

CS 101 Computer Science I (Spring 2001)

Lab 10: April 25-27, 2001

The Matchmaker is a program that helps lonely hearts to find their soul mates. Given a set of people as input, this program returns a relation (set of ordered pairs) that includes all the potentially compatible matches of one person in the set with another person in the set.

In the lab for this week, you will run experiments to compare the efficiency of two different methods of finding acceptable matches. You will also write some of the helper procedures you will need in the Matchmaker program. In the assignment for this week, you will write the Matchmaker program itself.

We have prepared file called "`cs101-dating.scm`" that describes a community of people. You will need to use some of the definitions in this file in your solutions to the lab exercises. You will also need to use some of the definitions in the file called "`cs101-sets.scm`". Both of these files are available on the class web.

1. Write a Scheme procedure called "`status`". This procedure takes a symbol called "`p`" representing a person, as its argument. It returns either the symbol `single` or the symbol `married`. This procedure operates by finding the symbol `p` in the `*status*` relation. After you have written the `status` procedure, you should use it to write a Scheme predicate called "`single?`". This predicate takes a symbol called "`p`" representing a person, as its argument. It returns either `#t` or `#f`, depending on the status of the person.
2. Write a Scheme procedure called "`gender`". This procedure takes a person `p` as its argument. It returns either the symbol `male` or the symbol `female`. This procedure operates by finding the symbol `p` in the `*gender*` relation. After you have written the `gender` procedure, you should use it to write two Scheme predicate procedures, called "`male?`" and "`female?`". Each of these predicates takes a person `p` as its argument, and returns either `#t` or `#f`, depending on the gender of the person.
3. Write a Scheme procedure called "`interests`". This procedure takes a person `p`, as its argument. It returns the set of the things in which person `p` is interested. Each interest is represented by a symbol. This procedure operates by finding the symbol `p` in the `*interest*` relation. After you have written the `interests` procedure, you should use it to write a Scheme predicate procedure called "`common-interest?`". This procedure takes two people `p1` and `p2` as its arguments. It returns `#t` if `p1` and `p2` have a common interest. Otherwise, it returns `#f`.
4. Experiment with two different definitions of a Scheme procedure called "`candidate-couples`". This procedure takes a set `s` of people as its argument. It returns a set of ordered pairs of people.

Version 1: First form the cartesian product of the given set with itself. Then form a smaller set, keeping only those pairs in which the first component is a single male, and the second component is a single female.

```
(define candidate-couples-1
  (lambda (set)
    (set-comprehension (cartesian-product set set)
                      (lambda (op) (and (male? (op-1st op))
                                         (single? (op-1st op))
                                         (female? (op-2nd op))
                                         (single? (op-2nd op)))))))
```

Version 2: First find the set of all single men. Then find the set of all single women. Then form the cartesian-product of these two sets.

```
(define candidate-couples-2
  (lambda (set)
    (cartesian-product
     (set-comprehension set
                       (lambda (p) (and (male? p)(single? p))))
     (set-comprehension set
                       (lambda (p) (and (female? p)(single? p)))))))
```

Type each definition into the Scheme interpreter. Evaluate each definition with ***people*** as the set parameter. Notice how long each version takes to run.