CS313K: Logic, Sets, and Functions

J Strother Moore
Department of Computer Sciences
University of Texas at Austin

Lecture 18 – Chap 5 (5.1, 5.2, 5.3)

Midterm 2

Midterm 2 is this coming Thursday, April 1, 3:30 – 5:00.

It will be in WEL 2.224.

Instructions

Write your name and EID above and circle the unique ID of your discussion section! Write your answers in the space provided. There are 10 questions, worth various amounts, totaling 200 points. You have until 5:00 pm.

On "yes/no" questions you need not provide proofs or counterexamples.

When defining functions you need not write the "declarations" that ACL2 sometimes forces you to write. For example, you can write `(defun foo (x) 23)` rather than `(defun foo (x) (declare (ignore x)) 23)`. In proofs, just write "Basic" to justify propositional reasoning with the basic axioms (pg 113–114); that is, you need not name the tautologies you use or cite `if-ax1`, etc. However, note all definitions and lemmas you use.

Instructions

Write your name and EID above and circle the unique ID of your discussion section! Write your answers in the space provided. There are 10 questions, worth various amounts, totaling 200 points. You have until 5:00 pm.

On "yes/no" questions you need not provide proofs or counterexamples.

When defining functions you need not write the "declarations" that ACL2 sometimes forces you to write. For example, you can write `(defun foo (x) 23)` rather than `(defun foo (x) (declare (ignore x)) 23)`. In proofs, just write "Basic" to justify propositional reasoning with the basic axioms (pg 113–114); that is, you need not name the tautologies you use or cite `if-ax1`, etc. However, note all definitions and lemmas you use.

Instructions

Write your name and EID above and circle the unique ID of your discussion section! Write your answers in the space provided. There are 10 questions, worth various amounts, totaling 200 points. You have until 5:00 pm.

On "yes/no" questions you need not provide proofs or counterexamples. When defining functions you need not write the "declarations" that ACL2 sometimes forces you to write. For example, you can write `(defun foo (x) 23)` rather than `(defun foo (x) (declare (ignore x)) 23)`. In proofs, just write "Basic" to justify propositional reasoning with the basic axioms (pg 113–114); that is, you need not name the tautologies you use or cite `if-ax1`, etc. However, note all definitions and lemmas you use.

Instructions

Write your name and EID above and circle the unique ID of your discussion section! Write your answers in the space provided. There are 10 questions, worth various amounts, totaling 200 points. You have until 5:00 pm.

On "yes/no" questions you need not provide proofs or counterexamples.

When defining functions you need not write the "declarations" that ACL2 sometimes forces you to write. For example, you can write `(defun foo (x) 23)` rather than `(defun foo (x) (declare (ignore x)) 23)`. In proofs, just write "Basic" to justify propositional reasoning with the basic axioms (pg 113–114); that is, you need not name the tautologies you use or cite `if-ax1`, etc. However, note all definitions and lemmas you use.

You may refer to the course notes (the blue book) during the exam. You may refer to your own notes if they are on paper. No computers are allowed. No talking is allowed. No cellphones. Remove hats, baseball caps, etc.

Square brackets are sometimes used in place of parentheses to make it easier for you to see groupings. For example, I might write:

$(p \wedge [q \rightarrow r]) \rightarrow s$

instead of

$(p \wedge (q \rightarrow r)) \rightarrow s.$

The last section of the exam shows the familiar definitions for `endp`, `app`, `true-listp`, `mem` and `rev`, for your convenience.

Opinion Poll (no points)

How should I handle questions during the exam?

A. Students ask questions from their seats and I answer to the whole room.

B. Students talk to me privately and if I hear the same question twice I talk to the whole room.

Result: Proposition A won, 60% to 40%.

The best way to study for the exam is to do the problems in Section 4.10 **Practice Proofs**, pg 142–149, and Section 5.3 **Practice Proofs**, pg 157–166.

Induction *will be covered on the exam*.

I will post last year's Midterm 2 on the class home page:

`http://www.cs.utexas.edu/users/moore/classes/cs313k/`

Remember: a free tutor is listed on the home page.

In the following, use the definitions at the end of Chapter 5.

Here are some lemmas that can all be proved by expanding the definitions of `insert` and `ordp`, and using properties of `lexorder` (namely, that it is reflexive, transitive, and *total*:
`(lexorder x y)` $\vee$ `(lexorder y x)`.

```
ordp-insert-singleton:
  ((¬(endp x))
   ∧
   (endp (rest x)))
→
  (ordp (insert e x))
```

```
ordp-non-singleton:
  ((¬(endp x))
   ∧
   (¬(endp (rest x))))
→
  ((ordp x)
  ↔
   ((lexorder (first x)(first (rest x)))
    ∧
    (ordp (rest x))))
```

```
ordp-insert-non-singleton:
  ((¬(endp x))
   ∧
   (¬(endp (rest x)))
   ∧
   (lexorder (first x) (first (rest x))))
→
  ((ordp (insert e x))
   ↔
   (if (lexorder e (first x))
       (ordp (rest x))
       (if (lexorder e (first (rest x)))
           (ordp (rest x))
           (ordp (insert e (rest x))))))
```

`ordp-insert:`

`(ordp x)` $\rightarrow$ `(ordp (insert e x))`

Induct on `x`.

