Cálculo de Predicados (3)

Carlos Hurtado L.

Depto de Ciencias de la Computación, Universidad de Chile

Clase Pasada: Regla Generalizada de Resolución

Dadas dos cláusulas

(L,Q1, ..., Qn) y (-M,R1,...,Rm)

tales que L y M tienen un UMG σ, obtenemos el resolvente:

($Q1\sigma$, ..., $Qn\sigma$,$R1\sigma$,...,$Rm\sigma$)

Clase Pasada: Unificador más General

- Un unificador σ para A Y B es el unificador más general (UMG) ssi para todo unificador φ para A y B existe una sustitución π tal que

$$\varphi = \sigma\pi.$$

Clase Pasada: Ejemplo: Regla Generalizada de Resolución

Dadas las cláusulas:

(P(x),Q(g(x))) y (-P(a),R(a),Q(z))

En este caso L=P(X), M=P(a) y su UMG es {x/a}. Luego obtenemos el resolvente:

(Q(g(a)),R(a),Q(z))

Clase Pasada: Ejemplo: Mundo de los Bloques

- Sea KB={on(b,c),on(c,d),$clear$(e), $\forall$ x,y,z.($on(x, y) \wedge on(y, z) \Rightarrow above(x, z)$)}
- KB $\models$ $above(b,d)$?
- Transformamos KB $\wedge$ -$above(b,d)$ a conjunto de cláusulas:

{(on(b,c)),(on(c,d)),($clear$(e)),

($-on(x, y), -on(y, z), above(x,z)$),

$-above(b,d)$}.

Ejemplo: Mundo de los Bloques

{(*on*(b,c)),(*on*(c,d)),(*clear*(e)),(-*on*(*x*, *y*),-*on*(*y*, *z*),*above*(*x*, *z*)),(-*above*(*b*,*d*))}.

(-*on*(*c*,*z*),*above*(*b*,*z*)) {x/b,y/c}

Ejemplo: Mundo de los Bloques

{(*on*(b,c)),(*on*(c,d)),(*clear*(e)),(-*on*(*x*, *y*),-*on*(*y*, *z*),*above*(*x*, *z*)),(-*above*(*b*,*d*))}.

(-*on*(*c*,*z*),*above*(*b*,*z*)) {x/b,y/c}

above(*b*,*d*)){z/d}

Ejemplo: Mundo de los Bloques

{(*on*(b,c)),(*on*(c,d)),(*clear*(e)),(-*on*(*x*, *y*),-*on*(*y*, *z*),*above*(*x*, *z*)),(-*above*(*b*,*d*))}.

(-*on*(*c*,*z*),*above*(*b*,*z*)) {x/b,y/c}

above(*b*,*d*)){z/d}

()

Resolución y UMG

- Para Lógica de Predicados la aplicación de la regla de resolución involucra calcular el UMG de dos átomos.
- Algoritmo UNIFICAR: entrega el UMG de dos átomos.

Algoritmo UNIFICAR

- Primero transformamos los a'tomos a listas en notacio'n infija

Ejemplo: A:=P(x,f(a,y)) y B:=P(x,f(z,b))

A y B transforman en:

Ejemplo: A:=(P x (f a y)) B:= (P x (f z b))

Algoritmo UNIFICAR

Ejemplo: A:=P(x,f(a,y)) y B:=P(x,f(z,b))

A y B transforman en listas (notación infija).

Ejemplo: A:=(P x (f a y)) B:= (P x (f z b))

UNIFICAR(A,B)

- Inicialmente σ:={}

5. Si A y B son expresiones compuestas
   - Para cada par de elementos Ai de A y Bi de B:
     - φ:=UNIFICAR(Ai,Bi)
     - A:=Aφ, B:=Bφ, σ:=σφ
6. En caso contrario:
   - Si A={}, σ:={}
   - Si A (resp. B) es una constante y no se puede unificar con B (resp. A), entonces retornamos fallo (no hay unificador).
   - Si A (resp. B) es una variable que no aparece en B (resp. A), entonces σ:={A/B} (resp. σ:={B/A})

