Università degli Studi di Bari
C.I.L.A.- Centro Interdipartimentale di Logica e Applicazioni

ESPLORAZIONE DI SPAZI DI RICERCA IN ILP

dott.ssa Teresa M. A. Basile
basile@di.uniba.it

Sommario

- ILP e modelli di generalizzazione
- Operatori di raffinamento
- Proprietà degli Operatori di raffinamento
- Condizioni di inesistenza di operatori di raffinamento ideali
- Condizioni di inesistenza valide per θ-sussunzione
- Condizioni di inesistenza non valide per θ_{OI}-sussunzione

ILP

La Programmazione Logica Induttiva è un'area di ricerca che si pone all'intersezione dell'apprendimento automatico e la programmazione logica [Muggleton,91].

La ILP estende la teoria e la pratica della logica computazionale investigando l'induzione come modello di base per l'inferenza piuttosto che la deduzione. Mentre l'attuale teoria della logica computazionale descrive inferenze deduttive da formule logiche fornite dall'utente, la teoria della programmazione logica induttiva descrive inferenze induttive di programmi logici da istanze e conoscenza di background.

Tipico Task ILP

Dati:

- una conoscenza di fondo (Background Knowledge - *BK*),
- un insieme di esempi positivi E^+ e negativi E^- relativi ad un concetto target *C*

Trovare:

una ipotesi *H* che sia *completa* e *consistente*

- ogni esempio positivo $e \in E^+$, relativo al concetto *C*, sia spiegato da $H \cup BK$ *(completezza)*
- nessun esempio negativo $e \in E^-$, contro-esempio per il concetto *C*, sia spiegato da $H \cup BK$ *(consistenza)*

Tools in ILP

- Le osservazioni (fatti ground), la conoscenza di fondo e le ipotesi ((insieme di) clausole definite) sono espresse nella logica del primo ordine
- Apprendimento Automatico Induttivo da Esempi: elaborazione di esempi al fine di *indurre* regole o una teoria che li spieghino.

ILP come ricerca di programmi clausali

- **Stati**: ipotesi costruite sul linguaggio di rappresentazione utilizzato (*spazio delle soluzioni candidate*);
- **Obiettivo**: trovare una ipotesi che soddisfi criteri di qualità, ad esempio completezza e consistenza;
- **Operatori**: per passare da uno stato all'altro