Base Case

`(endp x)` $\rightarrow$ `((ordp x)` $\rightarrow$ `(ordp (insert e x)))`

$\leftrightarrow$ {Promotion}

`((endp x)` $\wedge$ `(ordp x))` $\rightarrow$ `(ordp (insert e x))`

$\leftrightarrow$ {`insert`}

`((endp x)` $\wedge$ `(ordp x))` $\rightarrow$ `(ordp (cons e x))`

$$(\texttt{(endp x)} \wedge \texttt{(ordp x)}) \rightarrow \texttt{(ordp (cons e x))}$$

$$\leftrightarrow \hfill \{\text{ordp}\}$$

$$(\texttt{(endp x)} \wedge \texttt{(ordp x)}) \rightarrow True$$

$$\leftrightarrow \hfill \{\text{Short-Circuit}\}$$

$$True$$

Ind Step

```
((¬(endp x))
 ∧
 ((ordp (rest x))
  →
  (ordp (insert e (rest x)))))
→
((ordp x) → (ordp (insert e x)))
```

I will do cases on whether `(endp (rest x))` is true or false.

Case 1

```
  ((endp (rest x))
   ∧
   (¬(endp x))
   ∧
   ((ordp (rest x))
    →
    (ordp (insert e (rest x)))))
→
  ((ordp x) → (ordp (insert e x)))
↔                          {ordp-insert-singleton}
```

`ordp-insert-singleton:`

$$\texttt{((}\neg\texttt{(endp x))} \wedge \texttt{(endp (rest x)))} \longrightarrow \texttt{(ordp (insert e x))}$$

Case 1

```
  ((endp (rest x))
   ∧
   (¬(endp x))
   ∧
   ((ordp (rest x))
    →
    (ordp (insert e (rest x)))))
→
  ((ordp x) → (ordp (insert e x)))
↔                    {ordp-insert-singleton}
```

$$
\begin{array}{l}
\quad \texttt{((endp (rest x))} \\
\quad\ \wedge \\
\quad\ \texttt{(}\neg\texttt{(endp x))} \\
\quad\ \wedge \\
\quad\ \texttt{((ordp (rest x))} \\
\quad\quad \longrightarrow \\
\quad\quad \texttt{(ordp (insert e (rest x)))))} \\
\longrightarrow \\
\quad \texttt{((ordp x)} \longrightarrow True) \\
\leftrightarrow\ \ True \qquad\qquad\qquad \{\text{propositional simplification}\}
\end{array}
$$

Case 2

```
  ((¬(endp (rest x)))
   ∧
   (¬(endp x))
   ∧
   ((ordp (rest x))
    →
    (ordp (insert e (rest x)))))
→
  ((ordp x) → (ordp (insert e x)))
↔             {Promo, ordp-non-singleton, FC}
```

```
ordp-non-singleton:
  ((¬(endp x))
   ∧
   (¬(endp (rest x))))
→
  ((ordp x)
  ↔
   ((lexorder (first x)(first (rest x)))
    ∧
    (ordp (rest x))))
  ((¬(endp (rest x)))
   ∧
   (¬(endp x))
   ∧
   (ordp (insert e (rest x)))
   ∧
   (lexorder (first x) (first (rest x)))
   ∧
   (ordp (rest x)))
→
  (ordp (insert e x))
↔                            {ordp-insert-non-singleton}
```

```
ordp-insert-non-singleton:
  ((¬(endp x))
   ∧
   (¬(endp (rest x)))
   ∧
   (lexorder (first x) (first (rest x))))
→
  ((ordp (insert e x))
   ↔
   (if (lexorder e (first x))
       (ordp (rest x))
       (if (lexorder e (first (rest x)))
           (ordp (rest x))
           (ordp (insert e (rest x))))))
```

```
    ((¬(endp (rest x)))
     ∧
     (¬(endp x))
     ∧
     (ordp (insert e (rest x)))
     ∧
     (lexorder (first x) (first (rest x)))
     ∧
     (ordp (rest x)))
→
    (if (lexorder e (first x))
        (ordp (rest x))
        (if (lexorder e (first (rest x)))
            (ordp (rest x))
            (ordp (insert e (rest x)))))
↔                                          {Hyp (3 times)}
```

```
((¬(endp (rest x)))
 ∧
 (¬(endp x))
 ∧
 (ordp (insert e (rest x)))
 ∧
 (lexorder (first x) (first (rest x)))
 ∧
 (ordp (rest x)))
→
 (if (lexorder e (first x))
     True
     (if (lexorder e (first (rest x)))
         True
         True))
```

$\leftrightarrow$ $\{$propositional simplification$\}$

$True$

$\square$

So we've proved:

`(ordp x)` $\rightarrow$ `(ordp (insert e x))`.

We'll need this later.