RacketCon 2017

High-Coverage Hint Generation for Racket Programming Assignments

Phitchaya Mangpo Phothilimthana
Sumukh Sridhara

University of California, Berkeley

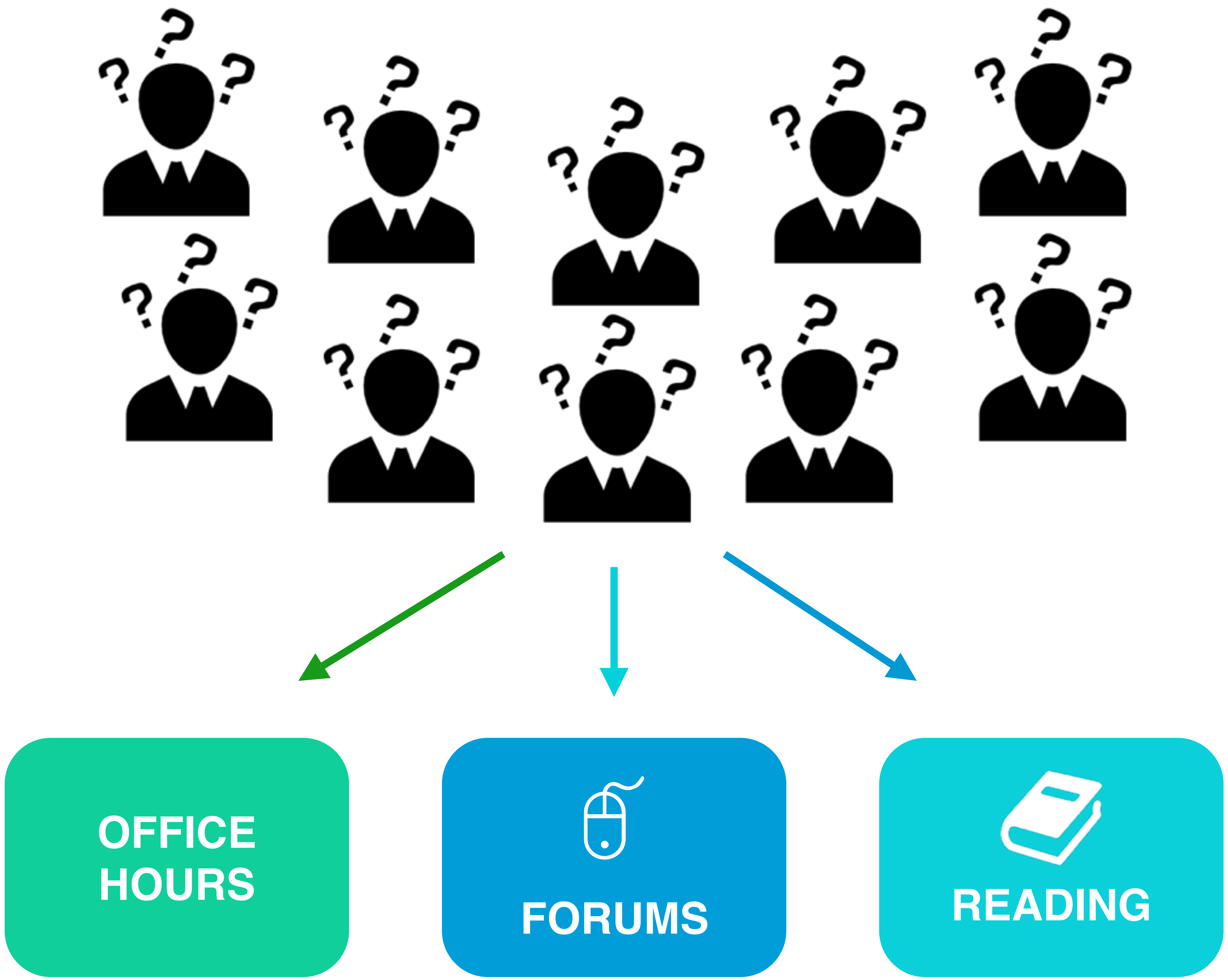
CS61A @ UC Berkeley

[cs61a.org](cs61a.org)

In-Person CS1 Course - Enrollment: 1600/semester

Hint system deployment: Spring 2016 — Present

Getting Help (in large courses)