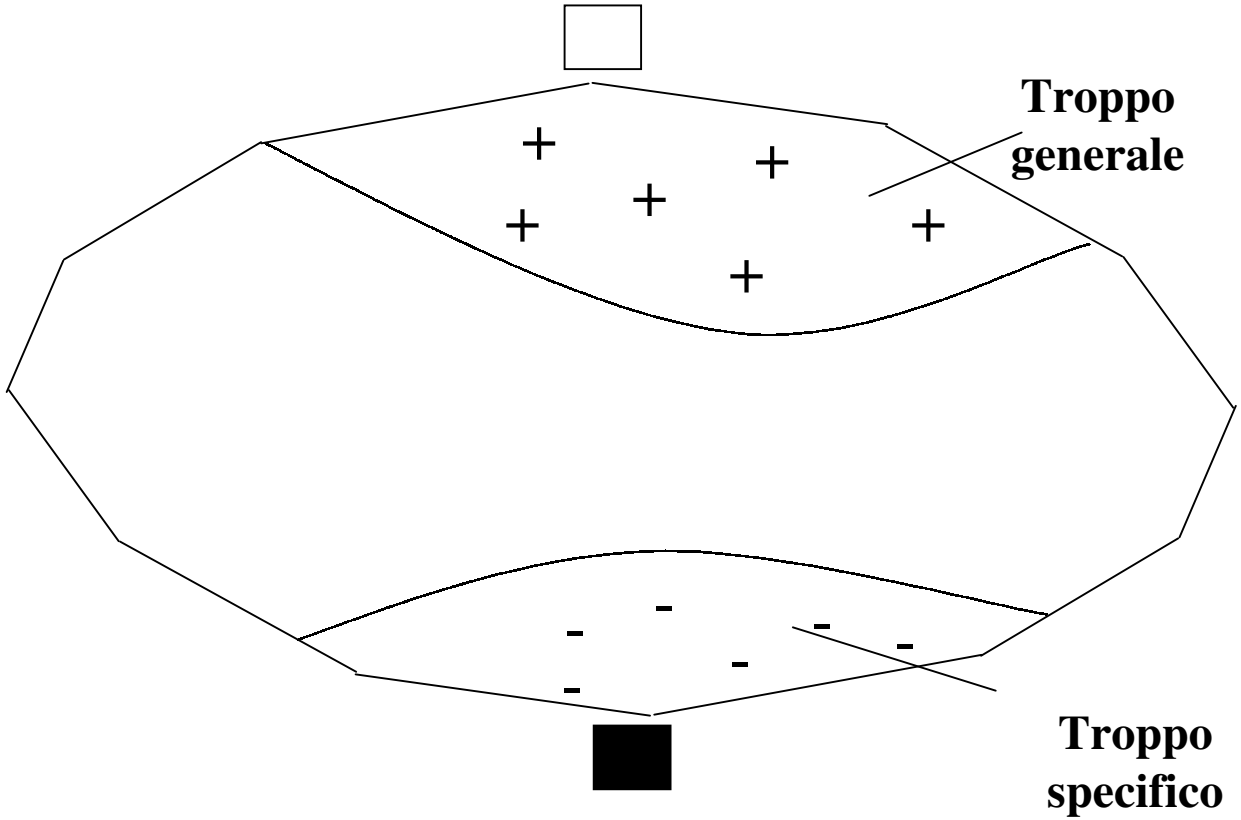
ILP come ricerca di programmi clausali

Si potrebbe pensare di risolvere il problema ILP con un semplice algoritmo *generate and test*.

Questo approccio è noto in letteratura come algoritmo di enumerazione che però risulta essere computazionalmente costoso per essere di interesse pratico.

Quindi, la questione si riduce a come *lo spazio delle possibili soluzioni dovrebbe essere strutturato in modo da consentire un pruning della ricerca*.

Spazio delle ipotesi: modello del Version Space



Come Strutturare lo spazio delle ipotesi?

Come Esplorare lo spazio?

Strutturare lo spazio delle ipotesi: Linguaggio di rappresentazione

- Logica del primo ordine
  - Il costo di una esplorazione esaustiva dello spazio delle ipotesi renderebbe l'apprendimento infattibile

Si ha bisogno di introdurre dei *bias* sul linguaggio che consentano di effettuare un *pruning* sullo spazio.

I bias risultano fondamentali tanto per il modo in cui la **conoscenza** dovrebbe essere rappresentata sia per quanto riguarda gli **algoritmi di esplorazione** dello spazio delle soluzioni impiegati per risolvere il problema.

Strutturare lo spazio delle ipotesi: Restrizioni sul Linguaggio di rappresentazione

- Trascurare alcune ipotesi che risultano non interessanti per l'obiettivo da raggiungere
  - Clausole Datalog: clausole di Horn function-free (termini consentiti: costanti o variabili) e linkate
- Conferire allo spazio di ricerca delle soluzioni proprietà teoriche per la definizione di operatori di ricerca "*validi*"
  - Assunzione di Object Identity (***In una clausola termini denotati con simboli diversi devono essere distinti, ovvero devono rappresentare oggetti distinti del dominio***)

Definizione (Linkedness):

Una clausola di Horn (Datalog) è linkata se tutti i suoi letterali sono linkati.

Un letterale è linkato se almeno uno dei suoi argomenti è linkato.

Un argomento di un letterale è linkato se il letterale è positivo (rappresenta la testa della clausola) o se un altro argomento dello stesso letterale è linkato.

Linkate: C_1 = bicycle(X) $\leftarrow$ wheel (Y,X), rim (Z,Y)

Non Linkate:

C_2 = bicycle(X) $\leftarrow$ rim (Z,Y)

C_3 = bicycle(X) $\leftarrow$ wheel (Y,X), rim (Z,Y), engine (V,W)

Equivalenza fra Clausole di Horn e Datalog

Definizione (Flattening) [Rouveirol, 92]

Dato un funtore n-ario *f*, chiamiamo *predicato di flattening* (o *di appiattimento*) f_p il predicato di arità n+1 così definito:

$$f_p\,(t_1, t_2, \ldots, t_n, X) \text{ :- } X = f(t_1, t_2, \ldots, t_n,)$$

Data una clausola di un programma logico P, definiamo *clausola flattened* (o *appiattita*) C′ la clausola ottenuta da C sostituendo ogni occorrenza di un termine con simbolo funzionale *f* con una nuova variabile *X* e aggiungendo nel suo corpo $f_p\,(t_1, t_2, \ldots, t_n, X)$, dove f_p è il predicato di flattening corrispondente ad *f*

(Procedura inversa: unflattening)

Equivalenza fra Clausole di Horn e Datalog

Il flattening è un cambio di rappresentazione che consente di tradurre un insieme di clausole contenti simboli funzionali in uno, ad esso semanticamente equivalente, costituito da clausole prive di tali simboli.

