Programación Lógica

Fundamentos teóricos:

Resolución

Instituto de Computación - Facultad de Ingeniería

Objetivos

- Regla de resolución para cláusulas de 1er orden
- Refinamientos de la resolución
  - Resolución lineal
  - Resolución con cláusulas de entrada
  - Resolución SLD
- Mecanismo de ejecución de Prolog y árbol-SLD

Resolución en 1er orden

Unificador más general (m.g.u.)

θ es un mgu para E1 y E2 si:

a. θ es un unificador para E1 y E2 y

b. si σ es un unificador para E1 y E2 , existe una sustitución γ t.q. $\theta \circ \gamma = \sigma$

Resolución

$$\frac{\alpha \vee \beta, \quad \gamma \vee \neg\beta}{\alpha \vee \gamma}$$

- Combinando estos 2 conceptos definiremos resolución para 1er orden.
- Nos restringimos a cláusulas.

Resolución en cláusulas de 1er orden

C1: {p(X), q(f(X),a), ¬r(Y)}

C2: {s(Z), t(a), ¬q(f(b),Z)}

Podemos aplicar resolución entre C1 y C2 ?

Resolución en cláusulas de 1er orden

C1: {p(X), q(f(X),a), ¬r(Y)}

C2: {s(Z), t(a), ¬q(f(b),Z)}

Podemos aplicar resolución entre C1 y C2 ?

No hay literales complementarios, pero podemos hacer que dos literales sean complementarios mediante sustituciones.

Resolución en cláusulas de 1er orden

C1: {p(X), q(f(X),a), ¬r(Y)}

C2: {s(Z), t(a), ¬q(f(b),Z)}

Podemos aplicar resolución entre C1 y C2 ?

No hay literales complementarios, pero podemos hacer que dos literales sean complementarios mediante sustituciones.

Debemos hacer idénticos q(f(X),a) , q(f(b),Z)

Resolución en cláusulas de 1er orden

C1: {p(X), q(f(X),a), ¬r(Y)}

C2: {s(Z), t(a), ¬q(f(b),Z)}

Podemos aplicar resolución entre C1 y C2 ?

No hay literales complementarios, pero podemos hacer que dos literales sean complementarios mediante sustituciones.

Debemos hacer idénticos q(f(X),a) , q(f(b),Z)

Para ello buscamos θ , mgu { q(f(X),a) , q(f(b),Z) }

Resolución en cláusulas de 1er orden

C1: {p(X), q(f(X),a), ¬r(Y)}

C2: {s(Z), t(a), ¬q(f(b),Z)

Para ello buscamos θ , mgu { q(f(X),a) , q(f(b),Z) } , θ = { X/b, Z/a}

Y aplicamos el unificador a ambas cláusulas:

C1 θ : {p(X), q(f(X),a), ¬r(Y)} = {p(b), q(f(b),a), ¬r(Y)}
C2 θ : {s(Z), t(a), ¬q(f(b),Z) } = {s(a), t(a), ¬q(f(b),a) }

La resolvente es C3: **{p(b), ¬r(Y), s(a), t(a)}**

Resolución en cláusulas de 1er orden

C1: {p(X), q(f(X),a), ¬r(Y)}

C2: {s(Z), t(a), ¬q(f(b),Z)

Por qué el mgu y no un unificador cualquiera ?

Para ello buscamos θ , mgu { q(f(X),a) , q(f(b),Z) } , θ = { X/b, Z/a}

Y aplicamos el unificador a ambas cláusulas:

C1 θ : {p(X), q(f(X),a), ¬r(Y)} = {p(b), q(f(b),a), ¬r(Y)}

C2 θ : {s(Z), t(a), ¬q(f(b),Z) } = {s(a), t(a), ¬q(f(b),a) }

La resolvente es C3: **{p(b), ¬r(Y), s(a), t(a)}**

Resolución en cláusulas de 1er orden

C1: {p(X), q(f(X),a), ¬r(Y)}

C2: {s(Z), t(a), ¬q(f(b),Z)

Por qué el mgu y no un unificador cualquiera ?

Para ello buscamos θ , mgu { q(f(X),a) , q(f(b),Z) } , θ = { X/b, Z/a}

Y aplicamos el unificador a ambas cláusulas:

C1 θ : {p(X), q(f(X),a), ¬r(Y)} = {p(b), q(f(b),a), ¬r(Y)}

C2 θ : {s(Z), t(a), ¬q(f(b),Z) } = {s(a), t(a), ¬q(f(b),a) }

Por qué aplicar el mgu a toda la clásula ?