- Retornar(σ)

Ejemplo 1

A:=P(x,f(a,y)) y B:=P(x,f(z,b))

- mgu:=UNIFICAR((P x (f a y)),(P x (f z b)))
  - φ:=UNIFICAR(P,P)
  - σ:={}{}
  - φ:=UNIFICAR(x,x)
  - σ:={}{}
  - φ:=UNIFICAR((f a y),(f z b))
    - φ:=UNIFICAR(f,f)
    - σ:={}{}
    - φ:=UNIFICAR(a,z)
    - σ:={}{z/a}
    - φ:=UNIFICAR(y,b)
    - σ:={z/a}{y/b}
  - σ:={}{z/a,y/b}
- mgu:={z/a,y/b}

Ejemplo 2

A:=P(f(y),y) B:=P(f(u),v)

- mgu:=UNIFICAR((P (f y) y),(P (f u) v))
  - φ:=UNIFICAR(P,P)
  - σ:={}{}
  - φ:=UNIFICAR((f y),(f u))
  - σ:={}{y/u}
  - A:=(P (f u) u) B:= (P (f u) v)
  - φ:=UNIFICAR(u,v)
  - σ:={y/u}{u/v}={y/v,u/v}
- mgu:={y/v,u/v}

Ejercicios

- UNIFICAR(P(x),P(a))= {x/a}
- UNIFICAR(P(f(x),y,g(y)),P(f(x),z,g(x))) ={y/x,z/x}
- UNIFICAR(P(f(x,g(a)),g(a)),P(f(x,z),z)) ={z/a}

Verificación de Ocurrencia

Ejemplo: A:=P(x,f(a,y)) y B:=P(x,f(z,b))

A:=(P x (f a y)) B:= (P x (f z b))

UNIFICAR(A,B)

- Inicialmente σ:={}

4. Si A y B son expresiones compuestas
   - Para cada par de subexpresiones Ai de A y Bi de B:
     - φ:=UNIFICAR(Ai,Bi)
     - A:=Aφ, B:=Bφ, σ:=σφ
5. En caso contrario:
   - Si A=, σ:={}
   - Si A (resp. B) es una constante y no se puede unificar con B (resp. A), entonces retornamos fallo (no hay unificador).
   - Si A (resp. B) es una variable que no aparece en B (resp. A), entonces σ:={A/B} (resp. σ:={B/A})

- Retornar(σ)

Verificación de Ocurrencia

Sin verificación de ocurrencia:

A:=P(x,x) B:=P(f(z),z)

- mgu:=UNIFICAR((P x x),(P (f z) z))
  - φ:=UNIFICAR(P,P)
  - σ:={}
  - φ:=UNIFICAR(x,(f z))
  - σ:={x/f(z)}
  - A:=(P (f z) (f z)) B:= (P (f z) z)
  - φ:=UNIFICAR((f z),z)
  - σ:={x/f(z)}{z/f(z)}
  - σ:={x/f(f(z)),z/f(z)}
  - A:=(P (f z) (f (f z))) B:= (P (f z) (f z))
- Luego UNIFICAR(A,B) retorna {x/f(f(z)), z/f(z)} que no es un unificador para A y B.

Obtención de Respuestas

- Hasta el momento hemos usado resolución para verificar si:

$$KB \models \alpha$$

- También podemos usar resolución para encontrar las constantes que satisfacen una cierta condición que se infiere de KB.
- Ejemplo: Calcular el conjunto

$$\{x \mid KB \models padre(x,juan)\}$$

- Esto es similar a consultar una base de datos.

Obtención de Respuestas

- Dado un átomo α cuyas variables libres son x1, …,xn
- Queremos encontrar el conjunto de tuplas

$$\{x1, \ldots, xn \mid KB \models \alpha\}$$

Obtención de Respuestas

- Dado un átomo α cuyas variables libres son x1, ...,xn
- Queremos encontrar el conjunto de tuplas

$$\{x1, \ldots, xn \mid KB \models \alpha\}$$

- Agregamos a la cláusula -α el átomo resp(x1,...,xn).
- Desde (KB ∧ -α), vía resolución intentamos encontrar una cláusulas sólo con átomos resp(t1,...,tn).
- Cada una de estas cláusulas se puede ver como una posible respuesta.