Esempio: data la clausola

E = member(blue,[blue]):-

clausola appiattita:

E' = member(X,Z):- $cons_p$(X,Y,Z), $blue_p$(X), nil_p(Y)

le tre clausole che definiscono i predicati di flattening

C = NIL: nil_p(nil):-

C′ = CONS: $cons_p$(X,Y,cons(X,Y)):-

C″ = BLUE: $blue_p$(blue):-

risolvendo E' con C,C′,C″ si ottiene E – semantica preservata

Equivalenza fra Datalog e Datalog^{OI}

Data la clausola Datalog

$C = p(X) \leftarrow q(X,X), q(Y,a)$

La seguente clausola, C', è la riscrittura di C sotto l'assunzione di Object Identity (Clausola Datalog^{OI})

$C' = p(X) \leftarrow q(X,X), q(Y,a), \neq (X,Y), \neq(X,a), \neq(Y,a)$

Equivalenza fra Datalog e Datalog^{OI}

Proposizione:

$\forall C \in$ Datalog $\exists C' = \{C_1, C_2, \ldots, C_n\} \subseteq$ Datalog^{OI} :
$T_C\uparrow = T_{C'}\uparrow$ (per ogni clausola Datalog è possibile individuare un insieme di clausole Datalog^{OI} che risulta semanticamente equivalente ad essa)

Corollario: $\forall P \subseteq$ Datalog $\exists P' \subseteq$ Datalog^{OI} :
$T_P\uparrow = T_{P'}\uparrow$

Equivalenza fra Datalog e Datalog^{OI}

Data la clausola Datalog

$C = p(X) \leftarrow q(X,X), q(Y,a)$

$C' = \{C_1, C_2, C_3, C_4, C_5\} \subseteq$ Datalog^{OI} è l'insieme di clausole equivalenti a C sotto l'assunzione di Object Identity:

$C_1 = p(X) \leftarrow q(X,X), q(Y,a), \neq (X,Y), \neq(X,a), \neq(Y,a)$

$C_2 = p(X) \leftarrow q(X,X), q(X,a), \neq(X,a)$

$C_3 = p(a) \leftarrow q(a,a), q(Y,a), \neq(Y,a)$

$C_4 = p(X) \leftarrow q(X,X), q(a,a), \neq(X,a)$

$C_5 = p(a) \leftarrow q(a,a), q(a,a)$

Esplorare lo spazio: la struttura

Una teoria generale per gli operatori di raffinamento non può essere formulata prescindendo dalla nozione di ordinamento definita sugli elementi dello spazio di ricerca.

Definizione (Ordine di Generalità):
Un ordine di generalità è un *quasi-ordinamento* $\geq$, cioè una relazione binaria riflessiva e transitiva, tale che, dati due oggetti dell'insieme C e D,

$$C \geq D \text{ implica che } C \models D$$

Esplorare lo spazio: dalla deduzione all'induzione

- Processo deduttivo
  - Mezzo: *unificazione*
  - Scopo: determinare l'unificatore più generale (mgu)
- Processo Induttivo
  - Mezzo: operazione duale dell'unificazione - *la generalizzazione*
  - Scopo: Determinare la generalizzazione meno generale (lgg)

Esplorare lo spazio: gli operatori

Nel concept-learning e ILP lo spazio di ricerca è tipicamente strutturato tramite le due nozioni duali di generalizzazione e specializzazione.

Definizione (generalizzazione e specializzazione):
Dato un ordine di generalità ≥, e due clausole C e D,
C ≥ D, si dirà che C è più generale di D e D è più specifica di C.
Si dirà anche che C è una generalizzazione di D o D è una specializzazione di C.

Modelli di generalizzazioni e proprietà

Implicazione logica: è il modello più naturale di generalizzazione essendo l'induzione definita in termini di conseguenza logica

- è un quasi-ordinamento
- $C \Rightarrow D$ indecidibile
- $C \Leftrightarrow D$ senza necessariamente essere varianti alfabetiche
- C è una riduzione di D indecidibile
- lgg_I di un insieme finito di clausole non è calcolabile
- lgg_I non è unica (a causa del quasi-ordinamento)

Modelli di generalizzazioni e proprietà

T-Implicazione: indebolimento della implicazione logica

- non è un quasi-ordinamento
- $C \Rightarrow_T D$ decidibile (essendo definita in termini di insieme finito di istanze ground di clausole)
- $C \Leftrightarrow_T D$ senza necessariamente essere varianti alfabetiche
- C è una riduzione di D decidibile
- lgg_T di un insieme finito di clausole è calcolabile

Modelli di generalizzazioni e proprietà

θ-sussunzione : indebolimento della T-implicazione

- è un quasi-ordinamento
- $C \leq_\theta D$ - decidibile (ma NP-completo)
- $C \sim_\theta D$ senza necessariamente essere varianti alfabetiche
- C è una riduzione di D - decidibile
- $C \sim_\theta D$ ridotte sono necessariamente varianti alfabetiche
- lgg_θ di un insieme finito di clausole è calcolabile [Plotkin]
- lgg_θ è unica

