

Lezione 10

- A) Le Liste
- B) Esempi

1. L'ADT delle liste

- Base: $\text{nil} \in \text{list}(T)$
- Passo: se $L \in \text{list}(T)$ e $H \in T$, allora $\text{cons}(H,L) \in \text{list}(T)$

$\text{isa}(\text{nil}, \text{list}(T)).$

$\text{isa}(\text{cons}(H,L), \text{list}(T)) \leftarrow \text{isa}(H,T) \wedge \text{isa}(L, \text{list}(T))$

2.1. Programmi su Liste

- Un primo approccio: ricorsione primitiva sulla def. Induttiva delle liste
- BASE: clausole della forma
 $\text{p}(\text{nil}, \dots).$ oppure
 $\text{p}(\text{nil}, \dots) \text{ :- } \dots$
- PASSO: clausole della forma:
 $\text{p}(\text{cons}(H,L), \dots) \text{ :- } \dots, \text{p}(L, \dots), \dots$

Append

Significato di $\text{app}(X,Y,Z)$: Z = concatenazione di X,Y

Modo: $\text{app}(\text{In}, \text{In}, \text{Out}), \text{app}(\text{Out}, \text{Out}, \text{In})$

Base:

$\text{app}(\text{nil}, X, X)$

Passo

$\text{app}(\text{cons}(H,L), X, \text{cons}(H,R)) \text{ :- } \text{app}(L,X,R).$

$\text{app}(\text{nil}, X, X).$
 $\text{app}(\text{cons}(H,L), X, \text{cons}(H,R)) \text{ :- } \text{app}(L,X,R).$

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

$\text{app}(\text{cons}(b, \text{nil}), \text{cons}(c, \text{nil}), R_1)$

$\text{app}(\text{cons}(a, \text{cons}(b, \text{nil})), \text{cons}(c, \text{nil}), U)?$

$\text{app}(\text{nil}, X, X).$
 $\text{app}(\text{cons}(H,L), X, \text{cons}(H,R)) \text{ :- } \text{app}(L,X,R).$

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

$\text{app}(\text{nil}, \text{cons}(c, \text{nil}), R_2)?$

$\text{app}(\text{cons}(b, \text{nil}), \text{cons}(c, \text{nil}), R_1)?$

$\text{app}(\text{cons}(a, \text{cons}(b, \text{nil})), \text{cons}(c, \text{nil}), \text{cons}(a, R_1))$

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

app(nil, X₃, X₃).
 {X₃/cons(c,nil), R₂/cons(c,nil)}

app(nil, cons(c,nil), R₂)?
 app(cons(b,nil), app(c,nil), cons(b,R₂))
 app(cons(a,cons(b,nil)), cons(c,nil), cons(a, cons(b,R₂)))

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))))

Correttezza positiva e copertura di append

- Terminazione: liste destrutturate in sottoliste per ricorsione primitiva
- Correttezza positiva: le clausole sono vere (dim. per esercizio)
- Copertura: Query app(In,In,Out) abbiamo copertura:

app(nil,g,g)
 app(t,g', R1)
 app(cons(h, t), g', cons(h,R1))

2.2. 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|X] cons(a,X)

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

[a|[b|[c|[]]]] si semplificata in [a,b,c] ;
 le seguenti scritture sono equivalenti:
 [a|[b|[c|[]]]] = [a|[b|[c]]] = [a|[b,c]] = [a,b,c]

Append 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:
 $[a,b,c,d] = [a[[b,c,d]]] = \dots$
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 $\text{app}([a,b],[c,d],R)$

Reverse

- Naive Reverse (ingenua):
 $\text{rev}([], []).$
 $\text{rev}([X|C], R) :- \text{rev}(C, R1), \text{app}(R1, [X], R).$

Esercizio: discutere correttezza positiva, copertura e terminazione.

Esercizio: discutere complessità in funzione della lunghezza delle liste

Altri esempi

- Alla lavagna o con il PC