CSU290 HWK4, Due Wed Mar 12 2008 at 10 AM

Homework 4:

- You *have to* work in pairs for this homework and you have to work with someone you have not worked with yet. Send the name of your partner to Zhifeng (austin@ccs.neu.edu) by the end of the day, on Friday Mar 7th. If you do not do this, you will get a 0 on your homework.

- Each pair should submit their homework via blackboard. Each pair should submit just once. That is, one person per team should submit the homework via their blackboard account. Your submission should be a text file. The format of the file should be:
  - the names of the team
  - clear solutions to the problems

PROBLEM 1:

Let’s revisit the finite-set functions we saw in Homework 2.

We take a set to a be a true list of elements. Thus,

```
  (defun setp (l)
    (if (endp l)
        (equal l nil)
      (setp (cdr l))))
```

Set membership:

```
  (defun set-memberp (a l)
    (if (endp l)
        nil
      (or (equal (car l) a)
          (set-memberp a (cdr l)))))
```

Set inclusion:

```
  (defun set-subsetp (l m)
    (if (endp l)
        t
      (and (set-memberp (car l) m)
           (set-subsetp (cdr l) m))))
```

Set equality:

```
  (defun set-equalp (l m)
    (and (set-subsetp l m)
         (set-subsetp m l)))
```

Union and intersection are straightforward:

```
  (defun set-union (l m)
    (if (endp l)
        m
      (cons (car l)
            (set-union (cdr l) m))))

  (defun set-inter (l m)
    (if (endp l)
        nil
      (if (set-memberp (car l) m)
```

```
          (cons (car l) (set-inter (cdr l) m))
        (set-inter (cdr l) m))))
```

Set size - recall that we do not count duplicate elements when we are considering the size of a set, that is, {a,b,b,c,d} has size 4.

```
  ;; remove all occurrences of a from l
  (defun remove-element (a l)
    (if (endp l)
        nil
      (if (equal (car l) a)
          (remove-element a (cdr l))
        (cons (car l) (remove-element a (cdr l))))))

  (defun set-size (l)
    (if (endp l)
        0
      (+ 1 (set-size (remove-element (car l) (cdr l))))))
```

We are asking you to formalize and then prove the following conjectures.

This means that you should go through the following steps for each of the questions below:

1. Write down a formula that expresses the conjecture we are asking you to prove.

2. Then, try to prove the formula you came up with. Remember, first try to come up with counterexample to get a sense of whether what you are trying to prove is actually true.

3. If you are proving the formula by induction, make sure that you clearly identify the proof obligations.

Some of these are hard, so make sure that you at least get the formula you are trying to prove right, and get the proof obligations right.

For (a), recall that recall that a relation R is reflexive if x R x for all x, a relation R is symmetric if x R y implies y R x for all x, y, and a relation R is transitive if x R y and y R z implies x R z for all x,y,z.

For (b)-(e), recall that a binary operation OP is commutative if (x OP y) = (y OP x) for all x,y, and associative if (x OP (y OP z)) = ((x OP y) OP z) for all x,y,z.

(a) Prove that set-equalp is an equivalence relation, that is, that it is reflexive, symmetric, and transitive.

(b) Prove that set-union is commutative.

(c) Prove that set-inter is commutative.

(d) Prove that set-union is associative.

(e) Prove that set-inter is associative.

PROBLEM 2:

Using the definitions in the previous problem, again formalize and prove the following conjectures.

Again, some of these are hard, so make sure that you at least get the formula you are trying to prove right, and get the proof obligations right.

(a) Prove that the size (as sets) of A union B is greater than or equal to the size of A

(b) Prove that the size (as sets) of A union B is greater than or equal to the size of B.

(c) Prove that the size (as sets) of A intersection B is less than or equal to the size of A.

(d) Prove that the size (as sets) of A intersection B is less than or equal to the size of B.

(e) Prove that if A is a subset of B, then the size of A is less than or equal to the size of B.

(f) Prove that if A and B are equal sets, then A and B have the same size.

PROBLEM 3:

Sets have the property that duplicate elements are "irrelevant". In particular, two sets are equal if they have the same elements, irrespectively of how many times those elements occur.

Our definition of union in Problem 1 can lead to duplicate elements being added to the resulting set. For instance, (set-union ’(1 2 3) ’(1 2 3)) produces (1 2 3 1 2 3), which as a set is equal to the simpler (1 2 3).

Here is a version of union that produces results guaranteed not to have any duplicated elements as long as the input sets themselves do not have any duplicated elements:

```
  (defun set-union-nd (l m)
    (if (endp l)
        m
      (cons (car l)
            (set-union-nd (cdr l)
                          (remove-element (car l) m)))))
```

Note that this function cannot be obtained from the design recipe we have seen until now. It is an instance of generative recursion.

(a) Write down a formula that says that set-union and set-union-nd return equal sets when given the same inputs.

(b) Write down the proof obligations for the formula you wrote down in (a).

* (c) Prove that set-union and set-union-nd return equals sets when given the same inputs.

This question is much harder than anything you’ve had to prove until now. But give it a stab.

Note that to prove that (and X Y) = t, it suffices to prove that X = t and that Y = t (under the same hypotheses).

If you hit a point in the proof where you need a particular result but you don’t know how to prove it (but believe it’s true, and have not been able to find counter-examples to it!), feel free to posit it as a conjecture, and continue the proof. Then try to see if you can prove the conjectures you have accumulated. Please make sure that you clearly write down all the conjectures you assume in your proof.

For instance, when proving the base cases of the induction, it turns out it is handy to conjecture that

```
    (set-subsetp (cdr b) b) = t
```

And of course this is actually provable.

*** PROBLEM 4: (This is optional. It will not be graded.)

Formalize and prove at least 2 interesting theorems using the above functions. Write a paragraph describing why the theorems are interesting.

*** PROBLEM 5: (This is optional. Send directly to the TA running your lab)

Stump your TA: Come up with one conjecture (it can be true or false), that can be expressed using 300 characters or less and challenge your "TA" to either exhibit a counterexample or a proof. If you come up with a really challenging conjecture, then you will get 20 extra points on your homework grade. Send this directly to the TA in charge of your lab.