Modelli di generalizzazioni e proprietà

θ_{OI}-sussunzione: indebolimento della **θ-sussunzione**

- è un quasi-ordinamento
- $C \leq_{\theta_{OI}} D$ decidibile
- $C \sim_{\theta_{OI}} D$ sono necessariamente varianti alfabetiche
- C è una riduzione di D - decidibile ($O(n^2)$)
- $lgg_{\theta_{OI}}$ di un insieme finito di clausole è calcolabile
- $lgg_{\theta_{OI}}$ non è unica

Gerarchia e relazioni tra i modelli di generalità

$\Rightarrow$

Se $C \Rightarrow_T D$ allora $C \Rightarrow D$

Se $C \Rightarrow D$ allora esiste un term set di $\{D\}$ t.c. $C \Rightarrow_T D$

$\Rightarrow_T$

Se $D \leq_\theta C$ allora $C \Rightarrow_T D$ da cui

Se $D \leq_\theta C$ allora $C \Rightarrow D$

θ

Se $D \leq_{\theta_{OI}} C$ allora $D \leq_\theta C$

θ_{OI}

Operatori di raffinamento: Perché

Uno dei problemi principali della ILP è la definizione di operatori di raffinamento "*ideali*" che attuino una revisione della teoria nel caso in cui essa risulti non corretta perché incompleta o inconsistente con gli esempi forniti.

La definizione di operatori di raffinamento risulta fortemente dipendente dallo spazio di ricerca e più in particolare dal modello di generalizzazione definita su di esso.

Operatori di Raffinamento: Tipologia

- Teoria inconsistente: [copre esempi negativi]
  - Procedura di specializzazione: *Operatore di raffinamento downward*
- Teoria incompleta: [non copre esempi positivi]
  - Procedura di generalizzazione: *Operatore di raffinamento upward*

Operatori di Raffinamento: *downward*

Dato un insieme quasi-ordinato $(S, \leq)$ e $C, D, E \in S$:

- ρ è un operatore di raffinamento downward se:

$$\forall\, C \in S\ \rho(C) \subseteq \{D \in S \mid D \leq C\}$$

Chiusura dell'operatore definita come:

- $\rho^0(C) = \{C\}$;
- $\rho^n(C) = \{D \mid \exists\, E \in \rho^{n-1}(C) \text{ e } D \in \rho(E)\ \}$;
- $\rho^*(C) = \cup_{n \geq 0}\, \rho^n(C) = \rho^0(C) \cup \rho^1(C) \cup \ldots\ \rho^i(C) \cup \ldots$

Operatori di Raffinamento: *upward*

Dato un insieme quasi ordinato $(S, \leq)$ e $C,D,E\ i \in S$:

- δ è un operatore di raffinamento upward se:

$$\forall\ C \in S\ \delta(C) \subseteq \{D \in S \mid C \leq D\}$$

Chiusura dell'operatore definita come:

- $\delta^0(C) = \{C\}$;
- $\delta^n(C) = \{D \mid \exists\ E \in \delta^{n-1}(C) \text{ e } D \in \delta(E)\ \}$;
- $\delta^*(C) = \cup_{n \geq 0}\ \delta^n(C) = \delta^0(C) \cup \delta^1(C) \cup \ldots\ \delta^i(C) \cup \ldots$

Operatori di Raffinamento: Proprietà [van der Laag & Nienhuys-Cheng, 94-95]

- Definizione (Completo)

  In un insieme quasi-ordinato $(S, \leq)$, un operatore di raffinamento downward ρ (risp. upward δ) è completo se e solo se $\forall C, D \in S$,
  $D < C$ implica che $\exists E \in S$: $E \in \rho^*(C)$ e $E \sim D$
  (risp. $C < D$ implica che $\exists E \in S$: $E \in \delta^*(C)$ e $E \sim D$)

Ogni passo di raffinamento di una clausola può essere connesso ad essa mediante una catena finita di raffinamenti ad un passo

Operatori di Raffinamento: Proprietà [van der Laag & Nienhuys-Cheng, 94-95]

- Definizione (Localmente finito)

  In un insieme quasi-ordinato $(S, \leq)$, un operatore di raffinamento τ (downward/upward) è localmente finito se e solo se $\forall C \in S$, $\tau(C)$ è un insieme finito e computabile

  L'insieme dei raffinamenti ottenibili in un passo, per ogni clausola appartenente allo spazio di ricerca, è finito e computabile

Operatori di Raffinamento: Proprietà [van der Laag & Nienhuys-Cheng, 94-95]

- Definizione (Proprio)

