A_1 \vee A_2) \vee (A_3 \vee \ (A_4 \wedge \neg A_1))$$

Paso 5:

$$((\neg A_1 \vee \neg A_2) \wedge (A_1 \vee A_2)) \vee ((A_3 \vee A_4) \wedge (A_3 \vee \neg A_1))$$

$$(\neg A_1 \vee \neg A_2 \vee A_3 \vee A_4) \wedge (\neg A_1 \vee \neg A_2 \vee A_3 \vee \neg A_1) \wedge$$
$$(A_1 \vee A_2 \vee A_3 \vee A_4) \wedge (A_1 \vee A_2 \vee A_3 \vee \neg A_1)$$

Simplificando:

$$(\neg A_1 \vee \neg A_2 \vee A_3 \vee A_4) \wedge (\neg A_1 \vee \neg A_2 \vee A_3) \wedge$$
$$(A_1 \vee A_2 \vee A_3 \vee A_4)$$

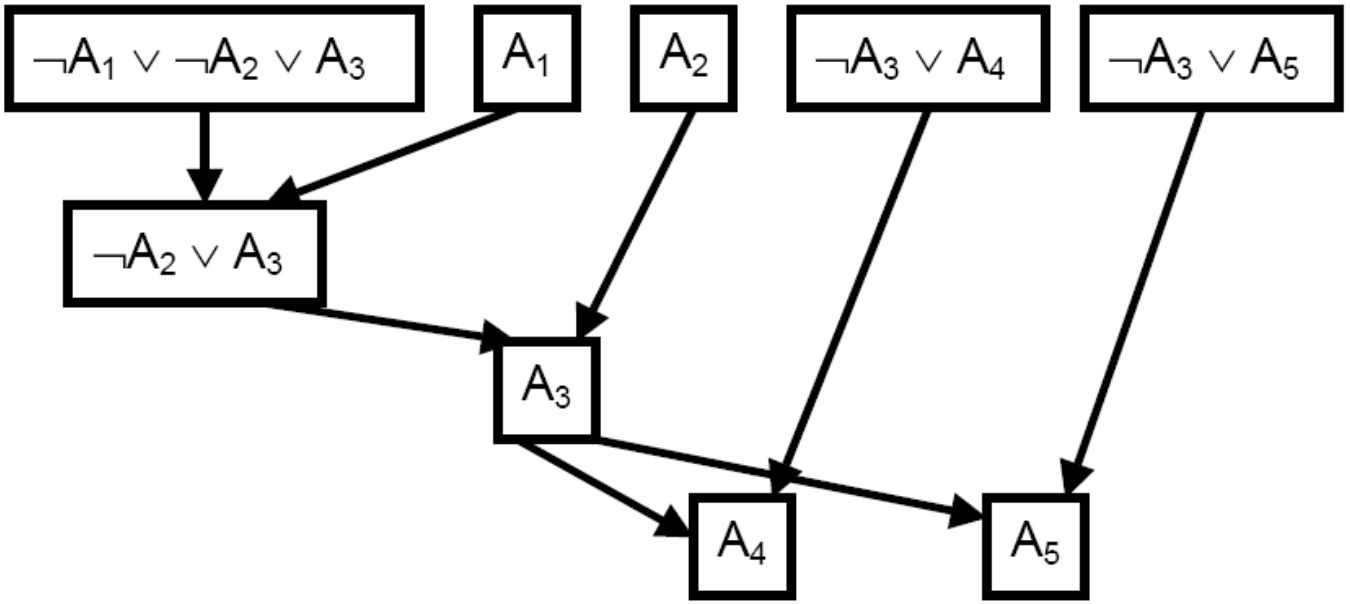
Complejidad de Implicancia Lógica en Cálculo Proposicional

- Sin embargo NP-completo es una caracterización para el peor de los casos
- Es posible enfocarse en fragmentos de Cálculo Proposicional donde el problema sea tratable.
- Tal vez el más importante de estos fragmentos es el de las Cláusulas de Horn.

Cláusulas de Horn

- Cláusulas no más un literal positivo
- Ejemplos: (P), $(\neg P \vee Q)$, $(\neg P \vee \neg Q)$
- Verificar consecuencia lógica para cláusulas de Horn toma tiempo polinomial:
  - Resolución lineal y Unitaria son completas para cláusulas de Horn
  - Algoritmo BUPP (Bottom-Up Proof Procedure).

Cláusulas de Horn y Resolución Unitaria



Teorema: si un conjunto de cláusulas de Horn no es satisfacible entonces existe una refutación unitaria positiva.

BUPP

- Dada KB, computa el conjunto de literales positivos derivables de KB
- Es decir, BUPP(KB) contienete literales positivos P tal que: KB $\models$ P
- BUPP sirve para verificar KB $\models \alpha$ donde α es de la forma:

$$(P1 \wedge P2 \wedge \ldots \wedge Pn)$$

- La respuesta es positiva ssi

{P1,P2, …,Pn} está en BUPP(KB)

BUPP: Computa el conjunto de literales positivos derivables de un conjunto de CHs

- Call a clause selectable if its body is a logical consequence of *KB* (its body is provable)
  - so adding it's head is justified (head is provable)

1. Let C = $\emptyset$ (C is the set of logical consequences)

2. Until no clause is selectable

   (a) Select a clause $h \leftarrow p_1 \,\&\, \ldots \,\&\, p_k$ such that $p_1 \in C, \ldots, p_k \in C$ and $h \notin C$

   (b) Add h to C

Algoritmo BUPP: Ejemplo

KB: (1) a <- b & c.
(2) b <- d & e.
(2') b <- c.
(3) b <- g & e.
(4) c <- e.
(5) d.
(6) e.
(7) f <- a & g.

