Búsqueda e inferencia lógica

Introducción y Principio de Resolución
Estrategias de resolución
Procedimientos de extracción de respuesta
Demostradores de teoremas
Programación lógica y Prolog

Contenidos

1. Introducción
2. Principio de resolución
3. Estrategias de resolución
4. Procedimiento de extracción de respuesta
5. Demostradores de teoremas
6. Programación lógica y Prolog

Introducción

- Utilidad lógica simbólica
  - Representar problemas mediante FBF's
    - Conjunto de FBF's consistente (teoría)
  - Solucionar problemas de forma deductiva
    - Reglas de inferencia de la lógica
    - Proceso de búsqueda en el espacio de FBF's
- Dificultad: la aplicación no controlada de Reglas de Inferencia da lugar a un problema de explosión combinatoria
  - Crecimiento exponencial del nº de FBF's generadas

Métodos uniformes de demostración

- Utilizan una única regla de inferencia
- Una forma de reducir la complejidad de la búsqueda
- Requieren transformar fórmulas a formato estándar

- Refutación por resolución
  - Para probar que $\Omega \models \alpha$, probar que $\Omega \cup \neg\alpha$ es inconsistente
  - única regla de inferencia: resolución

Dos aproximaciones básicas

- Sistemas de resolución
  - Transformar FBF's a forma de cláusula
  - Aplicar resolución hasta generar cláusula vacía
  - Reducción complejidad sin limitar capacidad representación: selección de estrategias adecuadas
- Programación lógica
  - Sólo cláusula de Horn
  - Resolución SLD
  - Reducción complejidad limitando capacidad representación: cláusulas de Horn

Principio de resolución

- Lógica proposicional
  - Formas normales
  - Resolución
- Lógica primer orden
  - Formas normales
  - Resolución

Formas Normales: literal y cláusula

- Def. Literal, λ: un átomo o su negación
- Def. Literales complementarios: λ, μ son complementarios sii $\lambda = \neg\mu$
- Def. Cláusula, K: Disyunción de literales
  - p, q, r átomos, $(p \vee \neg q \vee r)$ cláusula
  - Representación alternativa: conjunto de literales ente los que se supone disyunción $\{p, \neg q , r\}$
- Cláusula vacía, $\square$: no contiene ningún literal
  - Por convenio, insatisfacible

Forma clausulada

- Def. Forma Normal Conjuntiva, FNC
  - G está en FNC sii $G = k_1 \wedge k_2 \wedge \ldots \ldots \wedge k_n$, $n \geq 1$
  - $(p \vee \neg q) \wedge (q \vee r)$
- Def. Forma Clausulada: conjunto de cláusulas ente las que se supone la conjunción
  - $\{\{p, \neg q\}, \{q, r\}\}$
- Forma clausulada vacía, $\varnothing$: no contiene ninguna cláusula
  - Por convenio, satisfacible

Transformación FBF's a forma clausulada

- Teorema 1: Toda FBF se puede transformar en un fórmula en FNC equivalente
  - Paso 1: eliminar ocurrencias de $\leftrightarrow$, $\supset$ usando leyes 1 y 2
  - Paso 2: colocar $\neg$ inmediatamente delante de los átomos, usado leyes 10 (doble negación) y 11 (de Morga)
  - Paso 3: obtener fórmula en FNC utilizando ley 5 (distributiva $\vee$ respecto $\wedge$)

Principio de resolución

- Def. resolvente
  - k_1, k_2 cláusulas, con $\lambda \in k_1$, $\neg\lambda \in k_2$
  - $res_\lambda(k_1, k_2) = (k_1 - \{\lambda\}) \cup (k_2 - \{\neg\lambda\})$
    - $res_\lambda(k_1, k_2)$ resolvente de k_1 y k_2 respecto a λ
    - k_1 y k_2, cláusulas padres de $res_\lambda(k_1, k_2)$

Principales resultados

- Teorema 2. $k_1, k_2 \vDash res_\lambda(k_1, k_2)$
- Def. Derivación por resolución, $S \vdash_r k$
  - S conjunto finito de cláusulas, k cláusula
  - $S \vdash_r k$: $k_1, k_2, \ldots \ldots, k_n$, tales que
    - $k_n = k$
    - para cada k_l

      ó $k_l \in S$

      ó $\exists\ i, j < l$ y λ / $k_l = res_\lambda(k_i, k_j)$
- Teorema 3. Si $\exists\ S \vdash_r k$, $S \vDash k$

Teorema 4: teorema de resolución básica

- Sea S un conjunto finito de cláusulas.

