CS 135 Fall 2015

Tutorial 2: Stepping and Structures

Announcements

- MarkUs Basic tests:
  - Are set up for every assignment
  - Do not thoroughly test your code
  - Ensure we can run more thorough tests on your code after the due date
  - The results are automatically emailed to your uWaterloo email. You can also check the results on CS 135 Course Website by clicking Basic Tests under Assignments.
  - Are not related to the tests in your code

Announcements

- Tutorial Centre:
  - Location: MC 4065
  - Hours are posted on the Office & Consulting Hours page of course webpage
  - If you need help outside of these hours, please email the course account
- Professor Office Hours:
  - Office hours in professors' offices
  - Time and office are posted on the Office & Consulting Hours pages of the course webpage

Announcements

- Midterm 1:
  - Time: Monday, October 5, 2015, 7:00PM - 8:50PM
  - Exam seating is now available on the course webpage
  - The first midterm covers Modules 1 - 3 inclusive. It does not involve structures.
  - BRING your WatCard, pens and pencils
  - DO NOT bring books, notes, calculators, electronics - no aids are allowed

Group Problem - Converting cond to booleans

Write an equivalent expression without cond. You may use and, or and not.

```
(cond
  [(p1? x) (p2? x)]
  [else true])
```


Group Problem - Rearranging cond

Write an equivalent cond expression using only a single cond. You may use and and or.

```
(cond
  [(p1? x) (cond
             [(p2? x) (f1 x)]
             [(p3? x) (f2 x)]
             [else (f3 x)])]
  [else (f4 x)])
```

Note: Tests for conditional expressions

- Test all boundary points
- Write at least one test for each interval (not including the boundary)
- Tests should be simple and direct, aimed at testing that answer
- DrRacket highlights unused code
  - Having no code highlighted does not mean that your code is fully tested
  - However highlighted code means your testing is incomplete

Clicker Question - Testing

```
(define (foo x)
  (cond
    [(< x 0) (exp x)]
    [(<= x 100) (sub1 x)]
    [(< x 1000) x]
    [else (add1 x)]))
```

Minimally, how many tests would be required for this function?

A 4
B 5
C 6
D 7
E 8

Review: Stepping Rules

Application of built-in functions: (f v1 ... vn) yields v where f is a built-in function and v is the value of $f(v_1, \ldots, v_n)$.

Substitution of Constants: id yields val, where (define id val) occurs to the left.

Review: Stepping Rules

Application of user-defined functions: The general substitution rule is:
(f v1 . . . vn) yields exp'
where (define (f x1 . . . xn) exp) occurs to the left, and exp' is obtained by substituting into the expression exp, with all occurrences of the formal parameter xi replaced by the value vi (for i from 1 to n).

Group Question - Stepping sum-of-squares

The following have been processed in the Beginning Student language:

```
(define (sum-of-squares x y)
  (+ (sqr x) (sqr y)))
```

Step through the following:
(sum-of-squares 3 4)

Review: Stepping Rules

Substitution in cond expressions

There are three rules: when the first expression is false, when it is true, and when it is else.

(cond [false exp] . . .) yields (cond . . .).

(cond [true exp] . . .) yields exp.

(cond [else exp]) yields exp.

These suffice to simplify any cond expression. Here we are using an omission ellipsis to avoid specifying the remaining clauses in the cond.

Group Question - Stepping cond

The following have been processed in the Beginning Student language:

```
(define x 1)
(define y 1)
```

Step through the following:

```
(cond [(= x 0) 'one]
      [else (< (/ y x) c)])
```


Review: Stepping Rules

Simplification Rules for and and or

The simplification rules we use for Boolean expressions involving and and or differ from the ones the Stepper in DrRacket uses in the intermediate steps.

(and false ...) yields false.

(and true ...) yields (and ...).

(and) yields true.

(or true ...) yields true.

(or false ...) yields (or ...).

(or) yields false.

Group Question - Stepping and

The following have been processed in the Beginning Student language:

```
(define x 0)
(define y (+ x 1))
```

Step through the following:

```
(and (not (= x 0)) (<= (/ y x) c))
```

Review: Posn structures

- **constructor** function make-posn, with contract

  ;; make-posn: Num Num → Posn

- **selector** functions posn-x and posn-y, with contracts

  ;; posn-x: Posn → Num

  ;; posn-y: Posn → Num

Example:

```
(define mypoint (make-posn 8 1))
(posn-x mypoint) => 8
(posn-y mypoint) => 1
```


Review: Posn structures

Possible uses:

- coordinates of a point on a two-dimensional plane
- positions on a screen or in a window
- a geographical position

Note:

- An expression such as (make-posn 8 1) is considered a value, which will not be rewritten by the Stepper or our semantic rules.
- The expression (make-posn (+ 4 4) (- 3 2)) would be rewritten to (eventually) (make-posn 8 1).

Exploring Structures - Student Example

```
(define-struct student (quest-id first-name grade))
;; A Student is a (make-student Str Str Nat)
;; requires: grade <= 100
;;           first character in first-name should be capitalized

(define student1 (make-student "cpt6amrka" "Steve" 52))
(define student2 (make-student "ironman" "Tony" 100))

(student-quest-id student1)
=> "cpt6amrka"
(student-first-name student2)
=> "Tony"
(student-grade student2)
=> 100
```

Group Problem - Area of a Triangle

Write a racket function tri-area that consumes three points (as Posns) and produces the area of a triangle using Shoelace Theorem. Be sure to include a purpose, contract, and examples.

$$Area = |\frac{x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2)}{2}|$$