∈∉Φπσδη↔←→¬∨∧⊆≠∪

Lezione 8

DCL: Programmi in Tipi di Dati Astratti.
Esempi

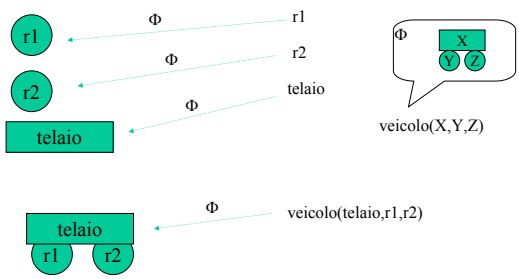
Dove siamo

- Abbiamo visto l'uso di DCL con costanti come:
  - linguaggio per basi dati
  - RRS
- Abbiamo introdotto le funzioni, le interpretazioni di Herbrandt, esteso le procedure top-down e bottom-up.
- **Vediamo ora l' assiomatizzazione di tipi di dati in DCL.**

1. Funzioni come costruttori

- Ricordiamo che Φ interpreta:
  f n-aria con $f^{\Phi} : D^n \rightarrow D$
- Nelle interpretazioni di Herbrandt D = U(KB) e:
  f n-aria si interpreta in $f^{\Phi_H} : U(KB)^n \rightarrow U(KB)$ definita da
  $f^{\Phi_H}(t_1,..,t_n) = f(t_1,..,t_n)$
- Ciò corrisponde a considerare f come costruttore di dati.
  Dal punto di vista del significato esterno:

Esempio



2. DOMINIO CHIUSO E UNIVERSO DI HERBRANDT

- L'esempio rappresenta il punto di vista da usare in DCL:
  - costanti: ***dati semplici***, non strutturati
  - termini ground (non costanti): ***dati strutturati***
- Funzioni = operazioni di composizione che generano dati strutturati a partire da dati semplici.
- Adatte a definire tipi dati come liste, alberi, ecc.
- Un ***tipo di dati*** si ottiene fissando i suoi ***costruttori***
  - cioè le costanti che corrispondono ai dati più semplici e agli operatori che costruiscono i dati strutturati.

Esempio: i numeri naturali.

- I numeri naturali sono generati dai costruttori
  0/0 , s/1
- L'universo di Herbrandt è 0, s(0), s(s(0)), …
- L'interpretazione "esterna" è:
  0 rappresenta 0, s(0) rappresenta 1, ecc.
- Si noti che s^{Φ_H} è proprio il successore; ad esempio:
  $s^{\Phi_H}(s(0)) = s(s(0))$

2.1. La segnatura di una base di conoscenza KB; universo e base di Herbrandt

- La segnatura di una KB è
  - l'insieme dei simboli di funzione e costante usati in KB;
    - genera ***l'universo di Herbrandt*** U_{KB}
  - l'insieme dei simboli di predicato usati in KB;
    - genera la ***base di Herbrandt*** B_{KB}
- Possiamo fare diverse ipotesi circa l'universo:

- ***Ipotesi del dominio chiuso:***
  - tutti gli individui sono rappresentati da termini di U_{KB}
- ***Ipotesi del dominio aperto:***
  - non conosciamo a priori l'insieme di tutti gli individui; alcuni non hanno denotazione in U_{KB}
- ***Ipotesi del nome unico:***
  - termini ground distinti rappresentano individui diversi.

Esercizio

- Si consideri il programma Vocali:

```
vocale(a).
vocale(e).
vocale(i).
vocale(o).
vocale(u).
consonante(d).
```

- Qual è la segnatura di Vocali?
- Qual è l'universo di Herbrandt?
- Pensando all'alfabeto, ha senso l'ipotesi del nome unico?
- E quella del dominio chiuso?

Esercizio critico

- Si vogliano rappresentare i record di naturali.
- Qual è la segnatura?
- Definire il programma per il predicato leq(A,B) che confronta due naturali secondo l'usuale ordinamento e due record componente per componente.
- L'universo di Herbrandt contiene ospiti indesiderati?
- Come risolvere il problema? I TIPI come predicati di tipo.