S es inconsistente sii $\exists$ S $\vdash_r \square$

Refutación por resolución: procedimiento

- Sea *T* una teoría sólida y completa y *t* una FBF. Para probar $\exists$ T $\vdash$ t mediante refutación por resolución:
  1. Convertir los axiomas de *T* a FNC. Crear el conjunto S_0 como la conjunción de todas las cláusulas obtenidas
  2. Negar *t* y convertir a FNC. Añadir las cláusulas obtenidas a S_0, obteniendo S
  3. Repetir hasta obtener $\square$ o no se generen nuevas cláusulas.
     a) Seleccionar dos cláusulas que se puedan resolver, formando su resolvente
     b) Si el resolvente no es $\square$, añadir a S

Refutación por resolución: parada

- Lógica proposicional es decidible: siempre termina
  - Generando cláusula vacía
    - S inconsistente, $S_0 \cup \{\neg t\}$ inconsistente, $S_0 \models t$, $T \models t$, si T completa $\exists\ T \vdash t$
  - No se generan nuevas resolventes
    - S consistente, $S_0 \cup \{\neg t\}$ consistente, $S_0 \not\models t$, $T \not\models t$, si T sólida, $\nexists\ T \vdash t$

Formas normales en lógica de primer orden: Forma Normal Prenexa, FNP

- G está en FNP sii

  $G = Q_1x_1\ Q_2x_2 \ldots\ \ldots Q_nx_n\ M(x_1, x_2, \ldots\ \ldots, x_n)$

  Con

  - Q_i: $\forall$ ó $\exists$
  - Prefijo: $Q_1x_1\ Q_2x_2 \ldots\ \ldots Q_nx_n$
  - Matriz: $M(x_1, x_2, \ldots\ \ldots, x_n)$
  - La matriz es una FBF que no contiene cuantificadores y cuyas únicas variables, libres, son: $x_1, x_2, \ldots\ \ldots, x_n$

Transformación de sentencias a FNP

- Teorema 5: toda sentencia se puede transformar a una FNP equivalente
  - Paso 1: eliminar ocurrencias de $\leftrightarrow$, $\supset$ usando leyes 1 y 2
  - Paso 2: colocar $\neg$ inmediatamente delante de los átomos, usado leyes 10 (doble negación), 11 (De Morgan) y 13 (De Morgan cuantificadores)
  - Paso 3: renombrar las variables hasta que no haya dos cuantificadores con la misma variable
  - Paso 4: colocar los cuantificadores al principios de la fórmula, para dejarla en FNP

    Siempre que sea posible, colocar los cuantificadores existenciales delante de los universales

Forma estándar de Skolem

- Sentencia que se obtiene a partir de FNP eliminando los cuantificadores existenciales del prefijo.

- También denominada Forma Normal de Skolem o simplemente Forma Estándar

Conversión FNP a Forma Estándar de Skolem

- $G = Q_1x_1\ Q_2x_2 \ldots \ldots Q_nx_n\ M(x_1, x_2, \ldots \ldots, x_n)$
- Q_r un cuantificador existencial del prefijo, $1 \leq r \leq n$
- Para eliminar Q_rx_r del prefijo
  a) No hay ningún cuantificar universal delante de Q_rx_r
     - Elegir constante C que no ocurra en M
     - Reemplazar todas las ocurrencias de x_r en M por C
  b) Sean $Q_{S_1}, Q_{S_2}, \ldots \ldots Q_{S_m}$ todos los cuantificadores universales que ocurren delante de Q_r
     - Elegir función de grado m, f_r, que no ocurra en M
     - Reemplazar todas las ocurrencias de x_r en M por $f_r(x_{S_1}, x_{S_2}, \ldots \ldots x_{S_m})$
- Cuando todos los cuantificadores existenciales han sido eliminados por este procedimiento, la última fórmula obtenida es la forma estándar de Skolem de G, G_s

Teorema 6: teorema de Skolem

- Sea G una sentencia y G_s su Forma Estándar de Skolem. G es inconsistente sii G_s es inconsistente.

