CLAUSOLE HORN DEFINITE: PROOF THEORY, SEMANTICA OPERAZIONALE, RISOLUZIONE SLD

Lloyd, Foundations of logic Programming

DERIVAZIONE SLD

P programma logico

$N = \leftarrow A_1, \dots, A_n$ goal

$C = A \leftarrow B_1, \dots, B_k$ clausola di P.

se per qualche i, $1 \le i \le n$, A_i e A unificano con mgu ϑ, il "nuovo" goal

$N' = \leftarrow (A_1, \dots, A_{i-1}, B_1, \dots, B_k, A_{i+1}, \dots, A_n)\vartheta$

è un risolvente di N e C con mgu ϑ.

- selezione dell'atomo A_i
- unificazione di A con A_i, che produce calcolo ϑ
- rimpiazzamento di A_i con $B_1, \dots, B_k$.
- applicazione dell'mgu ϑ alla clausola risultante.

- corrisponde esattamente alla definizione di risolvente data per generiche clausole

- iterando il processo di calcolo del risolvente, otteniamo una sequenza di risolventi chiamata derivazione SLD (spiegazione del nome nel seguito)

- SLD derivazione di $P \cup \{N\}$ è una sequenza massimale di goals $N_0, N_1, \dots$, dove $N_0 = N$, insieme ad una sequenza $C_0, C_1, \dots$ di varianti di clausole di P ed una sequenza $\vartheta_0, \vartheta_1, \dots$ di sostituzioni, tali che, per ogni $i = 0, 1, \dots$
  - N_{i+1} è un risolvente di N_i e C_i con mgu ϑ_i
  - C_i non ha variabili in comune con $N_0, C_0, \dots, C_{i-1}$
- $C_0, C_1, \dots$ sono le clausole di input
- se N_i è $\square$, la derivazione termina con successo e viene detta SLD-refutation
- una derivazione finita $\neq$ refutation è fallita

PERCHE' LE VARIANTI?

- la condizione

"C_i è una variante di una clausola di P che non ha variabili comuni con $N_0, C_0, \ldots, C_{i-1}$"

Viene detta standardization apart

- essenzialmente legata al fatto che non si vuole rendere il risultato di una derivazione dipendente da una particolare scelta dei nomi per le variabili

Esempio:

$\leftarrow p(X)$

da luogo ad una SLD-refutation nel programma

$p(f(X)) \leftarrow$

RISPOSTE CALCOLATE

- l'esistenza di una SLD-refutation di $P \cup \{N\}$, con $N = \leftarrow A_1, \dots, A_k$ significa che la negazione di N è conseguenza logica di P
  - $\neg N = \neg((\forall X_1)\dots(\forall X_r)(\neg A_1 \vee \dots \vee \neg A_k)) =$
    $= (\exists X_1)\dots(\exists X_r)(A_1 \wedge \dots \wedge A_k)$
- la sequenza di sostituzioni $\vartheta_0, \vartheta_1, \dots, \vartheta_m$ calcolate durante il processo di derivazione fornisce esattamente dei legami per le variabili $X_1, \dots, X_r$
  - come vedremo nel seguito, la SLD-refutation di $P \cup \{N\}$ può essere vista come una dimostrazione della formula $(A_1 \wedge \dots \wedge A_k)\vartheta_0\vartheta_1 \dots \vartheta_m$
- per questa ragione, la restrizione di $\vartheta_0\vartheta_1 \dots \vartheta_m$ alle variabili di N viene chiamata

computed answer substitution

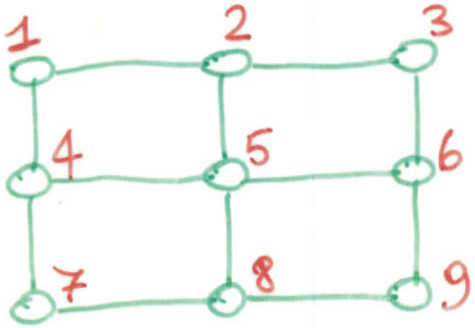
REGOLA DI SELEZIONE

- ad ogni passo di una derivazione SLD abbiamo due possibilità di scelta
  - l'atomo nel goal
  - le clausole nell'insieme di input, la cui testa (conclusione) unifica con l'atomo considerato.
- la prima scelta viene effettuata con la regola di selezione
- la scelta può, in generale, dipendere dall'intera storia della derivazione
  - sequenza $N_0, \dots, N_{K-1}$ di goals con atomi selezionati
  - N_K
  - $C_0, \dots, C_{K-1}$
  - $\vartheta_0, \dots, \vartheta_{K-1}$