La resolvente es C3: **{p(b), ¬r(Y), s(a), t(a)}**

Resolución en cláusulas de 1er orden

Def.

Para C1 y C2 cláusulas de 1er orden, sin variables en común

C1: {L1 , L2 , ... , Ln}
C2: {M1 , M2 , ... , Mh}
θ, mgu de Li y ¬Mj

resolvente C3 :
{L1 , L2 , ... , Li-1 , Li+1 , ... , Ln , M1 , … , Mj-1 , Mj+1 , … , Mh} θ

Resolución para cláusulas de 1er orden

Correctitud.

- **Sean C1 y C2 dos cláusulas de 1er orden. Si R es una resolvente de C1 y C2 se cumple:**

$$C1, C2 \models R$$

Resolución para cláusulas de 1er orden

Correctitud.

Sean C1 y C2 dos cláusulas de 1er orden. Si R es una resolvente de C1 y C2 se cumple:

$$C1, C2 \models R$$

1. $C \models C\theta$, C cláusula, θ sustitución cualquiera
2. A v B y C v ¬ B $\models$ A v C, siendo A, B y C flas. de 1er orden cualesquiera

Resolución para cláusulas de 1er orden

Completitud.

$S \models C$ **implica** $S \vdash_R C$ **?**

Resolución para cláusulas de 1er orden

Completitud.

$S \models C$ **implica** $S \vdash_R C$ **?**

No podemos derivar p(f(y)) a partir de p(x), aunque
$p(x) \models p(f(y))$

Sí podemos probar
{p(x), ¬p(f(y))} es inconsistente.

Procedimientos de resolución

Aplicación sistemática de resolución

- Saturación por niveles
- Resolución lineal
- Resolución con cláusula de entrada

Ejs.-

S = {{p,q}, {¬p,q}, {p, ¬q}, {¬p, ¬q}}

Saturación por niveles

$S^0 = S$

$S^{i+1} = \{$*resolventes de* $C1 \in (S^0 \cup \ldots \cup S^i)$ *y* $C2 \in S^i\}$

- Se resuelven sistemáticamente todas las cláusulas entre si.
- Se para cuando $\square \in S^i$

Saturación por niveles

$S^0 = S$

S^{i+1} = *{resolventes de* $C1 \in (S^0 \cup \ldots \cup S^i)$ *y* $C2 \in S^i$*}*

$S = \{\{p,q\}, \{\neg p,q\}, \{p, \neg q\}, \{\neg p, \neg q\}\}$

Saturación por niveles

$S^0 = S$

$S^{i+1} = \{$*resolventes de* $C1 \in (S^0 \cup \ldots \cup S^i)$ *y* $C2 \in S^i\}$

$S = \{\{p,q\}, \{\neg p,q\}, \{p, \neg q\}, \{\neg p, \neg q\}\}$

$S^0 = \{\{p,q\}, \{\neg p,q\}, \{p, \neg q\}, \{\neg p, \neg q\}\}$

$S^1 = \{\{p,q\}, \{\neg p,q\}, \{p, \neg q\}, \{\neg p, \neg q\}, \{q\}, \{p\}, \{\neg q\}\}$

$S^2 = \{\{p,q\}, \{\neg p,q\}, \{p, \neg q\}, \{\neg p, \neg q\}, \{q\}, \{p\}, \{\neg q\}, \ldots., \square, \}$

Saturación por niveles

$S^0 = S$

$S^{i+1} = \{$*resolventes de* $C1 \in (S^0 \cup \ldots \cup S^i)$ *y* $C2 \in S^i\}$

$S = \{\{p,q\}, \{\neg p,q\}, \{p, \neg q\}, \{\neg p, \neg q\}\}$

$S^0 = \{\{p,q\}, \{\neg p,q\}, \{p, \neg q\}, \{\neg p, \neg q\}\}$

$S^1 = \{\{p,q\}, \{\neg p,q\}, \{p, \neg q\}, \{\neg p, \neg q\}, \{q\}, \{p\}, \{\neg q\}\}$

$S^2 = \{\{p,q\}, \{\neg p,q\}, \{p, \neg q\}, \{\neg p, \neg q\}, \{q\}, \{p\}, \{\neg q\}, \ldots., \square, \}$

Procedimiento seguro pero muy costoso !!