Forma clausulada

- Def. literal
- Def. cláusula
- Def. Forma Normal Conjuntiva
  - G, FBF que no contiene cuantificadores
  - G está en FNC sii es una conjunción de cláusulas
- Def. Forma Clausulada
  - Una forma clausulada es un conjunto de cláusulas entre las que se supone la conjunción y se asume que todas las variables están cuantificadas universalmente

Transformación de sentencias a forma clausulada

- Corolario: cualquier sentencia se puede transformar a forma clausulada
  - Paso 1: obtener FNP
  - Paso 2: obtener forma estándar de Skolem
  - Paso 3: obtener FNC de la matriz

  El conjunto de cláusulas obtenidas es inconsistente sii la sentencia original es inconsistente.

Principio de resolución

- Def. resolvente binario
  - k_1, k_2 cláusulas que no tienen variables comunes, con $\lambda \in k_1$, $\neg\mu \in k_2$, $g = umg(\lambda, \mu)$
  - $res^b_{\lambda, \mu}(k_1, k_2) = (k_1 g - \{\lambda\} g) \cup (k_2 g - \{\neg\mu\} g)$
    - $res^b_{\lambda, \mu}(k_1, k_2)$ resolvente binario de k_1 y k_2 respecto a λ, μ
    - k_1 y k_2, cláusulas padres de $res^b_{\lambda}(k_1, k_2)$

Resolvente

- Def. factor
  - k cláusula, λ_1 , $\lambda_2 \in$k, g=umg(λ_1 , λ_2)
  - Factor de k: kg
- Def. Resolvente
  - k_1, k_2 cláusulas sin variables comunes
  - res(k_1, k_2): cualquiera de
    - $res^b_{\lambda, \mu}(k_1, k_2)$
    - $res^b_{\lambda, \mu}(k_1 g, k_2)$
    - $res^b_{\lambda, \mu}(k_1, k_2 g)$
    - $res^b_{\lambda, \mu}(k_1 g, k_2 g)$

Principales resultados

- Teorema 7. $k_1, k_2 \models res_\lambda(k_1, k_2)$
- Teorema 8. Si $\exists$ S $\vdash_r k$, S $\models k$

- Teorema 9. Teorema de resolución.

  Sea S un conjunto infinito numerable de cláusulas. S es inconsistente sii $\exists$ S $\vdash_r \square$

- Teorema 10. Teorema de compactidad

  Sea S un conjunto infinito numerable de cláusulas. S es inconsistente sii $\exists$ $S_0 \subset S$, S_0 finito y S_0 inconsistente.

Refutación por resolución: procedimiento

- Sea *T* una teoría sólida y completa y *t* una FBF. Para probar $\exists\ T \vdash t$ mediante refutación por resolución:
  1. Convertir los axiomas de *T* a FNC. Crear el conjunto S_0 como la conjunción de todas las cláusulas obtenidas
  2. Negar *t* y convertir a FNC. Añadir las cláusulas obtenidas a S_0, obteniendo S
  3. Repetir hasta obtener □, no se generen nuevas cláusulas o se agoten los recursos asignados
     a) Seleccionar dos cláusulas que se puedan resolver, formando su resolvente
     b) Si el resolvente no es □, añadir a S

Refutación por resolución: parada

- Lógica de primer orden es semidecidible: el cómputo de nuevas resolventes puede no terminar, finalizando el proceso por consumo de recursos
  - Si el cómputo termina, como en el caso proposicional
    - Generando cláusula vacía: S inconsistente
    - No se generan nuevas resolventes: S consistente
  - Si finaliza por consumo de recursos: no sabemos nada
    - S consistente, se pueden generar infinitas resolventes sin generar □
    - S inconsistente, "podríamos" generar □ asignando más recursos

Estrategias de Resolución

- Necesidad
  - la generación incontrolada de cláusulas hace que estas crezcan de forma exponencial
- Tipología
  - Simplificación: eliminan o reemplazan
  - Dirección: siguiente cláusula a considerar
  - Restricción: evitan generación de resolventes