  HIS = insieme di tutte le storie, con N_K non vuoto
- regola di selezione R: una funzione che applicata ad un elemento di HIS e a $N_K = \leftarrow A_1, \dots, A_t$, restituisce un A_i, $1 \le i \le t$.
  - si possono avere diverse selezioni in diverse occorrenze dello stesso risolvente
- SLD-derivazione di $P \cup \{N\}$ via R
  Tutte le scelte sono fatte con la regola di selezione R

SLD = **S**election rule driven **L**inear resolution for **D**efinite clauses (si dice anche LUSH)

UN ESEMPIO



- costruzione di un cammino fra due nodi di un grafo

- grafo

  arc (1,2) ←, arc (2,3) ←, ···· , arc (8,9) ← (1-6)
  arc (1,4) ←, arc (4,7) ←, ···, arc (6,9) ← (7-12)
  arc (X,Y) ← arc (Y,X) (13)

- stato iniziale

  path (1, nil) (14)

- cambiamento di stato

  path (X, cons(Y,L)) ← arc (X,Y), path (Y,L) (15)

- il goal (stato finale)

  ← path (5, L)

(1-6) arc(1,2)⇐ , arc(2,3)⇐ , ···· , arc(8,9)⇐

(7-12) arc(1,4)⇐ , arc(4,7)⇐ , ···· , arc(6,9)⇐

(13) arc(X,Y)⇐ arc(Y,X)

(14) path(1, nil)⇐

(15) path(X, cons(Y,L))⇐ arc(X,Y), path(Y,L)

UNA DERIVAZIONE SLD (SLD-refutation)

← path(5,L)	cl(15) $\{L \leftarrow L_1\}$	$\{X \leftarrow 5, L \leftarrow cons(Y, L_1)\}$
← arc(5,Y), path(Y,L1)	cl(13) $\{X \leftarrow X_1, Y \leftarrow Y_1\}$	$\{X_1 \leftarrow 5, Y_1 \leftarrow Y\}$
← arc(Y,5), path(Y,L1)	cl(3)	$\{Y \leftarrow 4\}$
← path(4,L1)	cl(15) $\{X \leftarrow X_2, Y \leftarrow Y_2, L \leftarrow L_2\}$	$\{X_2 \leftarrow 4, L_1 \leftarrow cons(Y_2, L_2)\}$
← arc(4,Y_2), path(Y_2,L_2)	cl(13) $\{X \leftarrow X_3, Y \leftarrow Y_3\}$	$\{X_3 \leftarrow 4, Y_3 \leftarrow Y_2\}$
← arc(Y_2,4), path(Y_2,L_2)	cl(7)	$\{Y_2 \leftarrow 1\}$
← path(1,L_2)	cl(14)	$\{L_2 \leftarrow nil\}$

□

- Viene sempre selezionato l'atomo più a sinistra
- la computed answer substitution è la composizione di

$\{L \leftarrow cons(Y, L1)\}, \{Y \leftarrow 4\}, \{L1 \leftarrow cons(Y_2, L_2)\}$
$\{Y_2 \leftarrow 1\}, \{L_2 \leftarrow nil\}$

$\{L \leftarrow cons(4, cons(1, nil))\}$

- la SLD-refutation è costruttiva
- la soluzione (risposta calcolata) viene resa disponibile in modo incrementale

UNA DERIVAZIONE SLD INFINITA

← path(5,L)	cl(15) {L←L1}	{X←5, L←cons(Y,L1)}
← arc(5,Y), path(Y,L1)	cl(13) {X←X1, Y←Y1}	{X1←5, Y1←Y}
← arc(Y,5), path(Y,L1)	cl(13) {X←X2, Y←Y2}	{X2←Y, Y2←5}
← arc(5,Y), path(Y,L1)		

- . . . . .

- con la stessa regola di selezione dell'atomo, ma scegliendo sempre la clausola (13)

- un'altra derivazione infinita, selezionando sempre l'atomo più a destra, per le clausole (15)

← path(5,L)
← arc(5,Y), path(Y,L1)
← arc(5,Y), arc(Y,Y1), path(Y1,L2)
← arc(5,Y), arc(Y,Y1), arc(Y1,Y2), path(Y2,L3)
. . . .

RISULTANTI E RISPOSTE CALCOLATE

- dato un goal $N = \leftarrow A_1, \ldots, A_k$

  $N^\vee$ denota la formula $A_1 \wedge \ldots \wedge A_k$.