Consequences (selecting first selectable clause in order they occur in KB)

$C = \emptyset$
$C = \{d\}$ (clause 5)
$C = \{d, e\}$ (6)
$C = \{d,e,b\}$ (2)
$C = \{d,e,b,c\}$ (4)
$C = \{d,e,b,c,a\}$ (1)

- No more clauses selectable
- g not provable (not in any head)
- (3), (7) could never be selected
- (2') could have been selected if used different order, but would not have changed C.

Notas sobre BUPP

- El orden en que seleccionamos cláusulas no importa
  - Ejercicio: explique por qué.
- Complejidad:
  - Cada regla se selecciona a lo más una vez
    - No hay más de $|KB|$ iteraciones
  - En cada iteración buscamos una regla seleccionable $O(|KB|)$
    - Cada iteración toma
  - Tiempo total $O(|KB|^2)$

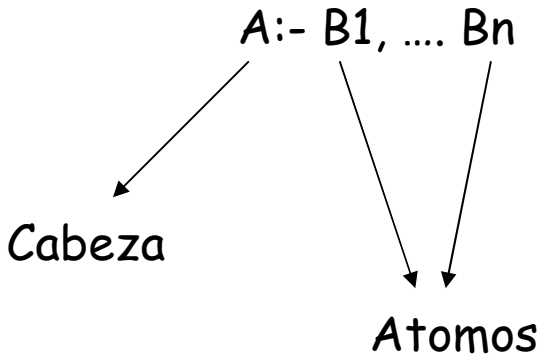
Programa en Prolog(Prop)

- Base de Conocimiento (KB):
  - Hechos:
    - Cláusulas con un único literal positivo
    - Ejemplo: "P:-" o "P" representa la cláusula (P)
  - Reglas
    - Cláusulas con un literal positivo y más de un literal negativo
    - Ejemplo: "Q:-P" representa la cláusula ($\neg P, Q$).
- Objetivo o consulta (α):
    - Conjunción de literales positivos
    - Al negar el objetivo queda un cláusula con literales negativos
    - Ejemplo: ":-P,Q" o tb. "?P,Q" representa: ($P \wedge Q$) y al negarlo queda como la cláusula ($\neg P, \neg Q$).
- Luego en Prolog(Prop) ($KB \wedge \neg\alpha$) es un conjunto de cláusulas de Horn.

Prolog y Resolución Lineal

- Teorema de Nerode & Shore:
  - Refutación mediante resolución lineal es completa para programas Prolog(Prop).
- Prolog implementa una variante de resolución lineal llamada resolución SDL.
- A diferencia de BUPP, SDL:
  - Dirige la construcción de la refutación usando el objetivo ("subgoaling")
    - Estrategia "backward" en lugar de "forward"
  - Se generaliza bien a Prolog con parámetros funciones y variables.

Reglas en Prolog



Resolución SLD

- Dada un objetivo *:-Q1,Q2 (1)*
- Supongamos que tenemos la regla
  Q1:-A,B,C (2)
- Reemplazamos en (1) Q1 por el cuerpo de (2) obteniendo :-A,B,C,Q2 (3)
- Esto es equivalente a resolver ¬(1) con (2)
- Se repite esta operación hasta obtener ":-"
- Prolog implemante SLD + estrategia primero en profundidad

Resolución SDL: Algoritmo

SDLRes (O: Objetivo)

- Para todo átomo Ai de O
  - Para toda cláusula Ai:-B1, …, Bn de KB
    - Sea O' el resultado de reemplazar Ai por B1,… ,Bn en O
    - Si O' es ":-" retornar (true)
    - SDLRes(O')
- retornar(falso)

Ejemplo

KB: (1) a <- b & c.
(2) b <- d & e.
(2') b <- c.
(3) b <- g & e.
(4) c <- e.
(5) d.
(6) e.
(7) f <- a & g.

Query: ?a

Derivation Attempt #1

<- a.	
<- b & c.	Select a; choose (1)
<- g & e & c.	Select b; choose (3)
<- g & c.	Select e; choose (6)
<- g & e.	Select c; choose (4)
<- g.	Select e; choose (6)

Select g: FAIL! no matching clause

Ejemplo (cont.)

KB: (1) a <- b & c.
(2) b <- d & e.
(2') b <- c.
(3) b <- g & e.
(4) c <- e.
(5) d.
(6) e.
(7) f <- a & g.

Query: ?a

Derivation Attempt #2

<- a.	
<- b & c.	Select a; choose (1)
<- d & e & c.	Select b; choose (2)
<- e & c.	Select d; choose (5)
<- c.	Select e; choose (6)
<- e.	Select c; choose (4)
<- .	Select e; choose (6)

QUERY IS TRUE: obtained answer

Resolución SLD

- ¿Importa la elección del átomo usado para resolver?
  - No, todos los átomos deben ser "resueltos"
- ¿Importa la elección de la regla para resolver?
  - Sí: malas elecciones pueden llevar a fracasos y backtracking.

Resolución SDL

- **S**election rule:
  - Siempre se resuelve el primer literal del objetivo parcial
- **L**inear Resolution:
  - Produce resoluciones lineales
- **D**efinite Clauses:
  - Cláusulas de Prolog(Prop)
- Problema:
  - No necesariamente termina (Prolog es incompleto)
  - Esto debido a potenciales loops en el programa
- Resolución SDL se generaliza naturalmente a Prolog (con variables, funciones y parámetros).

Loops en Prolog(Prop)

- Prolog(prop) puede entrar en loops infinitos.
- Prolog no necesariamente para (aunque exista una refutación).
- Esto debido a que implementa estrategia de primero en profundidad
- Ejemplo:

```
p :- q.
q :- p.
p :- r.
   r.
 :-p ?
```