Refinamientos de la resolución

- El método general (saturación por niveles) es muy costoso.
- Se proponen ***refinamientos*** para evitar resolver todas las cláusulas contra todas.
  - Resolución lineal
  - Resolución con cláusulas de entrada
- Se debe analizar la completitud.

Resolución lineal

- En cada paso de resolución, una de las cláusulas es la resolvente del paso anterior.
- Ej. *{{p,¬q,r} , {q, s}, {¬s}, {¬r}}*
- La resolución lineal es completa.
- Atención !! La elección de la cláusula inicial influye.

{{p,¬q,r},{q, s},{¬s}, {¬r},{a,b,c}}

Resolución lineal

- En cada paso de resolución, una de las cláusulas es la resolvente del paso anterior.
- Ej. *{{p,¬q,r} , {q, s}, {¬s}, {¬r}}*
- La resolución lineal es completa.
- Atención !! La elección de la cláusula inicial influye.

{{p,¬q,r},{q, s},{¬s}, {¬r},{a,b,c}}

- Es completa sin restricciones para conjuntos **mínimamente insatisfactibles**.

- **Conjunto mínimamente insatisfactible**: queda satisfactible si eliminamos una cláusula cualquiera.

Resolución con cláusula de entrada

- Resolución lineal en donde una de la clásulas pertenece al conjunto S de entrada.

- No es completa.

Resolución con cláusula de entrada

- Resolución lineal en donde una de la clásulas pertenece al conjunto S de entrada.

- No es completa.

Considerar:

$S = \{\{p,q\},\{\neg p,q\},\{p,\neg q\},\{\neg p,\neg q\}\}$

todas las clásulas tienen dos literales, lo que hace que cualquier resolvente siempre tenga al menos un literal.

Resolución SLD

- La resolución con cláusula de entrada es completa para cláusulas de Horn (cláusulas definidas + objetivo).
- SLD = Selection Linear Definite
- En el primer paso de resolución interviene necesariamente el objetivo. Por qué?

Resolución SLD

- Ejemplo:

p(x):- q(x),p(a).

q(b).

p(a).

r(c).

r(b)

Objetivo:

?- p(Z),r(Z).

Resolución SLD

Def. Resolvente-SLD.

Sea P un programa definido,

G un objetivo, $\leftarrow$A1,...,An

G' es la **resolvente SLD** de G y una cláusula

A $\leftarrow$B1,...,Bn de P con mgu θ si:

- Am es un átomo (átomo seleccionado) de G
- θ es un mgu de Am y A
- G' es $\leftarrow$(A1,...,Am-1,B1,...,Bn,Am+1,...,An)θ

Derivación SLD

Sea P un programa definido,

G un objetivo definido $\leftarrow$A1,...,An.

Una **derivación-SLD** de P $\cup$ {G} es

- una secuencia G0=G, G1, G2,... de objetivos,
- una secuencia C1,C2,... de variantes de cláusulas de P y
- una secuencia θ1, θ2,... de mgus tal que cada Gi+1 se deriva de Gi y Ci con unificador θi.

Refutación SLD

Una refutación-SLD para $P \cup \{G\}$ es una derivación-SLD finita cuyo último objetivo es $\square$.

Procedimientos de refutación-SLD.

- El objetivo de un interpréte es, dado un programa P y una consulta C, encontrar una refutación-SLD para $P \cup \{G\}$, siendo $G = \neg C$.
- Una refutación está compuesta por una cantidad finita de pasos de resolución-SLD. En cada uno de estos pasos hay 2 aspectos que podrían resolverse de más de un modo:
  - cuál es el átomo del objetivo que se selecciona para reducir
  - cuál es la cláusula del programa que reduce con el átomo seleccionado del objetivo.

Respuesta computada

Def.-Respuesta computada.

- Una respuesta computada para $P \cup \{G\}$ es la sustitución obtenida restringiendo a las variables de G la composición de mgus utilizados en una refutación-SLD de $P \cup \{G\}$.

- La correctitud de la resolución-SLD ya fue demostrada (fue demostrada respecto a resolución en general, se cumple para resolución-SLD). Es posible demostrar además un resultado de correctitud respecto a la respuesta computada.

Respuesta correcta

Def. Respuesta correcta.

Sea P un programa lógico, G un objetivo, θ una sustitución para las variables de G, θ es una respuesta correcta si

$\forall x1 ... \forall xk((A1,A2,...,Ak)\theta)$ es consecuencia lógica de P.