Estrategia completa

- Una estrategia de resolución es completa (para la refutación) sii usada con una regla de inferencia completa (para la refutación) garantiza que encuentra una derivación de □ a partir de una forma clausulada inconsistente

- La regla de resolución es completa para la refutación

Estrategia de saturación por niveles

- Estrategia de dirección (similar bpa)

  Conjunto base: Conjunto S de todas las cláusulas de partida

  $S^0 = S$. Si $k \in S$, k cláusula de nivel *0*

  $S^i = \{res(k_1, k_2) / k_1 \in (S^0 \cup S^1 \cup \ldots \ldots \cup S^{i-1}), k_2 \in S^{i-1}\}$

  Si $k \in S^i$, $k \notin S^{i-1}$, k cláusula de nivel *i-esimo*

- Estrategia de resolución por niveles: obtener primero todas las resolventes de nivel *i* antes de obtener una resolvente de nivel *i+1*
- COMPLETA e ineficiente

Estrategias de simplificación

- Eliminación de literales puros
- Eliminación de tautologías
- Eliminación de cláusulas subsumidas
- Asociación de procedimientos

Eliminación de literales puros

- Def. literal puro

  S forma clausulada, $k \in S$, $\lambda \in k$ es un literal puro sii $\nexists k' \in S$ con $\neg\mu \in k'$ y substituciones s_1 y s_2 / $\lambda\, s_1 = \mu\, s_2$

- Estrategia de eliminación de literales puros
  - eliminar todas las cláusulas que contengan literales puros
  - sólo al conjunto inicial

- COMPLETA

Eliminación de tautologías

- Las cláusulas tautológicas no afectan a la satisfacibilidad
- Estrategia de eliminación de tautologías:
  - eliminar cláusulas tautológicas
  - iniciales y las que se generen

- COMPLETA

Eliminación de cláusulas subsumidas

- Def. subsunción

  k_1, k_2 cláusulas.

  k_1 subsume a k_2 sii $\exists$ substitución s / $k_1 s \subseteq k_2$

  k_2 cláusula subsumida

- Estrategia de eliminación de cláusulas subsumidas
  - Hacia delante: la resolvente puede ser subsumida
  - Hacia atrás: la resolvente puede subsumir

Eliminación de cláusulas subsumidas

- Completa con saturación por niveles. Puede no serlo con alguna estrategia de restricción

- MUY EFICIENTE: su aplicación suele ser imprescindible

Asociación de procedimientos

- Estrategia de simplificación/demoduladores
- Consiste en evaluar funciones o literales básicos sobre un dominio (interpretación parcial)
- Afecta a la satisfacibilidad pues estamos fijando una interpretación parcial
- NO es completa

Evaluación de funciones

- Asociar un símbolo de función con un procedimiento cuya evaluación devuelva un elemento del dominio
- Evaluar particularizaciones básicas del término funcional
- Reemplazar el término funcional por el elemento de dominio

$K = P(x) \vee Q(suma(3,5), y)$

se transforma en

$K = P(x) \vee Q(8, y)$ (con la interpretación habitual de la suma)

Evaluación de literales

- Asociar un símbolo de predicado con un procedimiento, cuya evaluación devuelva un valor de verdad
- Evaluar particularizaciones básicas del literal
- Si el literal se evalúa a T: eliminar la cláusula
- Si el literal se evalúa a F: eliminar el literal de la cláusula

$K = P(x) \vee MAYOR(3,5)$,

se transforma en

$K = P(x)$ (asumiendo $V(MAYOR(3,5)) = F$)

Estrategias de restricción

- Conjunto soporte
- Resolución lineal
- Resolución por entradas
- Resolución unitaria

Estrategia del conjunto soporte

- Def. Conjunto soporte

  S forma clausulada y $T \subset S$, $T \neq S$, $\varnothing$

  T se denomina conjunto soporte de S

- Def. conjunto soporte nivel i-esimo

  $T_0 = T$, T conjunto soporte de S. T_0 se denomina conjunto soporte de nivel 0

  T_i, conjunto soporte de nivel i-esimo: conjunto de $res(k_l, k_m)$ con:

