CS2800 Exam 3 Review

Below are some extra problems in the form of the last homework.

1. Substitution

a. `(subst 'a a (foo b (rev (app y x))))`
   `(subst 'a (* x y) (foo 'b (rev (app (list y) (cdr x)))))`

b. `(+ x y (+ x y) (* z w))`
   `(+ (/ w z) x (+ (/ w z) x) (* (- (sum-n (1- z)) (expt 2 w)) w))`

Proofs

The remaining problems ask for equational proofs about ACL2 programs. When writing your equational reasoning be sure to justify each step in the style shown in class, e.g.

```
  (sum (app x y))
= { def. app }
  (sum (if (endp x)
           y
           (cons (car x) (app (cdr x) y))))
= { (not (endp x)) assumption, if-false axiom }
  (sum (cons (car x) (app (cdr x) y)))
```

Use basic arithmetic equalities as axioms and justify them as `arithmetic`, e.g.

```
  (+ (car x) (+ (sum (cdr x)) (sum y)))
= { arithmetic: + is associative }
  (+ (+ (car x) (sum (cdr x))) (sum y))

  (+ 0 (len y))
= { arithmetic: 0+a = a }
  (len y)
```

2. App nil identity

Prove the following

```
(and (implies (endp x)
              (= (listfix x) (app x nil)))
     (implies (and (consp x)
                   (= (listfix (cdr x)) (app (cdr x) nil)))
              (= (listfix x) (app x nil))))
```

using the definitions

```
(defun listfix (x)
  (if (endp x)
    nil
    (cons (car x) (listfix (cdr x)))))

(defun app (x y)
  (if (endp x)
    y
    (cons (car x) (app (cdr x) y))))
```

3. App rev rev is rev app

Assuming the following are theorems,

```
(implies (and (true-listp x)
              (true-listp y)
              (true-listp z))
         (= (app (app x y) z) (app x (app y z))))

(implies (true-listp y)
         (= (listfix y) y))

(= (listfix x) (app x nil))
```

prove or give a counter-example of the following conjecture. If the following is a non-theorem, what is a way to change the statement such that it is a theorem and represents the behavior we intend?

```
(and (implies (and (endp x)
                   (true-listp y))
              (= (rev (app x y)) (app (rev y) (rev x))))
     (implies (and (consp x)
                   (true-listp x)
                   (true-listp y)
                   (= (rev (app (cdr x)) y) (app (rev y) (rev (cdr x)))))
              (= (rev (app x y)) (app (rev y) (rev x)))))
```

using the definitions

```
(defun true-listp (x)
  (if (endp x)
    (= x nil)
    (true-listp (cdr x))))

(defun rev (x)
  (if (endp x)
    nil
    (app (rev (cdr x)) (list (car x)))))
```

4. Arithmetic identity

Prove or give a counter-example of the following conjecture. If the following is a non-theorem, what is a way to change the statement such that it is a theorem and represents the behavior we intend?

```
(and (implies (zp x)
              (= (sum-n x) (/ (* x (+ x 1)) 2)))
     (implies (and (not (zp x))
                   (= (sum-n (- x 1)) (/ (* (- x 1) x) 2)))
              (= (sum-n x) (/ (* x (+ x 1)) 2))))
```

using the definition

```
(defun sum-n (n)
  (if (zp n)
    0
    (+ n (sum-n (- n 1)))))
```