Lezione 11

RICORSIONE IN PROLOG

Le liste

1. Programmi su Liste

- Un primo approccio nella determinazione di un programma con le liste è la ricorsione strutturale
  - si applica in generale con i tipi di dati definiti induttivamente
- BASE: clausole della forma
  p(nil,…). oppure
  p(nil,…) :- ….
- PASSO: clausole della forma:
  p(cons(H,L),…) :- …., p(L,…), ….

Append

Base:

app(**nil**, X, X)

Passo

app(**cons**(**H**,**L**), X, cons(H,R)) :- app(**L**,X,R).

app(nil, X, X).
app(cons(H,L), X, cons(H,R)) :- app(L,X,R).

app(cons(H_1,L_1), X_1, cons(H_1,R_1)) :- app(L_1,X_1,R_1).
{H_1/a, L_1/cons(b,nil), X_1/cons(c,nil), U/cons(a,R_1)}

app(cons(b,nil), cons(c,nil), R_1)

app(cons(a,cons(b,nil)), cons(c,nil), U)?

app(nil, X, X).
app(cons(H,L), X, cons(H,R)) :- app(L,X,R).

app(cons(H_2,L_2), X_2, cons(H_2,R_2)) :- app(L_2,X_2,R_2).
{H_2/b, L_2/nil, X_2/cons(c,nil), R_1/cons(b,R_2)}

app(nil, cons(c,nil), R_2)?

app(cons(b,nil), cons(c,nil), R_1)?

app(cons(a,cons(b,nil)), cons(c,nil), cons(a,R_1))

app(nil, X, X).
app(cons(H,L), X, cons(H,R)) :- app(L,X,R).

app(nil, X_3, X_3).
{X_3/cons(c,nil),R_2/cons(c,nil)}

app(nil, cons(c,nil), R_2)?

app(cons(b,nil), app(c,nil), cons(b,R_2))

app(cons(a,cons(b,nil)), cons(c,nil), cons(a, cons(b,R_2)))

```
app(nil, X, X).
app(cons(H,L), X, cons(H,R)) :- app(L,X,R).
```

app(nil, cons(c,nil), cons(c,nil))

app(cons(b,nil), cons(c,nil), cons(b, cons(c,nil)))

app(cons(a,cons(b,nil)), cons(c,nil), cons(a, cons(b, cons(c,nil))))

2. Come si ragiona per esempi con il passo

- Dopo aver verificato che i casi base operano correttamente
  - di solito è semplice
- individuare un caso rappresentativo da risolvere con il passo e disegnare i passi all'indietro corrispondenti alle clausole passo;
- *assumere* (induzione) che i sottoproblemi ricorsivamente generati ottengano soluzione
- e verificare che le soluzioni ottenute portino ad una soluzione corretta del problema;
- se ogni caso rappresentativo è contemplato da almeno una clausola base o passo e il programma termina, si ha correttezza
  - andrebbe precisato

Esempio:

app(**nil**, X, X)
app(**cons**(**H**,**L**), X, cons(H,R)) :- app(**L**,X,R).

app(cons(2,nil),cons(3,nil),R1)?

app(cons(1,cons(2,nil)), cons(3,nil),cons(1,R1))

Supponiamo che si risolva correttamente:
R1 = cons(2,cons(3,nil)

Il risultato
cons(1,cons(2, cons(3, nil)))
è quello atteso?

SI

Tutte le liste sono contemplate da base e passo?
SI, perché una lista è nil o è cons(H,L).

app(**nil**, X, X)
app(**cons**(**H**,**L**), X, cons(H,R)) :- app(**L**,X,R).

Il caso **app(cons(1,cons(2,nil)), cons(3,nil), cons(1,R1))**
è rappresentativo di tutti i casi cons(H,L)?

SI, perché nessuna ipotesi su cons(**1, cons(2,nil)**), **cons(3,nil)**
è stata usata.

3. Le liste in Prolog

- Data l'importanza delle liste, Prolog usa una notazione più compatta e, generalmente, un insieme di predicati base sulle liste è predefinito; in particolare, concatenazione, unione, appartenenza.

Notazione per le liste

[a|[b|[c|[]]]] cons(a,cons(b,cons(c,nil)))

si usa la notazione semplificata [a,b,c] ; le seguenti scritture sono equivalenti:
[a|[b|[c|[]]]] = [a|[b|[c]]] = [a|[b,c]] = [a,b,c] ; così:

[a|X] cons(a,X)
[a,b|X] cons(a, cons(b, X))

Con la notazione abbreviata:

```
app ([], X, X).
app ([H|C], X, [H|R]) :- app (C,X,R).
```

Esercizi

- Dare tutte le rappresentazioni parentesizzate equivalenti a:

  [a,b,c,d] = [a|[b,c,d]] = ….

  Provare poi a testare le eguaglianze individuate mediante l'interprete Prolog
- Riscrivere la procedura app usando la notazione abbreviata
- Applicare la procedura top-down per derivare la soluzione di
  app([a,b],[c,d],R)

Reverse

- Naive Reverse (ingenua):
  rev([],[]).
  rev([X|C],R) :- rev(C,R1), app(R1,[X],R).

Esercizio: discutere la correttezza.

Reverse

- Versione con accumulatore
  rev(X,R) :- rev3(X, [], R).
  rev3([], A, A).
  rev3([T|C], A, R) :- rev3(C, [T|A], R)

- COMPLESSITA': quella ingenua è quadratica, questa è lineare.
- CORRETTEZZA: rev3(L, A, R) ha il significato inteso:
  - se appendiamo A alla reverse di L otteniamo R, cioè:
  - app(reverse di L, A, R)
- … alla lavagna …