  1. $\exists$ j / $k_j \in T_{i-1}$ (o factor de cláusula de T_{i-1})
  2. (la cláusula que no cumple 1) $\in$ S (o factor de cláusula de S)

Estrategia del conjunto soporte

- Def. T-soporte de una cláusula

  S forma clausulada, T conjunto soporte, k cláusula

  K tiene T-soporte sii $k \in T_i$, $i \geq 0$

- Estrategia del conjunto soporte

  S conjunto base, T conjunto soporte. La estrategia del conjunto soporte solo permite obtener cláusulas que tengan T-soporte

Teorema conjunto soporte

- Sea S forma clausulada inconsistente y T un conjunto soporte de S / S-T sea consistente.

  $\exists$ S $\vdash_r \square$ utilizando la estrategia del conjunto soporte, con T como conjunto soporte

- Elección habitual de T
  - Th(A) teoría de ax. propios A, *t* teorema
  - S: cláusulas de Th(A) y $\neg t$, inconsistente si *t* teorema
  - T: cláusulas que provienen de *t*
  - S-T: cláusulas de Th(A), normalmente consistente

Resolución lineal

- S forma clausulada, $kc_0 \in S$ cláusula central de partida

  La estrategia lineal sólo permite obtener como resolventes cláusulas centrales kc_{i+1}, con:

  1. $kc_{i+1} = res(kc_i, B_i)$
  2. $B_i \in S$ ó $B_i = kc_j$ con $j<i$ (o factor)

  B_i : cláusula lateral

Teorema complitud resolución lineal

- Sea S forma clausulada inconsistente y $kc_0 \in S$, de modo que $S-\{kc_0\}$ sea consistente.

  $\exists$ $S \vdash_r \square$ utilizando la estrategia de resolución lineal, con kc_0 como cláusula central de partida.

- Elección kc_0
  - Como conjunto soporte si la negación del teorema da lugar a una única cláusula.

Resolución por entradas

- Def. cláusula de entrada: cláusulas del conjunto base, S
- Def. resolvente de entrada: resolvente con al menos un padre cláusula de entrada

- Estrategia de resolución por entradas: sólo permite obtener resolventes de entrada

- NO es completa

Resolución unitaria

- Def. resolvente unitario: resolvente en el que al menos una de las cláusulas padres es unitaria.
- Estrategia de resolución unitaria: sólo permite obtener resolventes unitarios.
- NO es completa

Teorema equivalencia resolución unitaria y por entradas

Sea S forma clausulada.

$\exists$ S $\vdash_r \square$ utilizando la estrategia de resolución unitaria sii $\exists$ S $\vdash_r \square$ utilizando la estrategia de resolución por entradas

Cláusulas de Horn

- Def. Cláusula de Horn (definidas): cláusula que tiene, a lo sumo, un literal positivo.

- Las cláusulas de Horn se pueden interpretar como implicaciones, con el literal positivo a la derecha del símbolo implica:

$P(x) \vee \neg Q(x) \vee \neg R(y)$ $\quad \forall x \forall y\ (Q(x) \wedge R(y) \supset P(x))$

$P(x)$ $\quad \forall x\ (\ \supset P(x))$

$\neg Q(x) \vee \neg R(y)$ $\quad \forall x \forall y\ (Q(x) \wedge R(y) \supset\ )$

Teorema complitud resolución por entradas (unitaria)

Sea H una forma clausulada cuyos elementos son cláusulas de Horn.

H es inconsistente sii $\exists$ S $\vdash_r \square$ utilizando la estrategia de resolución por entradas (unitaria)

Procedimiento de extracción de respuesta

- Def. Extracción de respuesta mediante refutación por resolución: proceso de encontrar los elementos del dominio que hacen cierto el teorema a demostrar, mediante una prueba de refutación por resolución

Preguntas

- En general, cualquier FBF
- Por motivos prácticos, sentencias en FNP con
  - Matriz: conjunción de literales
  - Prefijo: sólo cuantificadores existenciales
- Interpretación
  - La demostración se puede interpretar como la pregunta: ¿Existen substituciones de variables que hagan cierto el teorema?
- Propiedad
  - La negación de estos teoremas da lugar a una única cláusula

Literal respuesta

- Def. Literal respuesta

Sea P pregunta y x_1, x_2,... ...,x_n las variables que ocurren en P.