  ($\square^\vee$ è la congiunzione vuota, true)

- data una derivazione SLD $N_0, N_1, \ldots$ con mgu's $\vartheta_0, \vartheta_1, \ldots$ di lunghezza $\geq i$,
  il risultante (di livello i) è la formula

  $$N_i^\vee \rightarrow N_0^\vee \vartheta_0 \ldots \vartheta_{i-1}$$

  (il risultante di livello 0 è $N_0^\vee \rightarrow N_0^\vee$)

- vogliamo dimostrare che la computed answer substitution di una derivazione SLD non dipende dalla scelta dei nomi delle variabili nelle clausole (sempre a meno di una ridenominazione)

- dimostreremo un risultato più forte
  i risultanti sono unici (a meno di varianza)

DALLE DERIVAZIONI SLD AGLI ALBERI SLD

- l'obiettivo delle SLD-resolution è trovare una SLD-refutation per il goal
  - per questo si costruiscono varie derivazioni SLD, il cui insieme forma lo spazio di ricerca
  - un modo di organizzare lo spazio di ricerca è quello di dividere le derivazioni SLD in gruppi, ciascuno caratterizzato da una specifica regola di selezione
  - alberi SLD

- P programma logico
  N goal
  R regola di selezione
  l'albero SLD per $P \cup \{N\}$ via R raggruppa tutte le derivazioni SLD di $P \cup \{N\}$ via R.
  - la radice è N
  - ogni nodo è un goal
  - i successori di un nodo sono tutti i suoi risolventi con (varianti di) clausole di P, con l'atomo selezionato in accordo con R

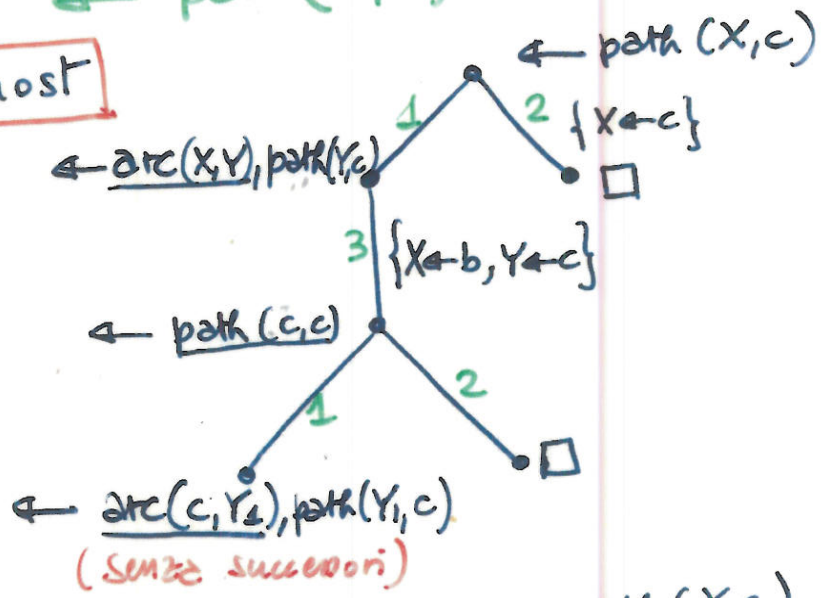
- un albero SLD ha successo se contiene la clausola vuota.

