CS135 Autotest Generator

Sam

Background Information

- Autotest VS Public Test:
- Autotest: on due date by ISAs. Public Test: after each submission by students
- Public Test: small number of tests (1-3 per question)
- Autotest: large number of tests (10-20 per question (Fall2013))
- Public Test: correct name and syntax
- Autotest: correctness (and efficiency for some assignments)

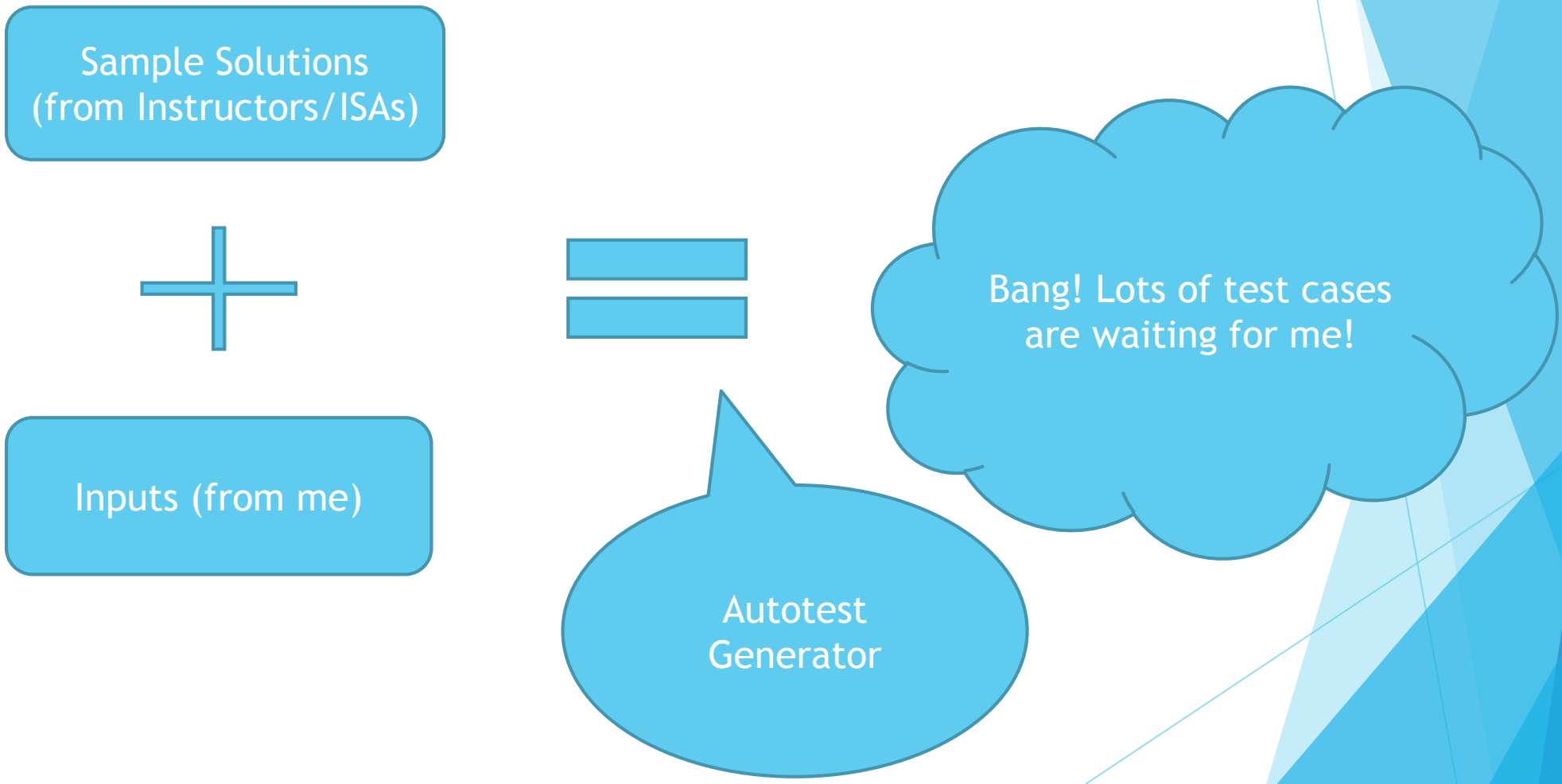
Dilemma between more tests and less debugging

- More test cases
- -> Higher possibility to find wrong answers
- -> More feedbacks to students
- -> More time and effort on debugging
- -> Less time on other things
- -> No time relax (no chatting, playing video games, watching YouTube)

What about:

- Less test cases -> less debugging -> everyone gets perfect -> no complaints ☺
- -> NO! I will get fired……☹

My thoughts


Sample Solutions
(from Instructors/ISAs)
Inputs (from me)
Autotest
Generator
Bang! Lots of test cases
are waiting for me!

