Lezione 25
Programmazione Deduttiva e Dichiarativa

April 23, 2012

Logic Programming Fundamentals

- Metodologie: Programmazione Deduttiva
- Meccanismi Addizionali: Is, if-then-else.
- Meccanismi Addizionali: cut
- Estensioni: Negation as Failure
- Data Base and Constraint Programming

Metodologie: Programmazione Deduttiva

Vogliamo usare linguaggi come PROLOG per scrivere programmi in grado di dare soluzione a problemi come *la torre di Hanoi* o come questi sotto.

Example

Si dispongano `N` regine su una scacchiera $\texttt{N} \times \texttt{N}$ senza che due regine si coprano reciprocamente. Si scriva un programma che mostra se e quante configurazioni diverse diverse possono essere fornite per dato `N`.

Example

Si consideri un piano suddiviso in un arbitrario naturale `n`, $\texttt{n} < \texttt{N} \in \texttt{Nat}$ di regioni chiuse. Si abbiano `k` diversi colori a disposizione con cui colorare regioni adiacenti in modo tale che regione adiacenti abbiano colori distinti. Si scriva un programma che mostra se e quante colorazioni diverse del piano possono essere ottenute per dato `(n,k)`.

Metodologie: Programmazione Deduttiva/1

- i componenti base di un programma:
  - simboli di funzione come costruttori di dato (non interpretati: iniezioni)
  - simboli di predicato come relazioni tra dati
  - variabili logiche (quantificate universalmente nelle clausole, esistenzialmente nel goal)
- programmazione induttiva: il fattoriale
- programmazione deduttiva:
  - La soluzione procede con trasformazioni conservative della sola soddisfacibilità
  - $\texttt{G1} \rightarrow_{\texttt{lush}} \texttt{G2}$ se $\texttt{G1} \Leftarrow \texttt{G2}$
- Nuova Metodologia di Programmazione: Applichiamola alla Torre di Hanoi

Metodologie: Programmazione Deduttiva/2

- Nuova Metodologia di Programmazione: Applichiamola alla Torre di Hanoi
- **Il Problema**: *Fornire la sequenza di mosse da attuare per trasferire n dischi ordinati da A a B, potendo utilizzare una posizione C, in modo tale che l'ordinamento dei dischi non sia mai violato*.
- Come scrivere un programma per tale problema?

Example

Se avvessi le mosse per spostare i primi n dischi, cosa dovrei fare per spostare n+1-esimo, quello maggiore dei precedenti?

Metodologie: Programmazione Deduttiva/3

- Come scrivere un programma per tale problema?

Example

Se avvessi le mosse per spostare i primi n dischi, cosa dovrei fare per spostare n+1-esimo, quello maggiore dei precedenti?

```
hanoi(s(0),A,B,C,M) :-append(Z,[move(A,B)],M).
hanoi(s(N),A,B,C,M) :-hanoi(N,A,C,B,M1),
                      hanoi(N,C,B,A,M2),
                      append(M1,[move(A,B)|M2],M).
```

Metodologie: Programmazione Dichiarativa/1

- programmazione deduttiva:
  - La soluzione procede con trasformazioni conservative della sola soddisfacibilità
  - $\texttt{G1} \rightarrow_{\texttt{lush}} \texttt{G2}$ se $\texttt{G1} \Leftarrow \texttt{G2}$
- Una Differente Lettura: Nel problema precedente abbiamo utilizzato un algoritmo (e composto soluzioni a sotto problemi, in forma di casi più semplici)
  - Possiamo essere ancora più ASTRATTI
  - In certi casi possiamo limitarci a descrivere solo ciò che ci aspettiamo di calcolare
  - Dichiaratività. Applichiamola al problema dei quattro colori (planarità grafi con outdegree n)

Metodologie: Programmazione Dichiarativa/2

- Dichiaratività. In certi casi possiamo limitarci a descrivere le proprietà del valore da calcolare
- Applichiamola al problema dei quattro colori (planarit grafi con outdegree n)
- **Il Problema**: *Dato un grafo (piano) con* `n` *vertici (regioni) di outdegree (lati) massimo* $\texttt{m} \leq \texttt{k}$, *e date* `k` *distinte etichette (colori), si chiede se sia possibile associare ad ogni vertice un etichetta in modo tale che vertici adiacenti, connessi da un arco, abbiano etichette diverse?*

Example

Se Sapessi caratterizzare le proprietà di ogni grafo soluzione, come potrei trovare la soluzione?