ALBERI SLD: UN ESEMPIO

1. path(X, Z) ← arc(X, Y), path(Y, Z)
2. path(X, X) ←
3. arc(b, c) ←

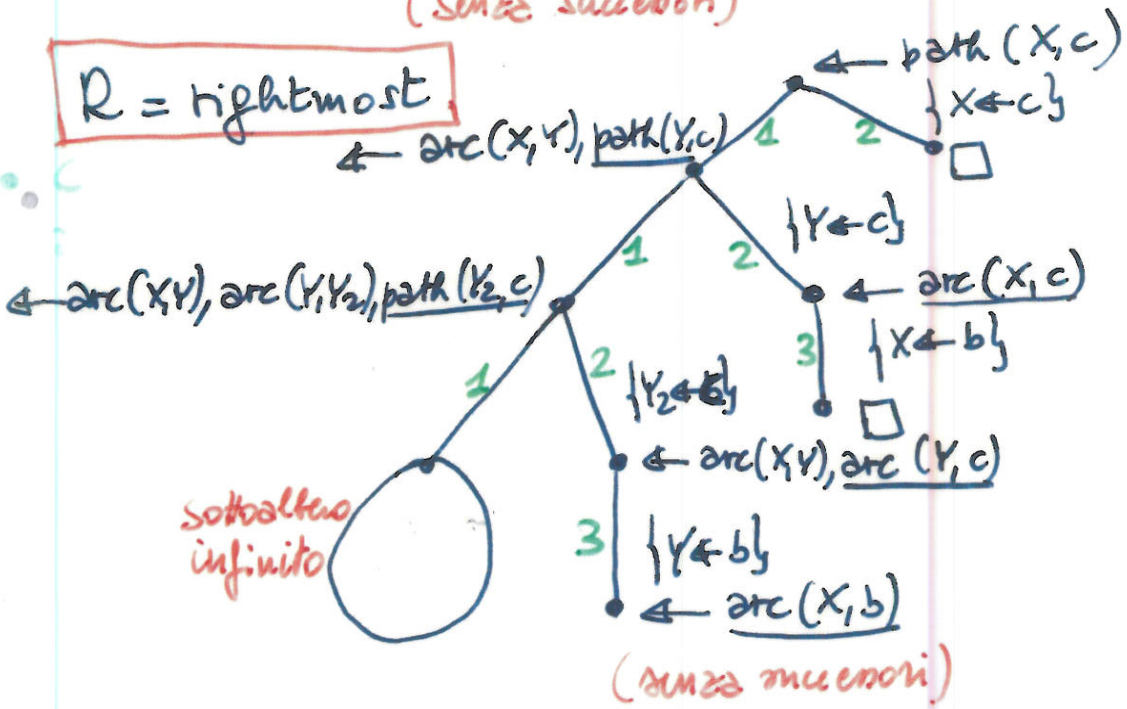
il goal ← path(X, c)

R = leftmost



(senza successori)

R = rightmost



(senza successori)

- Cambiando la regola di selezione, si possono ottenere alberi SLD molto diversi per forma e dimensione
  - finito vs infinito
  - le derivazioni SLD che terminano con la clausola vuota, sono (nell'esempio) "simili"

LA SEMANTICA OPERAZIONALE

- ne vedremo più avanti anche altre, questa è la più semplice e corrisponde alla più semplice proprietà osservabile

- programma P
  linguaggio L
  B_L base di Herbrand

$$Op(P) = \{ A \in B_L \mid P \cup \{\leftarrow A\} \text{ ha una SLD-refutation} \}$$

- si chiama <u>insieme di successo</u> (<u>success set</u>)

- è una interpretazione di Herbrand

  - è anche un modello?

CORRETTEZZA DELLA RISOLUZIONE SLD

programma logico P
goal $N = \leftarrow A_1, \dots, A_K$

- dal risultato generale di correttezza del principio di risoluzione, sappiamo che se esiste una SLD-refutation, la formula $(\exists X_1) \dots (\exists X_n)\ A_1 \wedge \dots \wedge A_K$ è conseguenza logica di P

- la correttezza ha, nel caso dei programmi logici, un significato più forte:
  - se $P \cup \{N\}$ ha una SLD-refutation caratterizzata dalla sequenza di sostituzioni $\vartheta_0, \dots, \vartheta_n$, allora è conseguenza logica di P la formula
    $(A_1 \wedge \dots \wedge A_K)\vartheta_0 \dots \vartheta_n$
    - le variabili di $(A_1 \wedge \dots \wedge A_K)\vartheta_0 \dots \vartheta_n$ sono ora universalmente quantificate
    - il discorso si può fare restringendo $\vartheta_0 \dots \vartheta_n$ alle variabili di N
    - la risposta calcolata ha un senso anche dal punto di vista <u>semantico</u> (o model-theoretic)

7

CLAUSOLE HORN DEFINITE
MODEL THEORY - SEMANTICA DICHIARATIVA

- Ogni insieme di clausole soddisfacibile ha un modello di Herbrand
  - ogni insieme di clausole Horn definite
    - è consistente
    - ha un modello di Herbrand minimo
  - il modello di Herbrand minimo può essere ottenuto come minimo punto fisso di una trasformazione da interpretazioni di Herbrand a interpretazioni di Herbrand
  - Il modello di Herbrand minimo contiene l'insieme di successo
- cominceremo con l'introduzione di un operatore su interpretazioni di Herbrand.

L'OPERATORE DELLE CONSEGUENZE IMMEDIATE

[Van Emden & Kowalski, 1976]

P : programma logico

I : una generica interpretazione di Herbrand
(sottoinsieme della base di Herbrand B_P)

$$T_P : 2^{B_P} \to 2^{B_P}$$

$$T_P(I) = \{ A \in B_P \mid \bullet\ A \leftarrow A_1, \dots, A_n \text{ è un'istanza ground di una clausola in } P \ \bullet\ \{A_1, \dots, A_n\} \subseteq I \}$$

$$= \{ A \in B_P \mid \bullet\ \text{esiste una clausola } A' \leftarrow A_1, \dots, A_n \text{ in } P \ \bullet\ \text{esiste una sostituzione } \vartheta \text{ per cui } \bullet\ A \equiv A'\vartheta \ \bullet\ \{A_1\vartheta, \dots, A_n\vartheta\} \subseteq I \}$$

- perché T_P si chiama "delle conseguenze immediate"?
- da notare, che se P contiene una clausola $A \leftarrow$, allora $T_P(I)$ contiene tutte le istanze ground di A, qualunque sia I

- cosa c'entra l'operatore T_P con i modelli di Herbrand?