Un literal respuesta para P es:

RES(x_1, x_2,... ...,x_n)

Obtención de la respuesta

- Si P es una pregunta y RES(x_1, x_2,… …,x_n) su literal respuesta, la respuesta se obtiene
  1. negando P y trasformándolo a cláusulas
  2. formando la disyunción de las cláusulas obtenidas en 1 con RES(x_1, x_2,… …,x_n)
  3. buscando derivaciones de cláusulas que solo contengan literales respuestas

Propiedades de la respuesta

- Puede contener más de un literal
- No es única
  - Depende de la derivación que la produce

Demostradores de teoremas

- Programas que dada Th(AP) y FBF t, intentan comprobar si $\exists$ AP $\vdash$ t
- Generalmente, aceptan FBF de LPO como entrada
- Generalmente basados en refutación por resolución
- Estrategias de control para limitar búsqueda

PTTP (*Prolog Technology Theorem Prover*)

- Búsqueda: descenso iterativo, completa (en vez de bpp)
  - La inferencia es completa con la regla de resolución lineal y por entradas
- Negación lógica: se implementa una rutina para probar P y otra para probar notP
- (Re)Introduce chequeo de ocurrencias en la unificación

OTTER (*Organized Techniques for Theorem-proving and Effective Research*)

- Refutación por resolución
- Uno de los primeros y más populares
  - sucesor: prover9
- Libre disposición
  - http://www.cs.unm.edu/~mccune/otter/
- Utilizado en
  - Investigación matemática
  - Verificación de hardware

Entrada de OTTER

- hechos importantes sobre el dominio (SOS, Set of Support):
- Conocimiento sobre el problema: axiomas de utilidad (Usables)
- Demoduladores: reglas de reescritura
- Parámetros y cláusulas que definen la estrategia de control

Procedimiento OTTER

Procedimiento OTTER(SOS, Usables)

entrada: SOS, Usables

repetir

cláusula <- elemento de SOS de menor peso

llevar cláusula de SOS a Usables

Procesar (Inferir (cláusula, Usables), SOS)

hasta SOS=∅ ó se ha encontrado refutación

end OTTER

Procedimiento OTTER

- Inferir (cláusula, Usables)
  - Resuelve cláusula con todas las de Usables
  - Filtra cláusulas
- Procesar (resolventes, SOS)
  - Estrategias simplificación
  - Llevar cláusulas a SOS según pesos
  - Comprobar presencia de cláusula unitaria y su complementario

Programación lógica y Prolog

- Principios de Programación Lógica
  - Introducción
  - Programas definidos: sintaxis de Edimburgo
  - Resolución SLD
  - Interprete abstracto de un Programa Lógico
  - Concepto de Respuesta
  - Programación lógica y negación
- Una implementación práctica: Prolog Estándar
  - Regla de cómputo y regla de búsqueda
  - Desviaciones del modelo básico

Introducción P.L.

- “parte de la informática que se ocupa de la lógica como lenguaje de programación”
  - Programa: conjunto finito de FBF’s
  - Computación: obtención de pruebas formales

Evolución histórica

- Origen: demostración automática de teoremas + IA
  - Herbrand(30),Davies-Putman(~60), Robinson(65)
- Aparición PL (~70)
  - Kowalsky, Colmerauer, Green, Hayes
- Primer interprete Prolog
  - Colmerauer, Rusell, Marsella 1972
- Primera implementación eficiente
  - Warren, AWM (Máquina Abstracta de Warren) Edimburgo, 1977

Cláusulas definidas

- A lo sumo, un literal positivo:

  $\neg b_1 \vee \neg b_2 \vee \ldots \quad \vee \neg b_n \vee a, \; n >= 0$

- En programación lógica se representa:

  $a \leftarrow b_1, b_2, \ldots \quad , b_n$

  - donde:
    - $a, b_1, b_2, \ldots \quad , b_n$ son literales positivos
    - todas las variables se consideran cuantificadas universalmente
  - a se denomina cabeza de la cláusula
  - $b_1, b_2, \ldots \quad , b_n$ se denomina cuerpo de la cláusula