OFFICE HOURS

FORUMS

READING

Automated Hint Generation System

Goals

1. **Hints that are useful but do not give away answers.**
2. **Robust (always able to produce a hint)**
3. **Easy to operate for instructors**

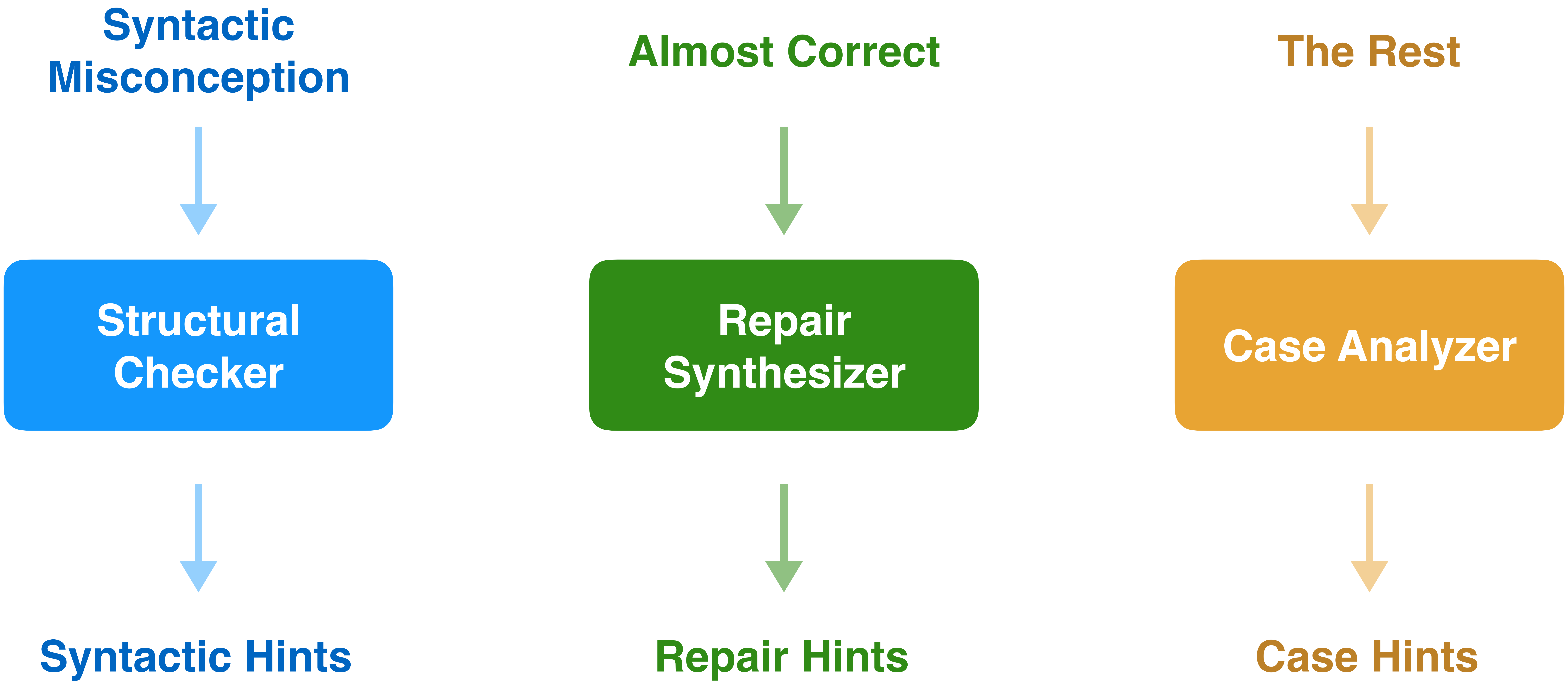
Types of Errors Analyzed from Past Data

Syntactic Misconception

Almost Correct

The Rest

Types of Errors Analyzed from Past Data



Syntactic Misconception

Structural Checker

Syntactic Hints

Almost Correct

Repair Synthesizer

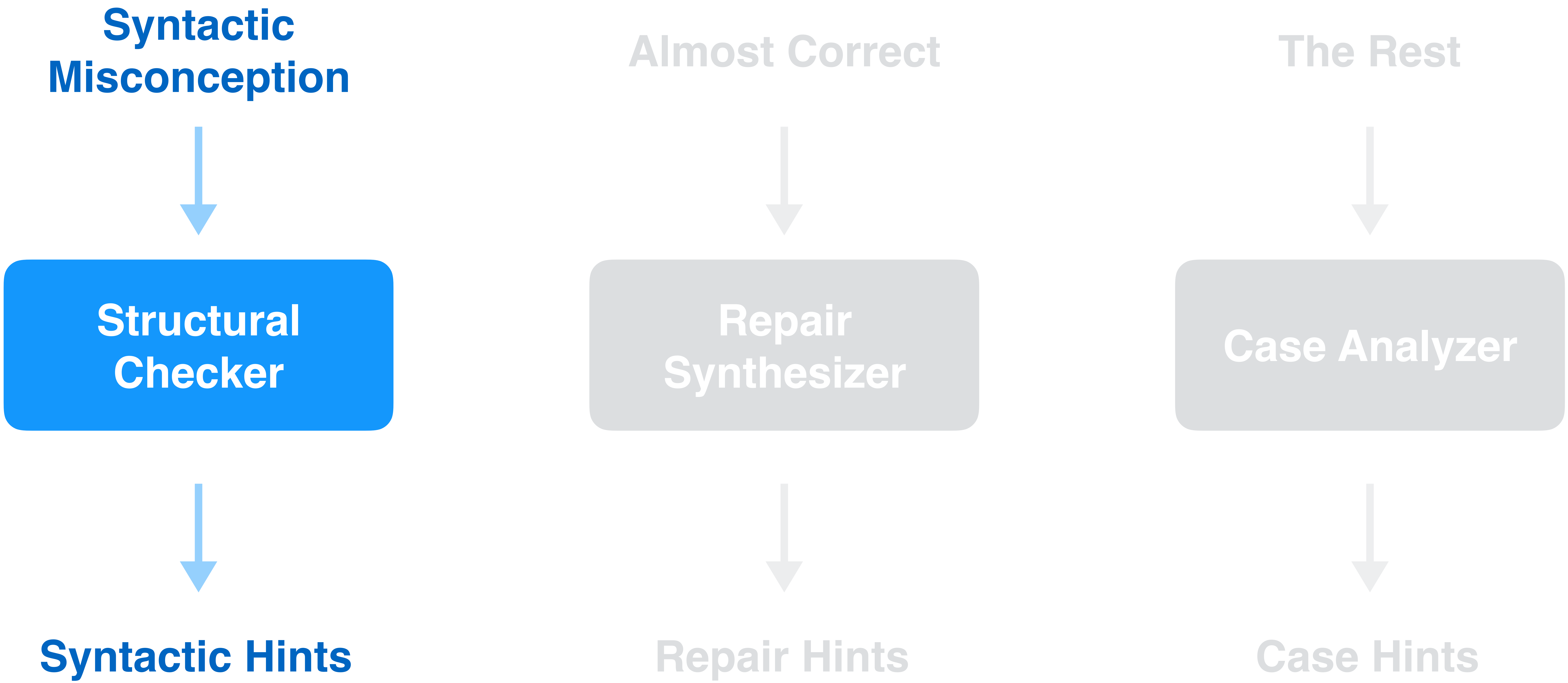
Repair Hints

The Rest

Case Analyzer

Case Hints

Structural Checker



Syntactic Misconception

Structural Checker

Syntactic Hints

Almost Correct

Repair Synthesizer

Repair Hints

The Rest

Case Analyzer

Case Hints

Syntactic Misconceptions —> Structural Checker

Student’s program:

```
(define (pow b n)
  (define (square x) (* x x))
  (cond ((= n 0) 1)
        ((even? n) (square (pow b (/ n 2))))
        ((odd? n) (* b (pow b (- n 1))))
))
```

missing parentheses

High-level hint:

The computer thinks that your program misses or has extra pairs of parentheses.

Detailed hint:

>> Syntax expert:
Check the syntax of the conditional clause **at line 4, 5**.

Example(s) of correct syntax:

```
(cond ((> a b) (* a b)) (else (func a b)))
```

Example(s) of bad syntax:

```
(cond (> a b) (* a b) else (func a b))
```

How Does Structural Checker Work?