2.2. Una soluzione: type checking positivo

```
/*  tipo dei naturali */
        is(0,nat).
        is(s(X),nat) :- is(X,nat).

/*  tipo dei record */
        is(rec(X,Y),rec) :- is(X,nat), is(Y,nat).
```



```
/* ordinamento totale dei naturali con type checking positivo*/
    leqnat(0,X) :- is(X,nat).
    leqnat(s(X), s(Y)) :- leqnat(X,Y).

/* ordinamento parziale dei record con type checking positivo*/
    leq(rec(A,B),rec(C,D)) :- leqnat(A,C), leqnat(B,D).

/*  Risposta SI: i tipi sono OK  e vale leq:
    Risposta NO: non vale leq o i tipi sono errati */
```

3. Assunzione del mondo chiuso (CWA, Closed World Assumption) e definizioni induttive

- A livello di tipi di dati, conviene collegare CWA e definizioni induttive.
- Un predicato p(X1,..,Xn) è definito induttivamente da:
  - Base: p(X1,…,Xn) = vero se ***Base***(X1,..,Xn)
  - Passo: se p(X1,…,Xn) = vero e ***Passo***(X1,..,Xn,Y1,..,Yn), allora p(Y1,…,Yn) = vero
  - Minimalità: se p(X1,..,Xn) non è derivabile da Base e Passo, allora è falso
- L'ipotesi di minimalità corrisponde al mondo chiuso.

In DCL

p(X1,..,Xn) :- ***Base****(X1,..,Xn).*

p(Y1,..,Yn) :- p(X1,…,Xn), ***Passo****(X1,…,Xn,Y1,…,Yn).*

- *Si tratta di programmare* ***Base****(X1,..,Xn) e* ***Passo****(X1,…,Xn,Y1,…,Yn).*
- *L'ipotesi di minimalità corrisponde ad assumere come falso ciò che non vale nel modello di Herbrandt minimo*

Esempio: definizione di un tipo

Tipo nat

Base: X:nat se X = 0

Passo: se X:nat e Y=s(X), allora Y:nat

Minimalità: nient'altro è di tipo nat

Siccome ***Base****(X)* è X=0 e ***Passo****(X,Y)* è Y=s(X), conviene

Base: 0:nat

Passo: se X:nat, allora s(X):nat

In DCL:

is(0,nat).
is(s(X),nat) :- is(X,nat).

Esempio: definizione di un predicato

Predicato X <= Y

Base: 0 <= Y

Passo: se X <= Y, allora s(X) <= s(Y)

Minimalità: in nessun altro caso, X <= Y

In DCL:

leq(0,X).
leq(s(X),s(Y)) :- leq(X,Y).

Esempio: relazioni funzionali

Definiamo half(X,Y) =def x div 2.

Base: half(X,0) se X=0 or X=s(0).

Passo: se half(X,Y), allora half(s(s(X)),s(Y))

Minimalità: al solito

In DCL

half(0,0).
half(s(0),0).
half(s(s(X)),s(Y)) :- half(X,Y).

3. Correttezza positiva e copertura

- Riguardano il legame fra semantica esterna
  - ciò che vogliamo modellare con una KB o un programma
- e semantica interna
  - risposte alle query nel modello minimo di Herbrandt.
- Correttezza positiva (o parziale): le sostituzioni di risposta (semantica interna) sono vere nella semantica esterna
- Copertura: ogni risposta nella semantica esterna è anche una risposta in quella interna
  - nel caso di terminazione per gli input desiderati, la completezza garantisce la correttezza delle risposte negative

Come verificare la correttezza positiva

- Tutte le clausole devono essere vere nell'interpretazione esterna.

Esempio

- Programma per leq(X,Y)
  - interpretazione esterna: leq(X,Y) significa $X \leq Y$ sui naturali
- Verità di leq(0,X):

  $0 \leq X$ vera per ogni naturale X
- Verità di leq(s(X),s(Y)) :- leq(X,Y):

  $(X+1) \leq (Y+1) \leftarrow X \leq Y$ vera per ogni X,Y:naturali

Come verificare la copertura.

- La verifica della copertura è più difficile. Vediamo come operare conoscendo l'interpretazione esterna (ciò che vorremmo) e analizzando il programma
- L'analisi è relativa a un insieme di ***query di interesse***

  Esempio: interessati a leq(X,ground);
  - una query ?leq(X, s(s(0)))
  - ci aspettiamo le risposte X = 0, X = s(0), X = s(s(0))

Condizione A). Per ogni query ?A di interesse che abbia soluzione nell'interpretazione esterna, considerare i possibili alberi ottenuti con un passo all'indietro:

ogni soluzione *$A\sigma$ (soluzione intesa, cioè nell'interpretazione esterna) deve essere ottenuta da almeno un'istanza di uno degli alberi, che abbia tutte le premesse vere nell'interpretazione esterna*

Condizione B. *Per ogni query ?A di interesse deve valere la terminazione.* Per verificarla:

le eventuali query ricorsive devono essere più ***vicine a casi di terminazione*** di quanto non lo sia la radice; ciò si può verificare associando agli atomi in un albero di prova un ***grado*** intero non negativo e ***strettamente decrescente*** man mano che percorre l'albero dalla radice.