Ejemplo 1

- Sea KB:

  {(father(art,jon),father(bob,jon))

  (father(bob,kim))

  (-father(y,z),parent(y,z))}

- Queremos responder la siguiente consulta:

  {x | parent(x,kim)}

Ejemplo 1 (cont.)

- 1: (father(art,jon),father(bob,jon))
- 2: (father(bob,kim))
- 3: (-father(y,z),parent(y,z))
- 4: -α = (-parent(x,kim),resp(x))
- 5: R[4,3]{y/x,z/kim} (-father (x,kim),answer(x))
- 6: R[5,2]{x/bob} (resp(bob))
- Luego la respuesta es resp(bob).

Ejemplo 2

- Sea KB:

  {(father(art,jon),father(bob,jon))
  (father(bob,kim))
  (-father(y,z),parent(y,z))}

- Queremos responder la siguiente consulta:

  {x | parent(x,jon)}

Ejemplo 2 (cont.)

- 1: (father(art,jon),father(bob,jon))
- 2: (father(bob,kim))
- 3: (-father(y,z),parent(y,z))
- 4: -α = (-parent(x,jon),resp(x))
- 5: R[4,3]{y/x,z/jon} (-father (x,jon),answer(x))

Ejemplo 2 (cont.)

- 1: (father(art,jon),father(bob,jon))
- 2: (father(bob,kim))
- 3: (-father(y,z),parent(y,z))
- 4: -α = (-parent(x,jon),resp(x))
- 5: R[4,3]{y/z,z/jon} (-father (x,jon),answer(x))
- 6: R[5,1]{x/art} (father(bob,jon), resp(art)
- 7: R[6,3]{y/bob,z/jon} (parent(bob,jon),resp(art))
- 8: R[7,4] (resp(bob),resp(art))

  Esta es una respuesta disyuntiva.

Factorización

- Las cláusulas (P(x),P(y)) y (-P(v),-P(w)) no son satisfacibles.
- Luego si refutación mediante resolución es completa (para lógica de predicado) debería existir una refutaci'on.

Factorización

- Sin embargo, no existe una refutaci'on para este par de cl'ausulas:

(P(x),P(y)) (-P(v),-P(w))

(P(y),-P(w)){x/v}

(P(y),P(x)){y/w}

Factorización

- Si dos o más literales de una cláusula C tienen un UMG σ, entonces Cσ (con eliminación de literales duplicados) es un factor de C.
- Ejemplo:
  - C=(P(x),P(f(y)), -Q(X))
  - σ={x/f(y)}
  - Luego (P(f(y)), -Q(f(y))) es un factor de C.
- Regla de resolución se generaliza de modo que C1 y C2 derivan C ssi existen factores C1’ y C2’ que resuelven a C.

Completitud de Refutación Mediante Resolución

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

  Existe una refutación usando resolución (con factorización) para (KB $\wedge \neg\alpha$) ssi

  KB $\models \alpha$.