Simple pattern matching

Construct	Error Pattern	Example

`cond`	missing a test expression or a body	`(cond ((> a b) #t)`
		`(🟥 #f))`

`cond`	missing a pair of parentheses around a body	`(cond ((> a b) (* a b))`
		`(else (func a b)))`

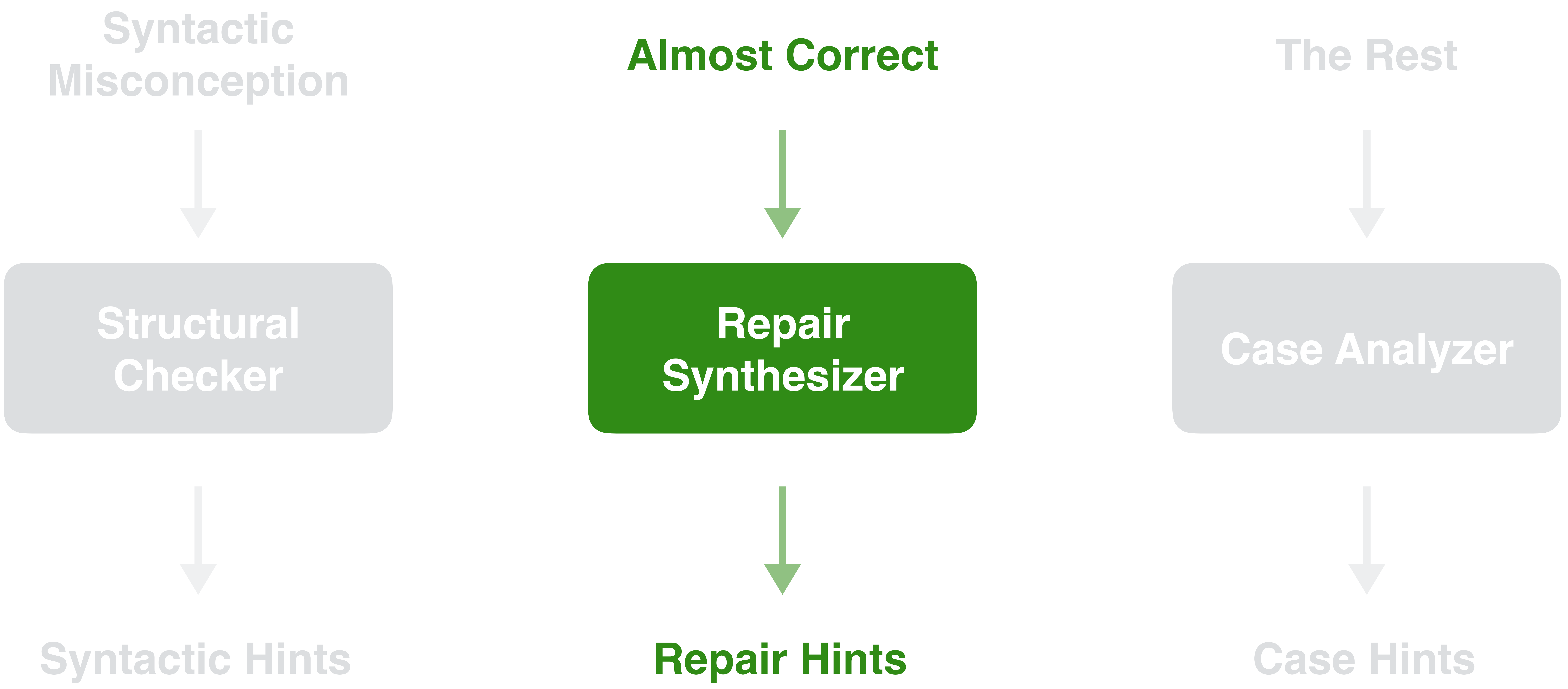
`cond`	missing a pair of parentheses around a test expression	`(cond ((> a b) #t)`
		`(else #f))`

`cond`	missing a pair of parentheses around a pair of test expression and body	`(cond ((> a b) #t)`
		`(else #f))`

`if`	not matching (it test-expo then-expo else-expr)	`(if (< a b) #t 🟥)`
`define`	no body	`(define (min a b) 🟥)`