T_P E MODELLI DI HERBRAND DI P

Teorema dati un programma logico P ed una interpretazione di Herbrand I,
I è un modello di P se e solo se
$T_P(I) \subseteq I$

prova:

- I è un modello di P (cioè se e solo se) sono vere in I tutte le istanze ground di clausole di P
- sia $A \leftarrow A_1, \dots, A_n$ una tale istanza ground
  - $\{A_1, \dots, A_n\} \subseteq I$ implica che $A \in I$ se e solo se
    $T_P(I) \subseteq I$

- se $T_P(I) \subseteq I$, I è un <u>pre-punto-fisso</u> di T_P
  - per studiare i modelli di Herbrand di P, basta studiare i pre-punti-fissi di T_P

- cominciamo con il ricordare un po' di proprietà generali di operatori e loro punti fissi.

IL NOSTRO SCENARIO

abbiamo:

- l'insieme delle interpretazioni di Herbrand
- un operatore su tale insieme (T_P)

sappiamo già che:

- i modelli di Herbrand ed i pre-punti-fissi di T_P sono la stessa cosa

la teoria generale ci dice che:

- se l'insieme delle H-interpretazioni è un reticolo completo
- se T_P è monotono, esiste

  $lfp(T_P)$ = minimo pre-punto-fisso (T_P) = modello di Herbrand minimo

- se T_P è anche continuo,

  $lfp(T_P) = T_P \uparrow \omega$

inoltre sappiamo che (correttezza della SLD resolution):

- se $\leftarrow G$ (ground) è refutabile in P, G è conseguenza logica di P
- se $G \in \mathcal{O}_P$ (insieme di successo), G è conseguenza logica di P

se dimostriamo che

- $lfp(T_P) \equiv \{$ conseguenze logiche di P atomiche ground$\}$

abbiamo che

- $\mathcal{O}_P \subseteq lfp(T_P) = F_P = M_P$

(operazionale; fixpoint; model-theoretic)

IL RETICOLO COMPLETO DELLE H- INTERPRETAZIONI

P programma logico

B_P base di Herbrand di P

2^{B_P} insieme delle interpretazioni di Herbrand

ordine parziale: inclusione fra insiemi $\subseteq$

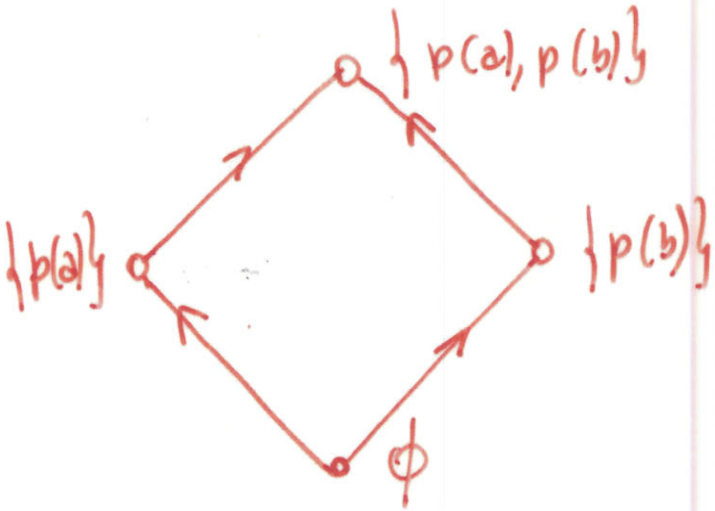
$$lub(X) \equiv \bigcup_{x_i \in X} x_i$$

$$glb(X) \equiv \bigcap_{x_i \in X} x_i$$

$$\top \equiv B_P$$

$$\bot \equiv \phi$$

Esempio: $B_P = \{p(a), p(b)\}$



L'OPERATORE DELLE CONSEGUENZE IMMEDIATE E' CONTINUO

$$T_P(I) = \{ A \in B_P \mid A \leftarrow A_1, \dots, A_n \text{ è un'istanza ground di clausola di } P, \{A_1, \dots, A_n\} \subseteq I \}$$

teorema: T_P è continuo, cioè
$T_P(\text{lub}(X)) = \text{lub}(T_P(X))$ per ogni diretto X di 2^{B_P}

lemma: $\{A_1, \dots, A_n\} \subseteq \text{lub}(X)$ sse $\{A_1, \dots, A_n\} \subseteq I$ per qualche $I \in X$

prova teorema

$A \in T_P(\text{lub}(X))$ se e solo se

$A \leftarrow A_1, \dots, A_n$ è un'istanza ground di clausola in P e $\{A_1, \dots, A_n\} \subseteq \text{lub}(X)$ se e solo se