  In un insieme quasi-ordinato $(S, \leq)$, un operatore di raffinamento downward ρ (risp. upward δ) è proprio se e solo se
  $\forall C \in S, D \in \rho(C): D < C$
  (risp. $\forall C \in S, D \in \delta(C): C < D$)

  Ogni passo di raffinamento non genera mai una clausola equivalente a quella di partenza

Operatori di Raffinamento: Proprietà

- Completo
  - Garanzia di riuscire a trovare tutte le clausole di un programma logico mediante specializzazioni di clausole più generali o generalizzazioni di clausole più specifiche
- Localmente finito
  - Garanzia di computabilità e di terminazione del processo di raffinamento
- Proprio
  - Garanzia di generare sempre un raffinamento che non sia una clausola equivalente a quella di partenza: non ne trarremmo nessun vantaggio ai fini del raffinamento

Idealità

Definizione

In un insieme quasi-ordinato $(S, \leq)$, un operatore di raffinamento downward ρ (risp. upward δ) è ideale se e solo se esso è localmente finito, proprio e completo

Relazione tra idealità e ricoprimento downward/upward

Durante la fase di ricerca di un programma logico, è importante che

- ogni passo di raffinamento sia "*abbastanza grande*" affinché ogni specializzazione o generalizzazione sia trovata in un numero finito di passi,
- il passo sia "*limitato*" affinché nessuna clausola sia persa durante il processo di raffinamento.

L'idealità di un operatore di raffinamento implica che l'insieme dei raffinamenti ottenibili in un passo includano tutte le specializzazioni o le generalizzazioni proprie e non esistano altre clausole tra le clausole da raffinare e gli stessi raffinamenti.

Relazione tra idealità e ricoprimento downward/upward

In [van der Laag,95] viene fornita una relazione fra la proprietà di idealità e i cosiddetti *ricoprimenti downward/upward* di una clausola

Definizione (*ricoprimento downward/upward*)

In un insieme quasi-ordinato $(S, \leq)$, $D \in S$ è un ricoprimento downward (risp. upward) di una clausola C se e solo se
$D < C$ e not $\exists E \in S: D < E < C$

(risp. $C < D$ e not $\exists E \in S: C < E < D$)

Relazione tra idealità e ricoprimento downward/upward

Definizione (*insieme di ricoprimento downward/upward*)

In un insieme quasi-ordinato (S, ≤), data C ∈ S, un insieme di ricoprimento downward (risp. upward) di C è l'insieme massimale di tutti i ricoprimenti downward (risp. upward) non-equivalenti di C.

dc(C): insieme di ricoprimento downward di una clausola C

uc(C): insieme di ricoprimento upward di una clausola C

Relazione tra idealità e ricoprimento downward/upward

Definizione (*completezza di un insieme di ricoprimento*)

In un insieme quasi-ordinato $(S, \leq)$, data $C \in S$, un insieme di ricoprimento downward dc(C) (risp. upward uc(C)) di una clausola C è completo se e solo se

$\forall D \in S, C > D \; \exists E \in d(C): E \geq D$

(risp. $\forall D \in S, C < D \; \exists E \in u(C): E \leq D$)

Insiemi quasi-ordinati dotati di insiemi di ricoprimento non completi sono comuni in matematica. Per esempio, se consideriamo l'insieme dei numeri razionali, ordinati in base all'ordinamento usuale sui numeri reali, non esiste alcune coppia di numeri X, Y tali che $X > Y$ e Y sia un ricoprimento downward di X.

Relazione tra idealità e ricoprimento downward/upward

Lemma:
Dato un operatore di raffinamento (upward/downward) ideale definito su un insieme quasi-ordinato (S, ≤), allora ogni elemento dell'insieme S ha insiemi di ricoprimenti (downward/upward) finiti e completi ($d(C) \subseteq \rho(C)$ / risp. $u(C) \subseteq \delta(C)$)

Relazione tra idealità e ricoprimento downward/upward

Corollario

In un insieme quasi-ordinato(S, ≤), se qualche C ∈ S non ammette insiemi di ricoprimento (downward/upward) completi e finiti, allora non esiste un operatore di raffinamento (downward/upward) ideale per (S, ≤)

Relazione tra insiemi di ricoprimento e catene infinite prive di ricoprimento

Definizione (*catene infinite prive di ricoprimento*)

Dato un insieme quasi-ordinato $(S, \leq)$
date $C, D_1, D_2, D_3, \ldots \in S$:

– se $D_1 > D_2 > D_3 > \ldots > D_n > D_{n+1} > \ldots > C$ e non esiste nessun $E \in uc(C)$ t.c. $D_n \geq E > C \; \forall \; n \geq 1$, allora $D_1, D_2, \ldots, D_3$ è una catena infinita strettamente discendente priva di ricoprimento di C

