University of Auckland, Department of Computer Science

07.363 — Homework 1

Christian Collberg
September 28, 1994

1 Introduction

Work these problems to prepare for assignments and exams. They don't need to be handed in and will not be marked. If you have problems with any of the exercises discuss them with a friend or bring it up in class.

2 Lists

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

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

3. Write a list subtraction procedure. Example:

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

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].
```

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]
```

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]
```

3 Strings

1. Write a procedure `tolower` which converts an atom containing upper case characters to the corresponding atom with only lower case characters. Example:

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

4 Arithmetic

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

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

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]
```

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

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

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
```

5 Manipulating the Database

1. Suppose our database contains facts of the form

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

   where `Sex` is either `male` or `female`. Write a procedure `combine` which extends the database with additional facts of the form

   ```
   person_full(Name, Age, Sex).
   ```

   The procedure should produce one such fact for each person who has both an age record and a sex record. Example: Given the following database

   ```
   person_age(chris, 25).  % Yeah, right...
   person_sex(chris, male).
   person_age(louise, 8).
   person_sex(louise, female).
   ```

   `combine` should produce these additional facts:

   ```
   person_full(chris, 25, male).
   person_full(louise, 8, female).
   ```

2. Write a Prolog procedure which reverses the order of Johns children in the database. For example, given the following database

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

   the goal `?- reversefacts.` should change it to

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

3. Write a Prolog procedure to assemble a list of someone's children from the facts in the database. The database should remain unchanged. Example:

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

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