Metodologie: Programmazione Dichiarativa/3

- Se sapessi caratterizzare le proprietà di ogni grafo soluzione, come potrei trovare la soluzione?
- La caratterizzazione forma il programma in grado di calcolare la soluzione

Example

```
coloured(P,N,K) :-allRegions(P,N),noConflict(P,N,K).
allRegions([put(_,_)],s(0)).   /*almeno 1 vertice*/
allRegions([put(_,reg(I))|[put(_,reg(s(I)))|P]],s(N)) :-
          allRegions([put(_,reg(s(I)))|P],N).
```

completare opportunamente

Possiamo essere meno dichiarativi/2

Example

Si mostri come deve essere definito un grafo per utilizzare il predicato coloured/3 dell'esercizio precedente e lo si usi in un modulo per PROLOG che risolve il problema dei colori per tale grafo.

Example

Si discutano altre definizioni per il predicato allRegions/2 e le si usino, in alternativa alla data, in un modulo per PROLOG

Meccanismi Addizionali: is, if-then-else

- Controllare la dichiaratività: vincolare gli algoritmi utilizzati
  - Tecniche diverse: tra queste meccanismi per controllo della risoluzione
  - simboli di predicato come relazioni tra dati
- **is**: match di un termine con il valore calcolato da un'espressione

  `Term is Expression`

Example

```
len([],0).
len([X|Xs],M):-len(Xs,N), M is N+1.
```

- **if-then-else**: discrimina tra due atomi

  `B → C;D`

Possiamo essere meno dichiarativi

Example

un programma in L.P.

$\mathtt{X, Y, Z, W}: \mathtt{int} \quad \mathtt{M, N}: \mathtt{nat} \quad \mathtt{s}: \mathtt{int} \rightarrow \mathtt{int}$

$\mathtt{plus}: \mathtt{int} \times \mathtt{int} \times \mathtt{int} \quad \mathtt{times}: \mathtt{int} \times \mathtt{int} \times \mathtt{int}$

$\mathtt{fact}: \mathtt{int} \times \mathtt{int}$

PROLOG *non ha tipi*.

```
(1) fact(0,1).
(2) fact(s(X),M) :-fact(X,N),cnv(Z,N),times(s(X),Z,Y),cnv(Y,M).
(3) times(0,Y,0).
(4) times(s(X),Y,W) :-times(X,Y,Z),plus(Y,Z,W).
(5) plus(0,Y,Y).
(6) plus(s(X),Y,s(W)) :-plus(X,Y,W).
(7) cnv(0,0).
(8) cnv(s(X),M) :-cnv(X,N),M is N + 1.
```

Possiamo essere meno dichiarativi/2

Example

Siano $\texttt{Fact}_{\texttt{int}}$ e $\texttt{Fact}_{\texttt{nat}}$ il primo e il secondo, rispettivamente, programma Prolog dato per il fattoriale, nei lucidi precedenti. Si dica cosa calcola $\texttt{Fact}(\texttt{X}, \texttt{s}(\texttt{s}(0)))$, rispettivamente $\texttt{Fact}(\texttt{X}, 2)$, giustificando il comportamento ottenuto.

Example

Si dia una definizione del predicato allRegions(P,N) introdotto nel problema del grafo planare (o dei k colori). Le regioni riportante in P sono disposte ancora in ordine crescente ma utilizzano gli interi nell'intervallo 1..N.

Meccanismi Addizionali: Cut

- Controllare la dichiaratività: vincolare gli algoritmi utilizzati
- Tagliare il calcolo nondeterministico di una stessa soluzione (altri algoritmi)
- Tagliare il calcolo di altre soluzioni quando ne basta una
- **cut**: !

  $\mathtt{H} :\!- \mathtt{A}_1, ..., \mathtt{A}_k, !, \mathtt{B}_1, ..., \mathtt{B}_h$

  sfronda lo spazio di ricerca tagliando i sottoalberi prodotti da (i risolventi sospesi per) $A_1, ..., A_k$ e quelli di H (impedendo la selezione delle successive clausole (risolventi sospesi) per H)

Example

```
fact(s(X),M):-fact(X,N),cnv(Z,N),times(s(X),Z,Y),cnv(Y,M).
fact(s(X),M):-fact(X,N),cnv(Z,N),!,times(s(X),Z,Y),cnv(Y,M),!.
```

Estensioni: Negation as Failure

- Questa è una vera estensione
- Nel calcolo clausale (a sequenti) possiamo asserire solo in forma positiva
- E se volessimo combinare tali asserzioni con altre negative
  ad esempio: un dato termine non è in una data lista
- **not**: \+
  \+ A
  fallimento finito: se la risoluzione di A termina con fallimento allora \+ A termina con successo.

Example

```
inG([X|Xs],X):-!.
inG([_|Xs],X):-inG(Xs,X).
```

Un goal: \+ inG([a, b, c], b).

Estensioni e Varianti

- Datalog: per basi di dati deduttive
- CLP: Logic Programming with constraints
- HOL: Higher Order Logic Programming
- Semantic Unification
- Integrazione con altri paradigmi: Narrowing, Funzionale, Imperativo
- ...