

CS313K: Logic, Sets, and Functions

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

(Lecture 9)

Announcements

Behnam Robatmili has changed his office hours to Monday 2:00–3:30 and Thursday 1:30–3:00.

The midterms will be held in WEL 2.224, *not in this room!* I updated the web page.

David posted the instructions last night but the actual instructions are a little bit different now.

Midterm Instructions

I will show you the instructions page of the exam,
and the last page of definitions.

More Advice About the Midterm

I will not question you on wff . The only functions you'll be asked to define are recursions on lists.

Recursion Made Easy: $\alpha\beta\gamma$

Suppose you want to answer some question about a list x .

Recursion Made Easy $\alpha\beta\gamma$

Suppose you want to answer some question about a list x .

α : What is the answer for the empty list?

Recursion Made Easy $\alpha\beta\gamma$

Suppose you want to answer some question about a list x .

α : What is the answer for the empty list?

Don't think about computing! Just think about the *question*. "What is the right answer for the empty list?"

Recursion Made Easy $\alpha\beta\gamma$

Suppose you want to answer some question about a list x .

α : What is the answer for the empty list?

Recursion Made Easy $\alpha\beta\gamma$

Suppose you want to answer some question about a list x .

α : What is the answer for the empty list?

β : How is the answer for x related to the answer for $(\text{cdr } x)$ when x is non-empty?

Recursion Made Easy $\alpha\beta\gamma$

Suppose you want to answer some question about a list x .

α : What is the answer for the empty list?

β : How is the answer for x related to the answer for $(\text{cdr } x)$ when x is non-empty?

Again, don't think about computing! Just think about how the right answer to the question for x is related to or derived from the right answer to the question for $(\text{cdr } x)$. Remember, think about the *question* not computing.

Recursion Made Easy $\alpha\beta\gamma$ (continued)

γ : Then put α and β into this template and give your function a name, f .

```
(defun  $f$  (...  $x$ )  
  (if (endp  $x$ )  
       $\alpha$   
      ( $\beta$  ... (car  $x$ ) ... ( $f$  ... (cdr  $x$ ))))))
```

Example 1

Define `len` so that `(len x)` is the number of elements in the list `x`.

Example 1

Define `len` so that `(len x)` is the number of elements in the list `x`.

α: What is the number of elements in the empty list? **0**

Example 1

Define `len` so that `(len x)` is the number of elements in the list `x`.

α : What is the number of elements in the empty list? **0**

β : How is the number of elements in `x` related to the number of elements in `(cdr x)`? **The number in `x` is 1 more than the number in `(cdr x)`.**

Example 1 (continued)

γ : Define

```
(defun len (... x)
  (if (endp x)
```

α

```
    ( $\beta$  ... (car x) ... (len ... (cdr x))))))
```

Example 1 (continued)

γ : Define

```
(defun len (x)
  (if (endp x)
```

α

```
    ( $\beta$  ... (car x) ... (len (cdr x))))))
```

Example 1 (continued)

γ : Define

```
(defun len (x)
  (if (endp x)
      0
      ( $\beta$  ... (car x) ... (len (cdr x)))))
```

Example 1 (continued)

γ : Define

```
(defun len (x)
  (if (endp x)
      0
      (+ 1 (len (cdr x)))))
```

Example 2

Define `mem` so that `(mem e x)` is `t` or `nil` depending on whether `e` is an element of `x`.

Example 2

Define `mem` so that `(mem e x)` is `t` or `nil` depending on whether `e` is an element of `x`.

α : Is `e` an element of the empty list? `nil`

Example 2

Define `mem` so that `(mem e x)` is `t` or `nil` depending on whether `e` is an element of `x`.

α : Is `e` an element of the empty list? `nil`

α : How is the answer for `x` related to the answer for `(cdr x)`? If `e` is `(car x)`, then the answer is `t` and it doesn't matter what the answer for `(cdr x)` is; but if `e` is not `(car x)`, the answer for `x` is the same as the answer for `(cdr x)`

Example 2 (continued)

```
(defun mem (e x)
```

```
  (if (endp x)
```

α

```
    ( $\beta$  ... (car x) ... (mem e (cdr x))))))
```

Example 2 (continued)

```
(defun mem (e x)
  (if (endp x)
      nil
      ( $\beta$  ... (car x) ... (mem e (cdr x)))))
```

Example 2 (continued)

```
(defun mem (e x)
  (if (endp x)
      nil
      (if (equal e (car x)) t (mem e (cdr x)))))
```