– se $C > \ldots > D_{n+1} > D_n > \ldots > D_3 > D_2 > D_1$ e non esiste nessun $E \in dc(C)$ t.c. $C \geq E > D_n \; \forall \; n \geq 1$, allora $D_1, D_2, \ldots, D_3$ è una catena infinita strettamente ascendente priva di ricoprimento di C

Relazione tra insiemi di ricoprimento e catene infinite prive di ricoprimento

Lemma

Dato un insieme quasi-ordinato $(S, \leq)$ e $C \in S$ e le seguenti proposizioni:

a. dc(C) (risp. uc(C)) è incompleto
b. C ha una catena infinita ascendente (risp. discendente) priva di ricoprimento
c. dc(C) (risp. uc(C)) è infinito o incompleto

risulta: **a** implica **b** e **b** implica **c**

Relazione tra insiemi di ricoprimento e catene infinite prive di ricoprimento

Corollario

In un insieme quasi-ordinato(S, $\leq$), se qualche C $\in$ S ammette catene infinite ascendenti (risp. discendenti) prive di ricoprimento, allora non esiste un operatore di raffinamento (downward/upward) ideale per (S, $\leq$)

Non esistenza di operatori di raffinamento ideali in θ-sussunzione

Teorema

Dato uno spazio di ricerca non ristretto $(S, \leq_\theta)$, contenente almeno un predicato o una funzione con arità > 1, allora non esiste un operatore di raffinamento ideale per tale spazio $(S, \leq_\theta)$

La dimostrazione viene dall'esistenza di catene infinite strettamente ascendenti di una clausola C o catene prive di ricoprimento in uno spazio di ricerca non ristretto.

Tuttavia, anche imponendo qualche bias sul linguaggio di rappresentazione (come la linkedness), non è possibile definire operatori di raffinamento ideali: *le catene continuano ad esistere perché intrinseche nella struttura a reticolo dello spazio, il bias eviterebbe solo di incappare in esse*

Esempio di catene infinite prive di ricoprimento

$C = q :- p(X_1,X_2), p(X_2,X_1)$

$E_n = q :- p(X_1,X_2), p(X_2,X_1),$
$p(Y_1,Y_2), p(Y_2,Y_3), \ldots,$
$p(Y_{n-1},Y_n), p(Y_n,Y_1) \qquad n \geq 3$

E_3^n è un esempio di catena infinita priva di ricoprimento strettamente ascendente di C poiché $E_3^n <_\theta E_3^{n+1} \ \forall n \geq 1$

$C' = q\ (X) :- p(X_1, X_2, X) , p(X_2,X_1, X)$

$E'_n = q\ (X) :- p(X_1,X_2, X), p(X_2,X_1, X),$
$p(Y_1,Y_2, X), p(Y_2,Y_3, X), \ldots,$
$p(Y_{n-1},Y_n, X), p(Y_n,Y_1, X) \quad n \geq 3$

Sono, ancora, catene linkate prive di ricoprimento

Esistenza di operatori di raffinamento ideali in θ_{OI}-sussunzione

Proposizione

In uno spazio di ricerca (Datalog^{OI}, $\leq_{\theta_{OI}}$), data una clausola C, non esistono catene infinite prive di ricoprimento a) ascendenti e b) discendenti per C.

Da cui la possibilità di definire operatori di raffinamento ideali in tale spazio

Operatore di raffinamento downward ρ_{OI}

Sia C una clausola Datalog^{OI}. Allora una clausola $D \in \rho_{OI}(C)$ se:

i. $D = C\theta$, dove $\theta = \{X/a\}$, $X \in$ vars(C) e $a \notin$ consts(C), cioè θ è una sostituzione che rimpiazza una variabile di C con una nuova costante

ii. $D = C \cup \{\neg A\}$, dove $\neg A$ è un letterale tale che: $\neg A \notin C$.

Nota:
$C = \{H, \neg B_1, \neg B_2, \ldots, \neg B_n\}$ ovvero $(H :- B_1, B_2, \ldots, B_n)$ dunque aggiungere/togliere $\neg A$ a C equivale a dire aggiungere/togliere un letterale al corpo della clausola.

Operatore di raffinamento upward δ_{OI}

Sia C una clausola Datalog^{OI}. Allora una clausola $D \in \delta_{OI}(C)$ se:

i. $D = C\sigma$, dove $\sigma = \{a/X\}$, $a \in$ consts(C) e $X \notin$ vars(C), cioè σ è una anti-sostituzione che rimpiazza una costante di C con una nuova variabile

ii. $D = C \setminus \{\neg A\}$, dove $\neg A$ è un letterale tale che: $\neg A \in C$.