Objectives

- Preparations:
- 1. understand how “rst”(run tests) command works (go to ISG TWIKI. Link: https://cs.uwaterloo.ca/twiki/view/ISG/RST)
- 2. read and understand the file system of “test.0”(contains all autotests) from AutotestCreationFall2011.docx created by YenTingChen (go to ISG TWIKI. Link: https://cs.uwaterloo.ca/twiki/view/ISG/CompSci135)
- 3. I will create documentations for my code and user guide later!

Objectives

- Autotest Generator is able to handle:
- 1. different language levels for course CS135:

a) B (Beginning Student)
b) BL (Beginning Student w/ List Abb.)
c) I (Intermediate Student)
d) IL (Intermediate Student w/ Lambda)

Objectives

- Autotest Generator is able to handle:
- 2. Students' answers require some teaching pack/user defined packs
- 3. Set up forbidden functions checking (original code provided by Nick Lee)
- 4. Set up the test running environment (other important settings for tests)
- 5. Generate tests for "rst" in several seconds!
- 6. Generate big data for efficiency test!

Demo – Assignment ☺

Assignment ☺

Language: Intermediate Student w/ Lambda

Due date: ☹ there's no due date

Q1: write a function called sum1, which consumes a list of Nat, and produces the sum of all the elements in that list. (submit "sum1.rkt")

Eg. (sum1 (list 1 2 3)) => 6 (testgen mode: 'list)

Q2: write a function called sum2, which consumes a list of Floating Num, and produces the sum of all the elements in that list. Hint: use check-within to test. (submit "sum2.rkt")

Eg. (sum2 (list -1.1 1.1 2.45)) => 2.45 (testgen mode: 'list with check-within)

Q3: write a function called tree-copy, which cosumes a BT, and produces a copy of that BT ;;A BT is either empty, or (make-btnode num BT BT) (submit "tree-copy.rkt")

Eg. (tree-copy (make-btnode 1 empty empty)) => (make-btnode 1 empty empty) (testgen mode: 'non-list)

Demo – Assignment ☺

Q4: (from Fall2013CS135 A10 BONUS)

Write a function called subsets1, which consumes a list of numbers and produces a list of all of its subsets. (submit “subset1.rkt”)

Eg. (subsets1 ’(1 2)) => (list ’(1 2) ’(1) ’(2) ’()). (testgen mode: ‘custom)

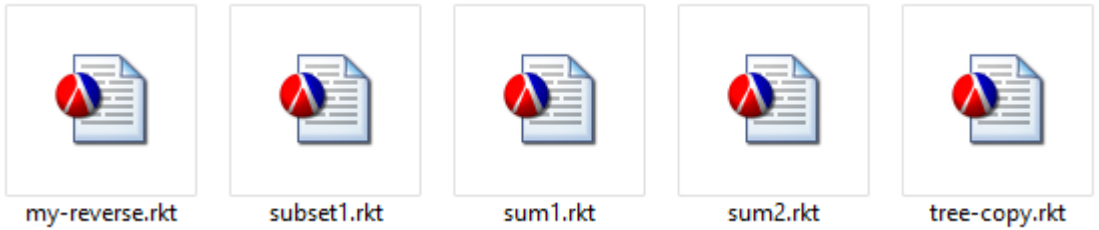
Q5: (forbidden functions)

write a function called my-reverse, which consumes a list of any value, and produce that list in reverse order. You cannot use reverse.(submit “my-reverse.rkt”)

Eg. (my-reverse (list 1 2 3)) => (list 3 2 1) (set up forbidden functions)

Autotest Generator

- 1. You need sample solutions



Autotest Generator

- 2. Copy all solution in one single file (ie. assn.rkt)

```
(define (sum1 lon)
  (foldr + 0 lon))

(define (sum2 lon)
  (foldr + 0 lon))

(define-struct btnode (v l r))
(define (tree-copy t)
  t)

(define (subsets1 lon)
```