Problema

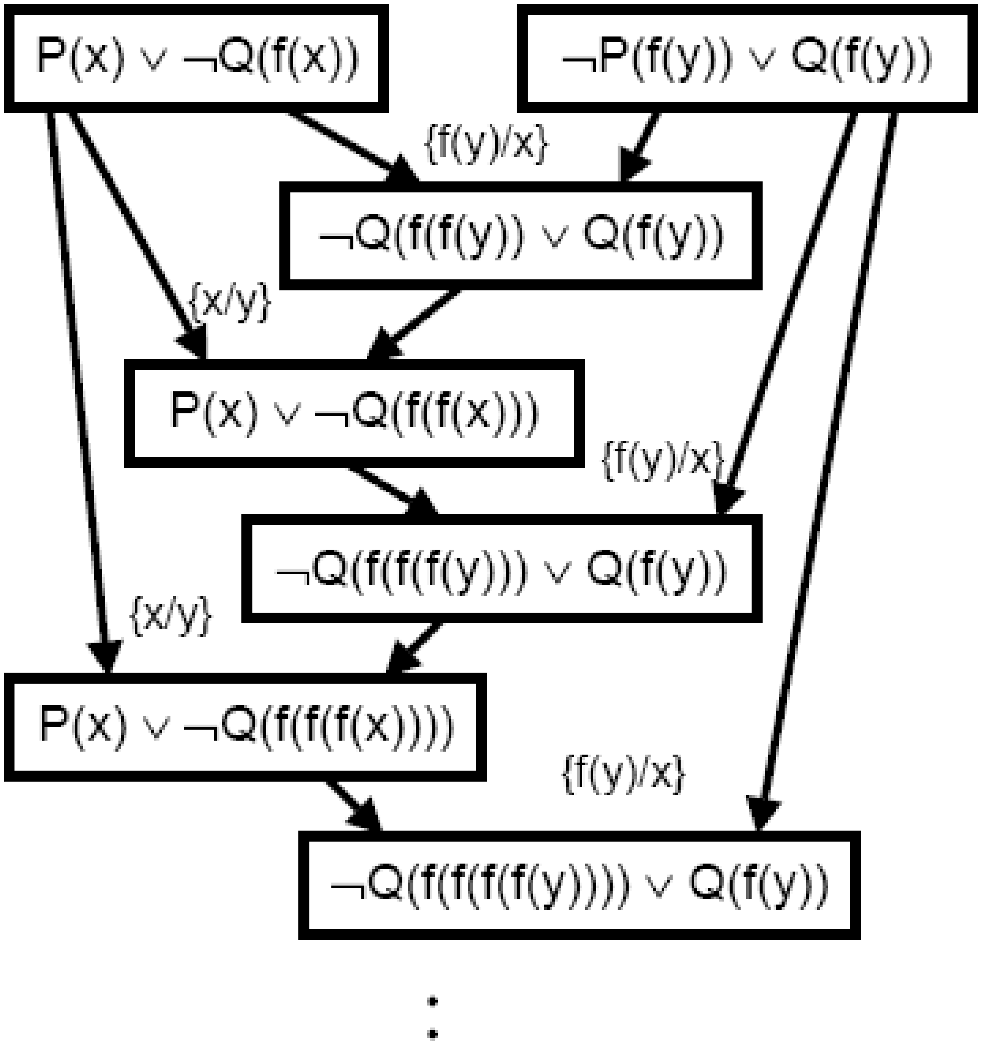
- El teorema de completitud no asegura que un algoritmo de b'usqueda de refutaciones aplicando resoluci'on debe terminar.

Algoritmo de Refutación Mediate Resolución

KB $\models \alpha$?

- S:= (KB $\wedge \neg \alpha$)
- Convertir S en CNF
- En forma iterativa, aplicar regla de resolución a cláusulas de S e ir añadiendo los resolventes a S. Parar cuando:
  - No tengamos nuevos resolventes, en este caso return(false)
  - Se generó la clásula vacía, en esta caso return(true)

No Términaci'on



No T'erminaci'on

- Si existe una reefutaci'on el algoritmo termina
- Pero si no existe, puede entrar en un loop infinito.
- El problema es que a diferencia de l'ogica de proposiciones, en l'ogica de predicados se pueden generar un n'umero infinito de cl'ausulas.

No-decidibilidad de Lógica de Primer Orden

- El problema es inebitable.
- No existe un algoritmo que decida si un conjunto arbitrario de cl'ausulas es satisfacible.
  - Se puede demostrar que este problema es equivalente al problema de parada de una máquina de Turing.
- Debido a esto, refutaci'on mediante resoluci'on no necesariamente para.

Teorema de Herbrand

- Dada una cláusula P, una cláusula base de P es una cláusula Pσ, donde σ es una sustitución de variables a términos sin variables.
- **Teorema de Herbrand:** Dado un conjunto finito de cláusulas A, A es insatisfacible ssi existe un conjunto de cláusulas base de A que son insatisfacibles.