$A \leftarrow A_1, \dots, A_n$ è un'istanza ground di clausola in P e $\{A_1, \dots, A_n\} \subseteq I$ per qualche $I \in X$ se e solo se

$A \in T_P(I)$ per qualche $I \in X$ se e solo se

$A \in \text{lub}(T_P(X))$

corollario: T_P è monotono

IL TEOREMA DI CARATTERIZZAZIONE

[van Emden & Kowalski, 1976]

P programma logico,

P ha un modello di Herbrand M_P con le seguenti proprietà

- M_P è il minimo modello di Herbrand di P
- M_P è il minimo pre-punto-fisso di T_P
- M_P è il minimo punto fisso di T_P
- $M_P = T_P \uparrow \omega$

Se
- dato che T_P è monotono, esiste $lfp(T_P)$ che è sia il minimo punto fisso che il minimo pre-punto-fisso
- i pre-punti-fissi sono i modelli di Herbrand $\Rightarrow$ $lfp(T_P)$ è il minimo modello di Herbrand
- dato che T_P è anche continuo, $lfp(T_P)$ può essere ottenuto come potenza ordinale $T_P \uparrow \omega$

Corollario: P programma logico, M = intersezione di tutti i modelli di Herbrand di P,

$$M = M_P$$

- dal teorema di Tarski

$$M_P = lfp(T_P) = glb\{x : T_P(x) \subseteq x\} =$$
$$= glb\{x : x \text{ è un H-modello di } P\} =$$
$$= \bigcap \{x : x \text{ è un H-modello di } P\} = M$$

NON ESISTENZA DEL MODELLO MINIMO PER CLAUSOLE GENERALI

Un controesempio (basta il calcolo proposizionale!)

$P = \{A \vee B\}$

tre modelli di Herbrand

$\{A\}$ $\{B\}$ $\{A, B\}$

la loro intersezione è $\{\}$ che non è un modello di Herbrand di P

UN'ALTRA PROPRIETÀ DEL MODELLO MINIMO DI HERBRAND

P programma logico

M_P = minimo modello di Herbrand di P

$M_P = \{A \in B_P : A \text{ è conseguenza logica di } P\}$

prova

A è conseguenza logica di P
se e solo se
$P \cup \{\neg A\}$ è insoddisfacibile
se e solo se
$P \cup \{\neg A\}$ non ha modelli di Herbrand
se e solo se
$\neg A$ è falso in tutti i modelli di Herbrand di P
se e solo se
A è vero in tutti i modelli di Herbrand di P
se e solo se

$A \in M_P$ (intersezione di tutti i modelli di Herbrand di P)

LE FAMOSE 3 SEMANTICHE DEI PROGRAMMI LOGICI

Operazionale

$$O_P = \{ A \in B_P \mid P \cup \{\leftarrow A\} \text{ ha una SLD-refutation} \}$$

punto fisso

$$F_P = lfp(T_P) = T_P \uparrow \omega$$

model-theoretic

M_P = minimo modello di Herbrand = F_P

Spesso chiamate semantiche dichiarative

- l'interpretazione canonica di un programma! (quella intesa dal programmatore?)
- l'onere di Herbrand è ragionevole perché il programmatore normalmente ha in mente l'interpretazione libera di costanti e funzioni (strutture dati)
- d'altra parte, abbiamo visto che M_P è esattamente l'insieme di atomi ground conseguenza logica di P
- che relazione c'è tra O_P da una parte e F_P e M_P dall'altra?