Sorry for that. subsets1 is a bonus question for a10.

```
(define (my-reverse lon)
  (reverse lon))
```

Autotest Generator

- 2. put testgen.ss into the same directory

  change to lang racket

  put (require “testgen.ss”) after that

```
#lang racket
(require "testgen.ss")

(define (sum1 lon)
  (foldr + 0 lon))

(define (sum2 lon)
  (foldr + 0 lon))

(define-struct btnode (v l r))
(define (tree-copy t)
  t)
```

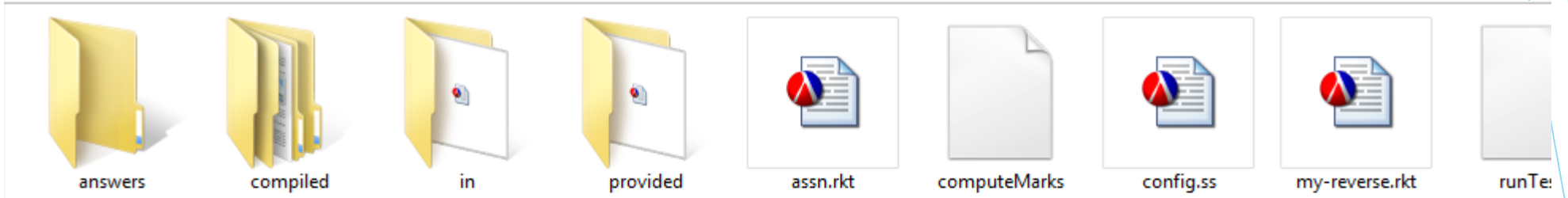
Autotest Generator

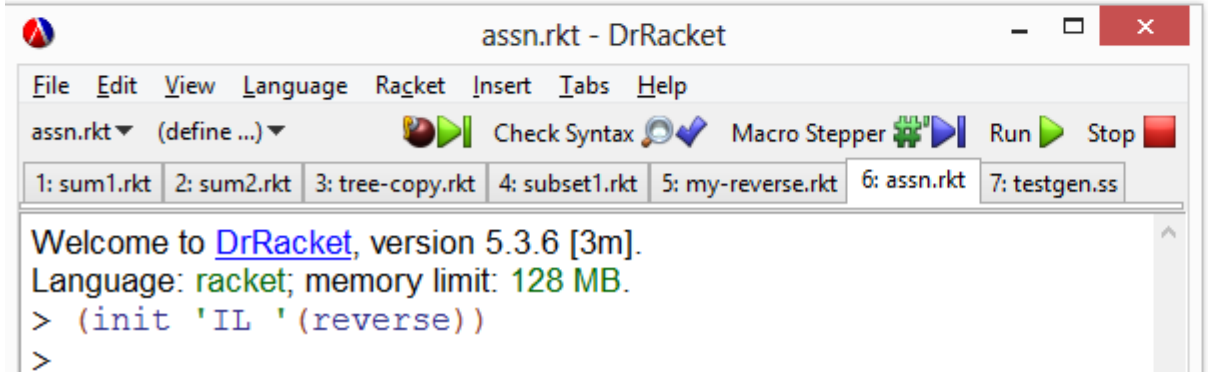
- 3. create test cases (a list of inputs) for each question:

```
;; Test
(define testq1 (list empty
                     (list 1 2 3)
                     (list -1 0 1)))
(define testq2 (list empty
                     (list 0 1 -1)
                     (list 1.2 2.4 3.6 4.8)
                     (list -1.256 -1.7779 9.999)))
(define testq3 (list empty
                     (make-btnode 1 (make-btnode 2 empty empty) empty)))
(define testq4 (list empty
                     (list 1)
                     (list 1 2 3)))
(define testq5 (list empty
                     (list 1 2 3)))
```

Autotest Generator

- 3. Initialization (setting up the test environment and forbidden functions):





Autotest Generator

- 3. run testgen for q1-q3:

```
;; Testing procedure
(init 'IL '(reverse))
(testgen "q1" "sum1.rkt" sum1 testq1 'list)
(testgen "q2" "sum2.rkt" sum2 testq2 'list 0.01)

(set-convertion (list btnode? bt2str))
(testgen "q3" "tree-copy.rkt" tree-copy testq3)
(reset-convertion)
```

```
Welcome to DrRacket, version 5.3.6 [3m].
Language: racket; memory limit: 128 MB.
Question q1. Sumbit File: sum1.rkt. Function: sum1 . TestGene Mode: list. Done!
Question q2. Sumbit File: sum2.rkt. Function: sum2 . TestGene Mode: list. Done!
Question q3. Sumbit File: tree-copy.rkt. Function: tree-copy . TestGene Mode: non-list. Done!
>
```

