CS313K: Logic, Sets, and Functions

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

Lecture 3 – Chap 2 (2.3, 2.4, 2.5, 2.6)

Announcements

We have changed the TA office hours to provide an evening slot. See the web page.

Question 34 (page 22) should ask "what is the value of `(fn-q34 x)` when `x` is a natural number?" instead mentioning the ill-formed `(fn-q34 x y)`.

Why did I make you define simple functions (like `plus`, i.e., +) recursively? Because it shows how powerful recursion is: with recursion and a few primitives you can compute anything that can be computed.

Much of computing is learning how to use what you have to build what you want. Hw1 made you practice that by giving you a very small toolbox.

Attend the TA section you're enrolled in. If you attended the wrong section last Friday I'll give credit for attendance, but I won't give credit this week or in the future.

About IF for Non-Programmers

$$(\texttt{IF}\ x\ y\ z) = \begin{cases} y & \text{if } x \neq \texttt{nil} \\ z & \text{otherwise} \end{cases}$$

$$(\texttt{IF}\ x\ y\ z) = \begin{cases} z & \text{if } x = \texttt{nil} \\ y & \text{otherwise} \end{cases}$$

```
(defun fact (n)
  (if (zp n)
      1
      (* n (fact (+ n -1)))))

(fact 3)
```

$=$ {definition of `fact`}

```
(if (zp 3)
    1
    (* 3 (fact (+ 3 -1))))
```

```
(if (zp 3)
    1
    (* 3 (fact (+ 3 -1))))
```
$=$ {since `(zp 3)` is `nil` (3 is not 0)}
```
(if nil
    1
    (* 3 (fact (+ 3 -1))))
```
$=$ {evaluation of `if`}
```
(* 3 (fact (+ 3 -1)))
```
$=$ {evaluation of `+`}
```
(* 3 (fact 2))
```

```
(* 3 (fact 2))
```
$=$ {definition of `fact`}
```
(* 3 (if (zp 2)
       1
       (* 2 (fact (+ 2 -1)))))
```
$=$ {since `(zp 2)` is `nil` (2 is not 0)}
```
(* 3 (if nil
       1
       (* 2 (fact (+ 2 -1)))))
```
$=$ {evaluation of `if`}
```
(* 3 (* 2 (fact (+ 2 -1))))
```

```
(* 3 (* 2 (fact (+ 2 -1))))
```
$=$ {evaluation of `+`}
```
(* 3 (* 2 (fact 1)))
```
$=$ {definition of `fact`}
```
(* 3 (* 2 (if (zp 1)
            1
            (* 1 (fact (+ 1 -1))))))
```
$=$ {since `(zp 1)` is `nil` (1 is not 0)}
```
(* 3 (* 2 (if nil
            1
            (* 1 (fact (+ 1 -1))))))
```

```
(* 3 (* 2 (if nil
                1
                (* 1 (fact (+ 1 -1))))))
```
$=$ {evaluation of `if`}
```
(* 3 (* 2 (* 1 (fact (+ 1 -1)))))
```
$=$ {evaluation of `+`}
```
(* 3 (* 2 (* 1 (fact 0))))
```
$=$ {definition of `fact`}
```
(* 3 (* 2 (* 1 (if (zp 0)
                     1
                     (* 0 (fact (+ 0 -1)))))))
```

```
(* 3 (* 2 (* 1 (if (zp 0)
                    1
                    (* 0 (fact (+ 0 -1)))))))
```
= {since `(zp 0)` is `t`}
```
(* 3 (* 2 (* 1 (if t
                    1
                    (* 0 (fact (+ 0 -1)))))))
```
= {evaluation of `if`}
```
(* 3 (* 2 (* 1 1)))
```
= {evaluation of `*` three times}
```
6
```