CORRETTEZZA DELLA SEMANTICA OPERAZIONALE RISPETTO A QUELLA DICHIARATIVA

Teorema: *L'insieme di successo è contenuto nel modello minimo di Herbrand,*

$$O_P \subseteq M_P$$

prova

$A \in O_P$

$\Leftarrow$ A è rifiutabile, $A \in B_P$

$\Downarrow$

A è conseguenza logica di P, $A \in B_P$

$\Updownarrow$

$A \in M_P$

QUALCHE ESEMPIO

$p(0, X, X) \leftarrow$

$p(f(X), Y, f(Z)) \leftarrow p(X, Y, Z)$

$B_P = \{ p(0,0,0),\ p(0,0,f(0)),\ \ldots\ldots \}$

$T \uparrow 0 = \emptyset$

$T \uparrow 1 = \{ p(0,0,0),\ p(0, f(0), f(0)),\ p(0, f(f(0)), f(f(0))),\ \ldots \}$

$T \uparrow 2 = \{ p(0,0,0),\ p(f(0), 0, f(0)),$
$p(0, f(0), f(0)),\ p(f(0), f(0), f(f(0))),$
$p(0, f(f(0)), f(f(0))),\ p(f(0), f(f(0)), f(f(f(0)))), \ldots \}$

$\vdots$

$T \uparrow \omega = M_P = \{ p(f^n(0), Y, f^n(Y)) \mid$
$Y = 0, f(0), \ldots\ldots$
$n = 0, \omega$

QUALCHE ESEMPIO

path(X,Z) ← arc(X,Y), path(Y,Z)
path(X,X) ←
arc(b,c) ←

$B_P = \{arc(b,c), arc(c,b), arc(b,b), arc(c,c), path(b,c), path(c,b), path(b,b), path(c,c)\}$

- $T_P \uparrow 0 = \phi$
- $T_P \uparrow 1 = \{arc(b,c), path(b,b), path(c,c)\}$
- $T_P \uparrow 2 = \{arc(b,c), path(b,b), path(c,c), path(b,c)\}$
- $T_P \uparrow 3 = T_P \uparrow 2 = lfp(T_P) = M_P$

- si può controllare facilmente che coincide anche con l'insieme di successo
  - infatti $O_P = \{arc(b,c), path(b,b), path(c,c), path(b,c)\}$

8 COMPLETEZZA DELLA SLD-resolution ed altre proprietà operazionali

- abbiamo già visto che
  - la risoluzione SLD è corretta
    se il goal $\leftarrow A_1, \dots, A_k$ ha una SLD-refutation in P, allora la formula
    $(\exists X_1) \dots (\exists X_n)\, A_1 \wedge \dots \wedge A_k$ è conseguenza logica di P
  - le risposte calcolate sono "corrette"
    se $\leftarrow A_1, \dots, A_k$ ha una SLD-refutation con computed answer substitution ϑ, allora la formula
    $(A_1 \wedge \dots \wedge A_k)\vartheta$ è conseguenza logica di P
  - la semantica operazionale (insieme di successi)
    è inclusa in quella dichiarativa
    se ci limitiamo a considerare gli atomi ground, quelli rifiutabili sono tutti contenuti proprio nel modello minimo di Herbrand

- affronteremo ora le proprietà di completezza
  - la risoluzione SLD è completa, cioè se $P \cup \{\leftarrow A\}$ è inconsistente, esiste una SLD-refutation
  - la semantica dichiarativa è inclusa in quella operazionale
  - per ogni correct answer substitution ϑ (vedi dopo), esiste una computed answer substitution che è più generale di ϑ

LA COMPLETEZZA RAFFORZATA

R-insieme di successo $O_P^R = \{A \in B_P \mid P \cup \{\neg A\}$ ha una SLD-refutation via $R\}$

(equivalenza [forte] delle semantiche)

P propsemme R regole di selezione

$O_P^R = M_P$

(completezza 1 [forte] goal non ground)

P propsemme N goal R regole di selezione

se $P \cup \{N\}$ è successdisprovabile, allora esiste una SLD-refutation di $P \cup \{N\}$ via R

(completezza 2 [forte] risposte calcolate) =

Completezza forte della risoluzione SLD

P propsemme N goal R regole di selezione

se σ è una sostituzione di risposta corretta per $P \cup \{N\}$ esiste una SLD-refutation di $P \cup \{N\}$ via R, tale che la sua sostituzione di risposta calcolata è più generale di σ.

PROPRIETA' DI COMPLETEZZA E ALBERI SLD

- Ricordiamo che un albero SLD per $P \cup \{N\}$ rappresenta tutte le SLD derivazioni via una particolare regola di selezione R.
  - i cammini dell'albero SLD sono esattamente le derivazioni di $P \cup \{N\}$ via R

- i teoremi di completezza 1 e 2 [rafforzati] possono essere ripresi in termini di proprietà degli alberi SLD.

(teorema di completezza forte della risoluzione SLD)

P programma N goal R regola di selezione

Se $P \cup \{N\}$ è insoddisfacibile, l'albero SLD di $P \cup \{N\}$ via R ha almeno un cammino di successo.