Respuesta computada / correcta

Nos interesa establecer relaciones entre respuestas computadas y correctas

1- ¿Toda respuesta computada es una respuesta correcta?

2-¿Toda respuesta correcta es una respuesta computada?

Completitud de resolución-SLD

Para cláusulas definidas la resolución-SLD es completa.

En otros términos, si $P \cup \{G\}$ es insatisfactible, existe una refutación-SLD para $P \cup \{G\}$.

Completitud respecto a respuestas

Si θ es una respuesta correcta, existe

una respuesta computada σ y

una sustitución γ

tales que $\theta = \sigma\gamma$.

(Notar que las respuestas correctas no son necesariamente las más generales).

Árbol-SLD

Regla de computación:

Política de selección del siguiente átomo a reducir en un objetivo.

La regla de computación no afecta la existencia de una refutación-SLD.

Considerando una regla de computación fija, el espacio de búsqueda (asociado a las distintas formas de elegir las cláusulas del programa) es un árbol.

Lo llamamos **árbol-SLD**.

Árbol-SLD

Def. Sea P un programa lógico, G un objetivo.

Un **árbol-SLD para P ∪ {G}** es un árbol que satisface:

1. Todo nodo es un objetivo (eventualmente vacío)
2. La raíz es G
3. Sea ← A1,...,Am,...,Ak , k>0, un nodo del árbol. Supongamos que Am es el átomo seleccionado. Entonces, para toda cláusula A ← B1,...,Bq tal que Am y A unifican con mgu θ, hay un hijo cuyo objetivo es la resolvente-SLD entre el objetivo y la cláusula: ← (A1,...Am-1,B1,...,Bq,Am+1,...,Ak)θ
4. Los nodos con la cláusula vacía no tienen hijos

Árbol-SLD

- Cada rama del árbol SLD corresponde a una posible derivación SLD.
- Por lo tanto, el árbol SLD representa todas las posibles derivaciones SLD a partir de un objetivo, utilizando una determinada regla de computación.
- La regla de computación determina las ramas del árbol: para un mismo programa lógico y objetivo, el árbol SLD posiblemente cambie al cambiar la regla de computación.
- A las ramas que corresponden a refutaciones (la hoja es un objetivo vacío) se les llama **ramas exitosas**.
- A las ramas que corresponden a derivaciones infinitas se les llama **ramas infinitas**.
- A las ramas que corresponden a derivaciones fallidas se les llama **ramas de falla**.

Árbol-SLD

Ejemplo

p(X,Z) ← q(X,Y), p(Y,Z)

p(X,X)

q(a,b)

Consulta : p(X,b)

con Reglas de Computación :

1 - átomo de más a la izquierda

2 - átomo de más a la derecha

Refutación SLD

Independencia de la regla de computación

Sea P un programa lógico, G un objetivo :

Si existe una refutación-SLD para $P \cup \{G\}$ con respuesta computada θ, entonces para cualquier regla de computación R existe una refutación-SLD con respuesta computada θ', tal que $G\theta'$ es una variante de $G\theta$.

Procedimiento de refutación SLD

Definición

Una regla de computación junto a una estrategia de búsqueda en un árbol SLD definen un **procedimiento de refutación SLD**.

Dos estrategias clásicas: BFS y DFS.

- Si el árbol es finito, ambas llegan a todos los nodos del árbol (posiblemente en distinto orden).
- Si no, BFS sigue llegando a cualquier nodo en una cantidad finita de pasos pero DFS no (aunque es más eficiente).

Procedimiento de refutación SLD

Un intérprete Prolog estándar utiliza el siguiente procedimiento de refutación SLD:

- regla de computación: elegir el átomo de más a la izquierda.
- estrategia de búsqueda: en profundidad, eligiendo las ramas del árbol según el orden de aparición de las cláusulas en el programa.

Por lo tanto, Prolog es incompleto: existen objetivos para los cuales hay alguna refutación SLD que el intérprete es incapaz de encontrar.

Resumen

- Resolución en 1er orden
- Algunos refinamientos, completitud
- Resolución-SLD
  - Derivación
  - Refutación
  - Árbol-SLD
  - Rama de éxito, rama fallida, rama infinita
- Cómo implementa Prolog la resolución-SLD

Próxima: predicados extralógicos

- Cut
- fail, not
- Análisis de terminos