NB. La terminazione è condizione sufficiente

Esempio

A) Query ?leq(X,ground). Abbiamo i casi

1 $\dfrac{}{\text{leq}(0,0)}$

2 $\dfrac{}{\text{leq}(0,s(g))}$

$\dfrac{\text{?leq}(X1,t)}{\text{leq}(s(X1),s(t))}$ 3

?leq(X,0)

?leq(X,g).

?leq(X,s(t))

leq(X,0) ha unica soluzione X=0: è ottenuta dall'albero 1.
Le soluzioni di leq(X,s(g)) possono essere
X=0 ottenuta dall'albero 2
della forma X=s(g'), con g' <= g ottenute dall'albero 3 (la premessa leq(g',g) è vera nell'interpretazione esterna)

B) Terminazione: considerando tutte le query, l'unico albero con premessa ricorsiva è:

$$\frac{?leq(X1,g)}{leq(s(X1),s(g))}$$

Siccome g è minore di s(g), la ricorsione non può essere infinita

- Possiamo anche vedere che leq è completo anche per le query di decisione ?leq(In1,In2). leq(g1,g2) rientra in uno dei casi:

$$\frac{}{leq(0,0)} \qquad \frac{}{leq(0,s(g))} \qquad \frac{fail}{leq(s(g),0)} \qquad \frac{leq(g,g')}{leq(s(g),s(g'))}$$

Le query con risposta positiva si ottengono con i primi due alberi e l'ultimo, e si procede come prima.
La terminazione è garantita per tutti o goals (l'unico nuovo albero, il terzo, non è ricorsivo)

Esercizio

- Dimostrare che il programma half è corretto e completo rispetto ai goal half(ground,X).
- Dimostrare che il programma sum è corretto rispetto ai goal sum(ground,ground',X) e calcola X=ground"

4. Propagazione dei modi In, Out e analisi di groundness

- Programmi con più predicati debbono conservare i modi di input/output previsti.
- È utile un'analisi di groundness.

Modi di input output e analisi di groundness

Consideriamo i seguenti ***modi di ingresso/uscita***, che sono i più frequentemente usati:

- In: ingresso ground
- Out: uscita con risposta ground
- Esempio: half(In, Out) prevede che ogni chiamata: half(ground,X) dia una risposta X= ground' o fallisca finitamente
- Con le modalità considerate è utile ***un'analisi di groundness.***

Esempio

- La somma ha i seguenti modi:
  sum(In,In,Out!)
  sum(Out,Out,In)
  con Out! Indichiamo che esiste una e una sola risposta.

DIM. Per induzione sulla prof. albero, lecita una volta appurata le terminazione

Base: Out ground

$\overline{\text{sum(g,0,g)}}$

sum(In,In,Out)

Passo: Ipotesi ind. V ground
segue s(V) ground

$$\frac{\text{sum(g,g',V)}}{\text{sum(g,s(g'),s(V))}}$$

Base: Out1, Out2 ground

$\overline{\text{sum(0,0,0)}}$

sum(Out1,Out2,In)

Passo: Ipotesi ind. A,B ground
segue A, s(B) ground

$$\frac{\text{sum(A,B,g)}}{\text{sum(A,s(B),s(g))}}$$

ESEMPIO

prod(X,0,0).
prod(X,s(Y),Z) :- prod(X,Y,V), sum(V,X,Z).

prod(In,In,Out) calcola Out ground

prod(g,0,0) → ground

ground → ground

$$\frac{\text{prod(g,g',V)} \quad \text{sum(V,g',Z)}}{\text{prod(g,s(g'),Z)}}$$

Esercizio

- Determinare tutti i cammini lunghi n in un grafo. Il grafo sia rappresentato dagli archi e la lunghezza dai naturali 0, s(0), ….;
- dividere la soluzione in due parti, una generale (definizione dei cammini con lunghezza in un grafo) e una contingente (il grafo specifico);
- provare la parte generale con diversi grafi specifici.
- NB. Pensare sempre in termini correttezza positiva e completezza, individuando i modi In, Out.