# Lezione 11

A) Le Liste
B) Esempi

## 1. L'ADT delle liste

- Base: nil ∈ list(T)
- Passo: se L ∈ list(T) e H ∈ T, allora cons(H,L) ∈ list(T)

isa(nil, list(T)).
isa(cons(H,L), list(T)) ← isa(H,T) ∧ isa(L, list(T))

## 2.1. Programmi su Liste

- Un primo approccio: ricorsione primitiva sulla def. Induttiva delle liste
- BASE: clausole della forma
  p(nil,…). oppure
  p(nil,…) :- ….
- PASSO: clausole della forma:
  p(cons(H,L),…) :- …., p(L,…), ….

## Append

Significato di app(X,Y,Z): Z = concatenazione di X,Y
Modo: app(In,In,Out), app(Out,Out,In)

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))))

## 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(g,g',X)

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]] = ….
  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 correttezza positiva, copertura e terminazione.

Esercizio: discutere complessità in funzione della lunghezza delle liste

## Altri esempi

- Alla lavagna o con il PC