Autotest Generator

- 4. run testgen for q4-q5:

```
;; Question 4 using genetest mode: 'custom
(define test4bstr"
(result (local[(define (lists-equiv? l1 l2)
                   (and (= (length l1) (length l2))
                        (andmap (lambda (x1) (ormap (lambda (x2) (equal? x1 x2)) l2)) l1)
                        (andmap (lambda (x2) (ormap (lambda (x1) (equal? x1 x2)) l1)) l2)))]
     (lists-equiv? (subsets1 ~a)
                   ~a)))
(expected true)
")
(testgen "q4" "subsets1.rkt" subsets1 testq4 'custom test4bstr)

(testgen "q5" "my-reverse.rkt" my-reverse testq5 'list)
```

```
Welcome to DrRacket, version 5.3.6 [3m].
Language: racket; memory limit: 128 MB.
Question q1. Sumbit File: sum1.rkt. Function: sum1 . TestGene Mode: list. Done!
Question q2. Sumbit File: sum2.rkt. Function: sum2 . TestGene Mode: list. Done!
Question q3. Sumbit File: tree-copy.rkt. Function: tree-copy . TestGene Mode: non-list. Done!
Question q4. Sumbit File: subsets1.rkt. Function: subsets1 . TestGene Mode: custom. Done!
Question q5. Sumbit File: my-reverse.rkt. Function: my-reverse . TestGene Mode: list. Done!
>
```

Autotest Generator

- 5. now copy assn.rkt testgen.ss to course accout and run it!

```
cs135@linux024:/u3/cs135/marking/trsong_test/test.0$ ls
assn.rkt  testgen.ss
cs135@linux024:/u3/cs135/marking/trsong_test/test.0$ racket assn.rkt
Question q1. Sumbit File: sum1.rkt. Function: sum1 . TestGene Mode: list. Done!
Question q2. Sumbit File: sum2.rkt. Function: sum2 . TestGene Mode: list. Done!
Question q3. Sumbit File: tree-copy.rkt. Function: tree-copy . TestGene Mode: non-list. Done!
Question q4. Sumbit File: subsets1.rkt. Function: subsets1 . TestGene Mode: custom. Done!
Question q5. Sumbit File: my-reverse.rkt. Function: my-reverse . TestGene Mode: list. Done!
cs135@linux024:/u3/cs135/marking/trsong_test/test.0$
```

Autotest Generator

- 5. now submit your solutions, and let's start testing

```
Running test q2_001
  Mark is 100%

Running test q2_002
  Mark is 100%

Running test q2_003
  Mark is 100%

Running test q2_004
  Mark is 100%

Running test q3_001
  Mark is 100%

Running test q3_002
  Mark is 100%

Running test q4_001
  Mark is 100%

Running test q4_002
  Mark is 100%

Running test q4_003
  Mark is 100%

Running test q5_001
   (reverse ...) is not allowed.
 Exception234 caught when running test: (reverse ...) is not allowed.
  Mark is 0%

Running test q5_002
   (reverse ...) is not allowed.
 Exception234 caught when running test: (reverse ...) is not allowed.
  Mark is 0%
Finished running tests. Continuing with computeMarks...
Doing diff checks

12/14   Automarking total

 ** Question q1: 3/3
 ** Question q2: 4/4
 ** Question q3: 2/2
 ** Question q4: 3/3
 ** Question q5: 0/2
```

For q5 I did use forbidden function "reverse".
It works!

Conclusions

- 1. Autotest Generator can save us a lot of time and effort if we use it wisely
- 2. By following this tutorial, anyone can write Autotest for cs135
- 3. This tutorial also works for CS 115 and part of CS116

Future Work

- 1. Provide enough documentations / video demos on ISG TWIKI
- 2. Add “Advance Student” language
- 3. Debug (I just finish coding, so very little time to debug)

Thank you!

- **1. Hope you have a fun time being a TUTOR!!!**
- **2. Hope you enjoy this tutorial!**