Caracterización

- Programas Definidos
  - Cláusulas Horn o definidas
- Programas Normales
  - Cláusulas normales: extensión cláusula de Horn, admitiendo literales negativos en cuerpo cláusulas
- Programas: cualquier FBF

Programas Definidos: Sintaxis de Edimburgo

- Términos
  - constantes (numéricas, atómicas), variables, funciones
- Átomos

- Hecho o cláusula unitaria de programa: $a \leftarrow$
- Regla o cláusula de programa: $a \leftarrow b_1, b_2, \ldots , b_n$
- Pregunta o meta: $\leftarrow b_1, b_2, \ldots , b_n$

- Programa: conjunto finito de cláusulas de programa

Resolución SLD

Resolución **L**ineal con función de **S**elección para Cláusulas **D**efinidas.

C: $a \leftarrow b_1, b_2, \ldots b_n$ $n >= 0$, cláusula de programa

G: $\leftarrow a_1, a_2, \ldots a_k$ $k > 0$, pregunta

f_s: función de selección (regla de computo)

a_s: $f_s(G)$, literal seleccionado

Si a_s y a unifican con umg. θ, se denomina resolvente SLD de G y C a la meta:

$\leftarrow (a_1, a_2, \ldots a_{s-1}, b_1, b_2, \ldots b_n, a_{s+1}, \ldots a_k)\,\theta$

Derivación SLD (computo de G por P)

Sean P un programa y G una meta. Una derivación SLD de P U {G} consiste en tres secuencias, posiblemente infinitas, de:

$G_0=G, G_1, G_2, \ldots$	Metas
$C_1, C_2, C_3, \ldots$	Cláusulas de P renombradas
$\theta_1, \theta_2, \theta_3, \ldots$	umg's de C_i, G_{i-1}, respectivamente

tal que G_{i+1} es el resolvente SLD de G_i y C_{i+1} usando θ_{i+1}

ESTRATEGIA DE RESOLUCIÓN LINEAL Y POR ENTRADAS

Refutación SLD

- Def. Derivación SLD de □

Interprete abstracto de un Programa Lógico

InterpreteAbstracto(P, G)
    resolvente ← G;
    mientras (resolvente ≠ Ø) **hacer**
        Q=f_s(resolvente)
        Si (existe cláusula C de P cuya cabeza unifique con Q)
        Entonces
            resolvente ← resolvente_SLD de resolvente y C
                con umg θ
            G ← G θ
        Sino SalirMientras
        finsi
    finMientras
    Si (resolvente **=** Ø) **Entonces** (**Devolver** G)
    Sino (**Devolver** fallo)

Dos elecciones

- Regla de computo: literal sobre el que se resuelve, dado por al función de selección
- Regla de búsqueda: criterio de selección de la cláusula que resuelve (reduce) la meta

Efecto regla cómputo, búsqueda

- Regla cómputo: arbitrario
  - No afecta a la terminación
  - Quizás distintas respuestas
- Regla de búsqueda: no determinista
  - Afecta a la terminación

Concepto de respuesta

- Def. P programa definido, G meta definida

Una respuesta para P U {G} es

- Una substitución para las variables de G
- “no”

Respuesta correcta

Def. P programa definido,
G meta definida, G: $\leftarrow a_1, a_2, ... a_k$
θ una respuesta de P U {G}

θ es una respuesta correcta para P U {G} sii
$P \models \forall(a_1, a_2, ... a_k)\theta$
(P $\models \neg$ G θ sii P U {G θ} inconsistente)

"no" es una respuesta correcta para P U {G} sii
P U {G} es satisfacible

Respuesta computada

- Sea P programa definido, G meta definida / P U {G} tiene una refutación SLD
- Sean θ_1, θ_2, θ_3,... ...θ_n la secuencia de umg's utilizada en la refutación SLD

θ es una respuesta computada para P U {G} sii θ es la substitución obtenida seleccionando de θ_1 θ_2 θ_3... ...θ_n las ligaduras de las variables que ocurren en G.

Teorema solidez resolución SLD

Sea P un programa definido y G una meta definida.