Operatore di raffinamento downward ρ'_{OI}

Sia C una clausola Datalog^{OI}. Allora una clausola $D \in \rho'_{OI}(C)$ se:

i. $D = C\theta$, dove $\theta = \{X/a\}$, $X \in$ vars(C) e $a \notin$ consts(C), cioè θ è una sostituzione che rimpiazza una variabile di C con una nuova costante

ii. $D = C \cup \{\neg A\}$, dove $\neg A$ è un letterale tale che: $\neg A \notin C$ e consts(A) $\subseteq$ consts(C).

Operatore di raffinamento upward δ'_{OI}

Sia C una clausola Datalog^{OI}. Allora una clausola $D \in \delta'_{OI}(C)$ se:

i. $D = C\sigma$, dove $\sigma = \{a/X\}$, $a \in$ consts(C) e $X \notin$ vars(C), cioè σ è una anti-sostituzione che rimpiazza una costante di C con una nuova variabile

ii. $D = C \setminus \{\neg A\}$, dove $\neg A$ è un letterale t.c.: $\neg A \in C$ e consts(A) $\subseteq$ consts($C \setminus \{\neg A\}$).

Idealità degli operatori di raffinamento in θ_{OI}

Teorema

Dato uno spazio di ricerca (Datalog^{OI}, $\leq_{\theta_{OI}}$), gli operatori ρ_{OI} , δ_{OI} , ρ'_{OI} , δ'_{OI} sono operatori di raffinamento ideali

Una ulteriore proprietà degli operatori definiti è quella di poter calcolare il numero massimo di passi raffinamento da effettuare in base ad una misura statica (*size*) delle clausole.

La *size* di una clausola può essere definita come una funzione che associa ad ogni clausola una qualche forma di misura di complessità strutturale di essa:

size(C) = simboli - variabili distinte

C = p(X) :- q(X,Y)

size(C) = 5-2 = 3

Proposizione

Se C θ_{OI}-sussume propriamente una clausola D, allora risulta che size(D) > size(C)

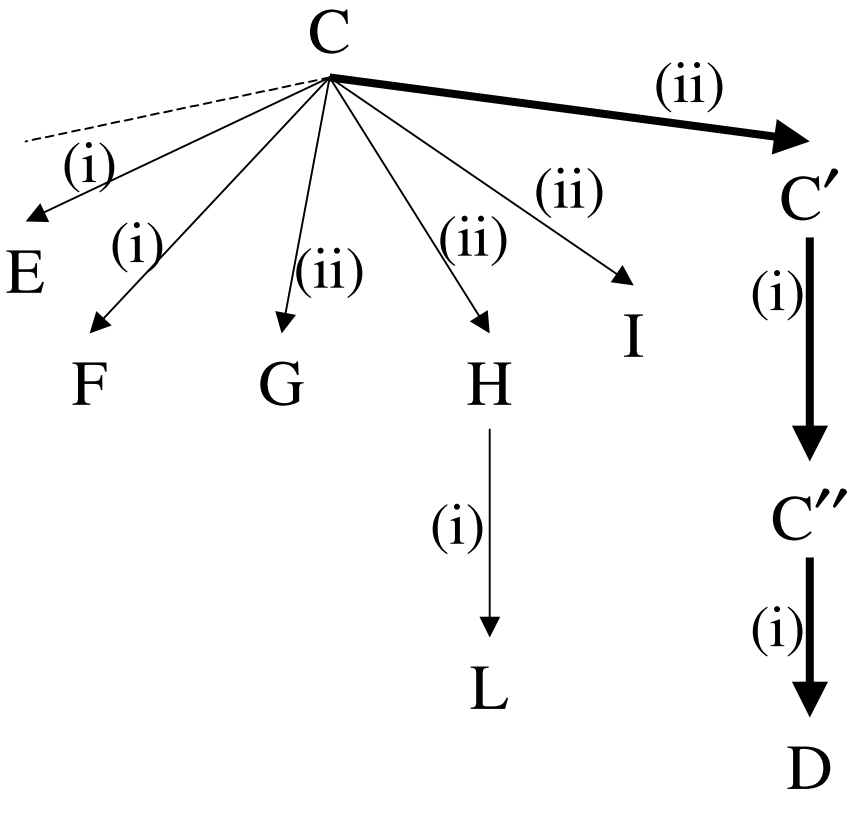
ed inoltre

D è nel set di *k*-passi di raffinamento downward di C

C è nel set di *k*-passi di raffinamento upward di D

$D \in \rho^k_{OI}(C)$, $C \in \delta^k_{OI}(D)$ con $k \leq size(D) - size(C)$

Ciò garantisce che la catena di raffinamento da C a D abbia una lunghezza non superiore a *k*.