- un altro interessante teorema che si può dimostrare ricorrendo allo stesso lemma

P programma N goal

ogni albero SLD di $P \cup \{N\}$ ha infiniti cammini di successo, oppure ogni albero SLD di $P \cup \{N\}$ ha lo stesso numero finito di cammini di successo

IL TEOREMA DEI SUCCESSI

riassume in un unico teorema le proprietà (operazionali e dichiarative) per

- atomi ground
- successo

programma P atomo $A \in B_P$

le seguenti affermazioni sono equivalenti

a) A è conseguenza logica di P

b) $A \in T_P \uparrow \omega = lfp(T_P) = M_P$

c) A appartiene all'insieme di successo di P
$A \in O_P = O_P^R$ per R qualunque

d) ogni albero SLD di radice in $\leftarrow A$ ha almeno un cammino di successo

- niente atomi non ground
- niente sostituzioni di risposta calcolate
- non prende in considerazione i fallimenti finiti, che sono certamente un comportamento osservabile.

LA REGOLA DI RICERCA

- una strategia di ricerca all'interno di un albero SLD, con lo scopo di trovare un cammino di successo

- le regole di ricerca più comuni

  - depth-first
    si sceglie un cammino (per esempio, con un ordinamento fra le clausole) e si continua in profondità fino a che si raggiunge la clausola vuota oppure un goal non più risolvibile (fallimento finito). In questo caso, si pensa ad altro cammino. Con il backtracking si ritorna all'ultimo punto di scelta.
  - breadth-first
    si sviluppano tutti i cammini un po' alla volta (pseudo-parallelismo) fino a che non si raggiunge la clausola vuota

- le regole depth-first con backtracking e ordinamento fra le clausole fisso è molto semplice da implementare e porta ad implementazioni efficienti.

- le regole depth-first non è fair, cioè non è garantito trovare un cammino di successo, se l'albero SLD ne contiene uno (vedremo più avanti una definizione di fairness relativa alle regole di selezione!)

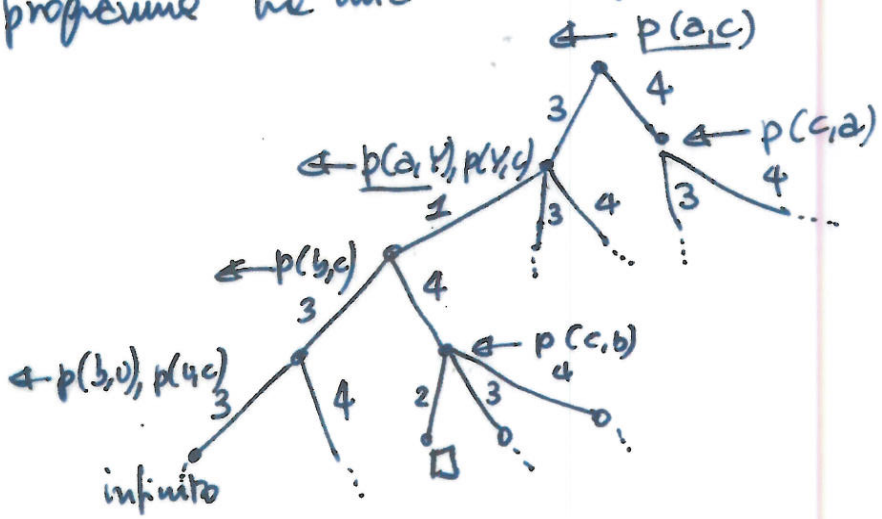
- con alberi SLD infiniti è necessario avere regole di ricerca con una componente breadth-first

- non è sufficiente il ricorso a diverse regole di selezione o a modifiche nell'ordinamento delle clausole.

UN BRUTTO ESEMPIO PER LE REGOLE DI RICERCA DEPTH FIRST

1. $p(a,b) \leftarrow$
2. $p(c,b) \leftarrow$
3. $p(X,Z) \leftarrow p(X,Y), p(Y,Z)$
4. $p(X,Y) \leftarrow p(Y,X)$
5. $\leftarrow p(a,c)$

- il programma ha una SLD-refutation (R = leftmost)



- il cammino più a sinistra è ∞ $\Rightarrow$ depth-first non trova il cammino di successo
- nelle refutazioni (in realtà, in ogni refutazione) sono necessarie sia le clausole 3 che le 4
- con un'etichettamento fino alle clausole, nel "primo cammino" o si usa solo le 3 o solo le 4 perché entrambi con gli stessi pedi
- tutte due le clausole da sole generano cammini ∞.
-