Toda respuesta computada de P U {G} es una respuesta correcta de P U {G}

Teorema complitud resolución SLD

Sea P un programa definido, G una meta definida y θ una respuesta correcta de P U {G}

∃ Respuesta computada σ y substitución γ / θ y σ γ tienen el mismo efecto sobre las variables de G

(La respuesta computada puede ser más general que la correcta)

Teorema independencia de la regla de cómputo

- Sea P programa definido, G meta definida / P U {G} tiene una refutación SLD con respuesta computada θ.

Para cualquier otra regla de computo, R, existe una refutación SLD de P U {G} via R con respuesta computada θ' / Gθ' es una variante alfabética de Gθ

Programación lógica y negación

- Programa definido: conjunto de hechos y reglas que describen explícitamente *que es cierto*, sin información explicita sobre *que es falso*

- Dado Programa P, meta G, definidos, sólo podemos obtener respuestas computadas, σ, que también son correctas: sólo podemos derivar consecuencias lógicas

Suposición de mundo cerrado

- Regla de inferencia:

  Sea P programa definido y *a* átomo básico.

  Si $P \nvDash a$, inferir $\neg a$

- Observaciones:
  - SMC natural y efectiva en contexto de bases de datos
  - Regla de inferencia no-monotónica
  - Problemática en el contexto de Programación Lógica, pues no se puede garantizar el cómputo de $P \vDash a$

Necesidad negación

- Teóricamente, innecesaria: “Toda función computable en el sentido de Turing se puede computar con un programa definido” (1977, Tärlund)

- En la práctica, su ausencia limita capacidad expresiva
  - ¿Cómo definir que dos conjuntos son distintos sin la negación?

Negación por fallo

- Regla de inferencia

  Sea P programa definido y *a* átomo básico.

  Si $P \nvdash a$ tiene una prueba finita, inferir $\neg a$

- Prueba finita de $P \nvdash a$ (informal)
  - Número finito de derivaciones SLD
  - Todas finitas
  - Ninguna permite derivar *a*

Negación por fallo y programas definidos: Resolvente SLDNF

P programa definido, G_i meta normal,

$l_s = f_s(G_i)$ literal seleccionado

El resolvente SLDNF de P y G_i sobre l_s, G_{i+1}, es:

a) l_s literal positivo:
   - resolvente SLD de G_i y C_{i+1}, con C_{i+1} cláusula de programa cuya cabeza unifique con l_s

b) l_s literal negativo básico y existe prueba finita $P \not\models \neg l_s$
   - meta resultante de eliminar l_s de G_i

Programas normales

- Cláusulas Normales: admiten átomos negativos en el cuerpo de las cláusulas
- Negación: negación por fallo
- Regla de inferencia SLDNF
  - Similar a SLDNF con programas definidos y metas normales
  - Técnicamente, más compleja
  - Admite literales negativos con variables

Principales resultados en programas normales

- No se mantiene el teorema de independencia de la regla de cómputo
- SLDNF no es sólida

PROLOG

- Implementación secuencial modelo de programación lógica
- Programas normales
- Regla de cómputo: primer literal a la izquierda
- Regla de búsqueda: primero en profundidad
  - (implementación: backtracking)

Dificultades: complitud

- Prolog no es completo (por la regla de búsqueda)
  - Incluso en programas definidos puede no encontrar una refutación cuando esta existe.

Dificultades: solidez

- No es sólido, pues SLDNF no lo es
  - Sugerencia: evitar variables libres en literales negativos seleccionados
    - No incluyéndolos
    - Forzando ligadura operacionalmente
      - Aún así, problemas con significado declarativo

Desviaciones modelo lógico

- Ausencia chequeo de ocurrencias
  - Respuestas computadas no correctas
    - P={p(X, f(X)). q(a):-p(X,X).}, G=?-q(a).
  - Bucles infinitos
    - P={q(a):-p(X,X), p(X, f(X)):-p(X,X).}, G=?-q(a).

Desviaciones modelo lógico

- Corte: ! (cut)
  - Árbol SLD (programa P definido, meta G definida): árbol de derivaciones SLD para la meta G con el programa P
  - Efecto Corte: impide explorar algún subárbol
  - Afecta complitud programa definidos y normales
  - Afecta solidez programas normales(Prolog)