C = p(X) :- q(X,Y)

C′ = p(X) :- q(X,Y), q(U,V)

C″ = p(X) :- q(X,Y), q(a,V)

D = p(X) :- q(X,Y), q(a,b)

E = p(X) :- q(X,b)

F = p(a) :- q(a,Y)

G = p(X) :- q(X,Y), q(Y,X)

H = p(X) :- q(X,Y), q(Y,Z)

I = p(X) :- q(X,Y), q(X,Z)

L = p(X) :- q(X,Y), q(Y,a)

size(C) = 3

size(D) = 6

$k = 3 \Rightarrow D \in \rho^3_{OI}(C)$

Ma D può essere ottenuto in un solo passo da C applicando il punto (ii) della definizione di ρ_{OI} con A = q(a,b)

Mentre la *size* ci consente di ottenere un criterio per definire un limite superiore al numero di passi di raffinamenti, una nuova misura, $size_{OI}$, ci permette di ottenere il numero esatto di passi di raffinamento nel caso in cui gli operatori impiegati siano ρ'_{OI} , δ'_{OI} .

$size_{OI}(C) = |C| + |consts(C)|$

$C = p(X) :- q(X,Y)$

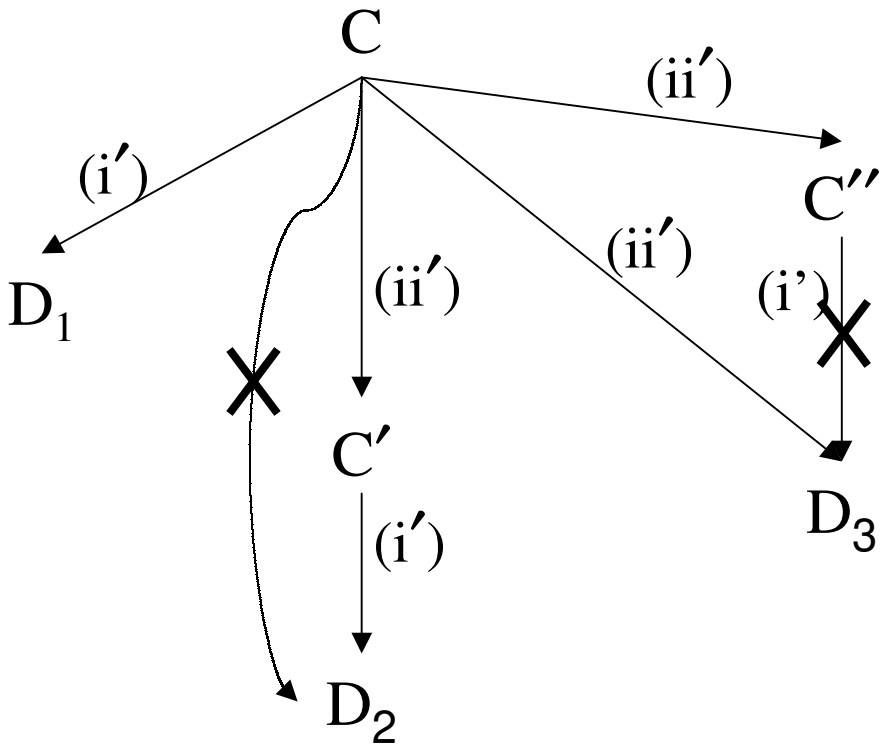
$size_{OI}(C) = 2+0 = 2$

Se C θ_{OI}-sussume propriamente una clausola D

D è raggiungibile in *k*-passi di raffinamento downward da C

C è raggiungibile in *k*-passi di raffinamento upward da D

$D \in \rho^k_{OI}(C)$, $C \in \delta^k_{OI}(D)$ con $k = size_{OI}(D) - size_{OI}(C)$



C = p(X) :- q(X,a)

D_1 = p(X) :- q(c,a)

D_2 = p(X) :- q(X,a), r(b)

D_3 = p(X) :- q(X,Y), r(a)

C′ = p(X) :- q(X,a),r(Y)

C″ = p(X) :- q(X,a),r(Y)

$size_{OI}(C) = 3$

$size_{OI}(D_1) = 4$

$size_{OI}(D_2) = 5$

$size_{OI}(D_3) = 4$

$k = 1 \Rightarrow D_1 \in \rho^1_{OI}(C)$

$k = 2 \Rightarrow D_2 \in \rho^2_{OI}(C)$

$k = 1 \Rightarrow D_3 \in \rho^1_{OI}(C)$

Conclusione

- Necessità di operatori di raffinamento
- Proprietà degli operatori di raffinamento
- Importanza di tali proprietà
- Non-esistenza di operatori di raffinamento ideali in θ -sussunzione
- Esistenza di operatori di raffinamento ideali in θ_{OI} -sussunzione
- Misura statica di passi di raffinamento necessari