University of Auckland

# 07.363

Functional and Logic Programming

Christian Collberg

October 7, 1994

## Homework 1



### Homework – Lists 1

- Write a procedure `islist` which succeeds if its argument is a list, and fails otherwise.

Answer

```
islist([]).
islist([_|T]) :- islist(T).
```

**Slide 11–1**

### Homework – Lists 2

- Write a procedure `alter` which changes English sentences according to rules given in the database. Example:

```
change(you, i).
change(are, [am, not]).
change(french, german).
change(do, no).
?- alter([you,are,a,computer],X).
   X = [i,[am,not],a,computer]
```

Answer

```
change(X, X).
alter([],[]).
alter([H|T],[X|Y]) :-
   change(H, X), alter(T, Y).
```

**Slide 11–2**

### Homework – Lists 3

- Write a list subtraction procedure. Example:

```
?- sub([1,2,4,6,8], [2,6], L).
   L=[1,4,8].
```

Answer

```
sub(L,[],L) :- !.
sub([H|T],L,U) :-
   member(H, L), !, sub(T, L, U).
sub([H|T],L,[H|U]) :-
   !, sub(T, L, U).
sub(_,_, []).
```

**Slide 11–3**

## Homework – Lists 4

- Write a procedure `pick` which returns the first `N` elements of a given list. Example:

```
?- pick([1,2,4,6,8], 3, L).
   L=[1,2,4].
```

Answer

```
pick([],_,[]) :- !.
pick(L,N,_) :-
   length(L, N2), N2 < N, !,
   write('Nonsense'), nl.
pick([X|L],N,[X|P]) :- N>0,
   N1 is N-1, pick(L, N1, P).
pick(_,0,[]).
```

**Slide 11–4**

## Homework – Lists 5

- Write a procedure `alt` which produces every other element in a list. Example:

```
?- alt([1,2,3,4,5,6], A).
   A = [1,3,5]
```

Answer

```
alt([],[]).
alt([X],[X]).
alt([A,B|T],[A|U]) :- alt(T,U).
```

**Slide 11–5**

## Homework – Lists 6

- Write a procedure `del` which removes duplicate elements from a list. Example:

```
?- del([a,c,x,a,g,c,d,a],A).
   A = [a,c,x,g,d]
```

Answer

```
del([],[]).
del([H|T],L) :- member(H,T),
        del(T,L), !.
del([H|T],[H|L]) :- del(T,L).
```

**Slide 11–6**

## Homework – Strings 1

- Write a procedure which converts an atom to lower case. Example:

```
?- tolower('hEj_HoPp3', A).
   A = hej_hopp3
```

Answer

```
tolower(X,Y) :- name(X,L),
   low_list(L, L1), name(Y,L1).
low_list([],[]).
low_list([H|T],[H|T1]) :- % a-z
   H >= 96, H =< 122, !,
   low_list(T,T1).
low_list([H|T],[H1|T1]) :- % A-Z
   H >= 65, H =< 90, !,
   H1 is H - 32, low_list(T,T1).
low_list([H|T],[H|T1]) :- % Other
   !, low_list(T,T1).
```

**Slide 11–7**

## Homework – Arithmetic 1

- Write a procedure `max3` which produces the largest of three integers. Example:

```
?- max3(3,5,1,X).
   X = 5
```

Answer

```
max3(X,Y,Z,M) :-
   max(X,Y,T), max(Z,T,M).
max(A,B,B) :- A < B.
max(A,B,A) :- A >= B.
```

**Slide 11–8**

## Homework – Arithmetic 2

- Write a procedure `double` which multiplies each element in a list of numbers by 2. Example:

```
?- double([1,5,3,9,2], A).
   A = [2,10,6,18,4]
```

Answer

```
double([],[]).
double([H|T],[K|U]) :-
   K is 2*H, double(T,U).
```

**Slide 11–9**

## Homework – Arithmetic 3

- Write a procedure `ave` which computes the average of a list of numbers. Example:

```
?- ave([1,5,3,9,2], A).
   A = 4
```

Answer

```
sum([],0).
sum([H|T], P) :-
   sum(T,Z), P is Z+H.

ave(L, X) :-
   sum(L, S), length(L,N),
   X is S/N.
```

**Slide 11–10**

## Homework – Arithmetic 4

- Write a procedure `sum` which produces the sum of the integers up to and including its first argument. Example:

```
?- sum(5, S).
   S = 15
```

Answer

```
sum(1,1).
sum(N, R) :-
   N1 is N-1,
   sum(N1, R1),
   R is R1+N.
```

**Slide 11–11**

## Homework – Database 1

- Suppose our database contains facts of the form

  `person_age(Name, Age).`
  `person_sex(Name, Sex).`

  Write a procedure `combine` which adds facts of the form

  `person_full(Name, Age, Sex).`

Answer

```
combine :-
   person_age(N, A),
   person_sex(N, S),
   assert(person_full(N, S, A)),
   fail.
combine.
```

**Slide 11–12**

## Homework – Database 2

- Write a Prolog procedure which reverses the order of John's children in the database. Example:

```
child(mary, john).
child(jane, john).
child(bill, john).
```

  is changed to

```
child(bill, john).
child(jane, john).
child(mary, john).
```

Answer

```
reversefacts :-
   retract(child(X, john)),
   asserta(child(X,john)), fail.
reversefacts.
```

**Slide 11–13**

## Homework – Database 3

- Write a procedure to assemble a list of someone's children from the facts in the database. Example:

```
child(mary, john).
child(jane, john).
child(bill, john).
?- assemble(john, L).
   L = [mary, jane, bill]
```

Answer

```
assemble(P, L):-makelist(P,[],L).
makelist(P, S, L) :-
   retract(child(X, P)), !,
   makelist(P, [X|S], L),
   asserta(child(X,P)).
makelist(P, L, L).
```

**Slide 11–14**