Ejemplo: Teorema de Herbrand

- Dado S={

1: (P(a,x,f(g(y))),

2: (-P(z,f(z),f(w)), Q(w,z)),

3:(-Q(g(u),u)) }

- Para determinar si S es insatisfacible debemos buscar un conjunto de cl'ausulas bases de S que sean insatisfacibles.

Ejemplo: Teorema de Herbrand

- Dado S={

1: (P(a,x,f(g(y))),

2: (-P(z,f(z),f(w)), Q(w,z)),

3:(-Q(g(u),u)) }

- El conjunto de cl'ausulas base insatisfacibles es S'={

1': (P(a,f(a),f(g(a))),

2': (-P(a,f(a),f(g(a))), Q(g(a),a)),

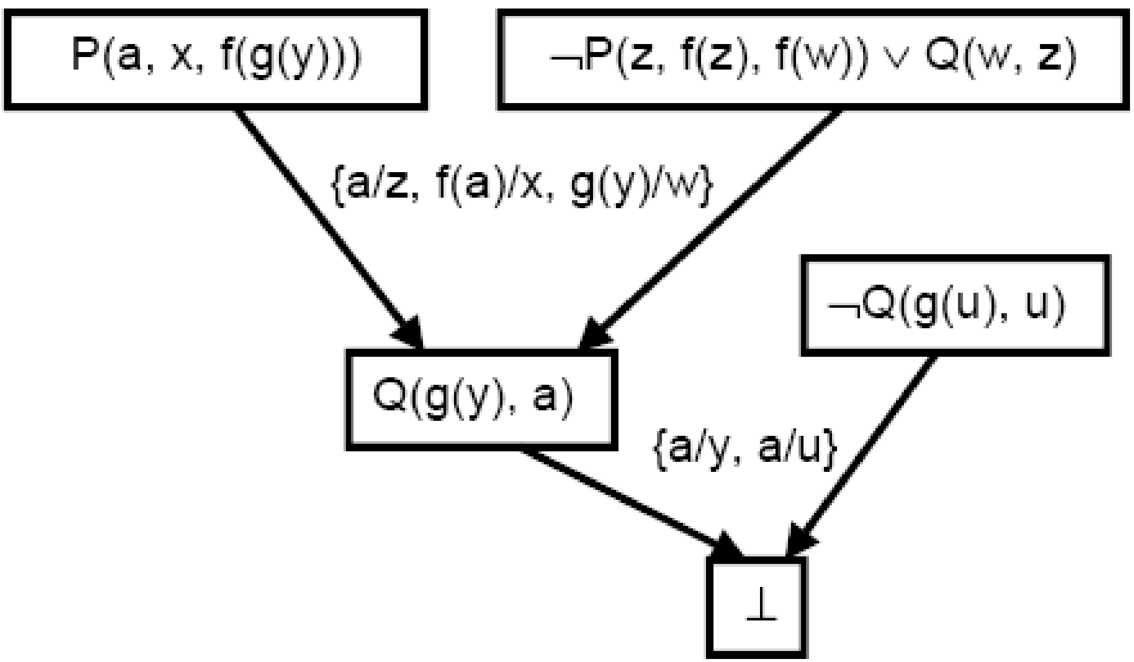
3': (-Q(g(a),a)) }

Teorema de Herbrand

Suppose that we are given the following clauses Φ:

$$P(a, x, f(g(y))),$$
$$\neg P(z, f(z), f(w)) \vee Q(w, z),$$
$$\neg Q(g(u), u).$$

Here is a resolution refutation, taken from previous slides:



Teorema de Herbrand

- Teorema de Herbrand reduce satisfabilidad en lógica de prédicados a satisfacibilidad en lógica proposicional (lógica de las cláusulas base).
- El problema está en que el número de cláusulas base puede ser infinito.
- Incluso si necesitamos un conjunto finito de cláusulas para establecer insatisfacibilidad, no sabemos cuál es este conjunto y debemos buscar en un espacio infinito de conjuntos.

Teorema de Herbrand y No Términaci'on

- Corolario de Teorema de Herbrand: para cláusulas sin funciones refutación mediante resolución termina.