`define`	multiple bodies that return non-void values	`(define (min a b)`
		`(if (<= a b) a)`
		`(if (<= b a) b))`

Repair Synthesizer



Syntactic Misconception

Structural Checker

Syntactic Hints

Almost Correct

Repair Synthesizer

Repair Hints

The Rest

Case Analyzer

Case Hints

Almost Correct —> Repair Synthesizer

Student's program:

```
1 (define (square x) (* x x))
2
3 (define (pow b n)
4   (cond
5     ((even? n) (square (pow b (/ n 2))))
6     ((odd? n) (* n (pow b (- n 1))))
7 ))
```

Hint:

The computer thinks that:

1. The body of the body expression at **line 6** has some logical errors.
   What value should you multiply by?
2. You may have **forgotten** to specifically handle some of these following cases or handle them incorrectly in function `(pow b n)`:
   `(= n 0)`

How Does Repair Synthesizer Work?

Follow the **mutation-based approach** by *Singh et al., PLDI' 2013*

▶ for **Python** programs
▶ define error models (mutations) by overriding internal functions to mutate different types of AST nodes

- Instructors must know about:
  - mutation functions they need to override
  - provided utility functions that can be used

- A typical implementation of a mutation function for one question requires **300 lines of code.**

How Does Repair Synthesizer Work?

Example error models:

```
(define-error-model          ; rule 1
   [context      '(* ? _)]  [type 'replace]
   [mutate-from '$arg1]     ; arg1 = argument 1 of the function
   [mutate-to   '($arg0)] ; arg0 = argument 0 of the function
   [hint        "What value should you multiply by?")
```

Student's program:

```
(define (pow b n)
  (cond
    ((even? n) (square (pow b (/ n 2))))
    ((odd? n) (* n (pow b (- n 1))))))
)
```

```
(define (pow b n)
  (cond
    ((even? n) (square (pow b (/ n 2))))
    ((odd? n) (* b (pow b (- n 1))))))
)
```

How Does Repair Synthesizer Work?

Example error models:

```
(define-error-model                                  ; rule 2
   [context      ‘(define (pow _ _) ?)]  [type ‘case]
   [mutate-from ‘$x]                     ; $x match anything
   [mutate-to   ‘((cond ((= $arg1 0) 1) (else $x)))])
```

Student’s program:

```
(define (pow b n)
  (cond
    ((even? n) (square (pow b (/ n 2))))
    ((odd? n) (* n (pow b (- n 1)))))
)
```

```
(define (pow b n)
  (cond ((= n 0) 1)
        (else (cond
                ((even? n) (square (pow b (/ n 2))))
                ((odd? n) (* b (pow b (- n 1))))))
)
```

How Does Repair Synthesizer Work?

Student’s program:

```
(define (pow b n)
  (cond
    ((even? n) (square (pow b (/ n 2))))
    ((odd? n) (* n (pow b (- n 1))))
))
```

```
(define c (make-vector 2))
(define (pow b n)
 ((list-ref
   (list
    (lambda ()
      (cond
       ((even? n) (square (pow b (/ n 2))))
       ((odd? n)
        (* (list-ref (list n b) (vector-ref c 0))
           (pow b (- n 1)))))))
    (lambda ()
      (cond
       ((= n 0) 1)
       (else
        (cond
         ((even? n) (square (pow b (/ n 2))))
         ((odd? n)
          (* (list-ref (list n b) (vector-ref c 0))
             (pow b (- n 1))))))))))
   (vector-ref c 1))))
```

rule 2

rule 1

rule 1

rule 2

Search for **c** that make the program correct:

#(0 0)
#(0 1)
#(1 0)
#(1 1)

rule 1 rule 2

How Does Repair Synthesizer Work?

Example student’s program:

```
1 (define (square x) (* x x))
2
3 (define (pow b n)
4   (cond
5     ((even? n) (square (pow b (/ n 2))))
6     ((odd? n) (* n (pow b (- n 1))))
7 ))
```

Hint:

The computer thinks that:

1. The body of the body expression at **line 6** has some logical errors.
   What value should you multiply by? rule 1 `n –> b`
2. You may have **forgotten** to specifically handle some of these following cases or handle them incorrectly in function `(pow b n)`:
   `(= n 0)` rule 2 add a base case

Correct Program?

Soft Correctness

Correct on all test cases

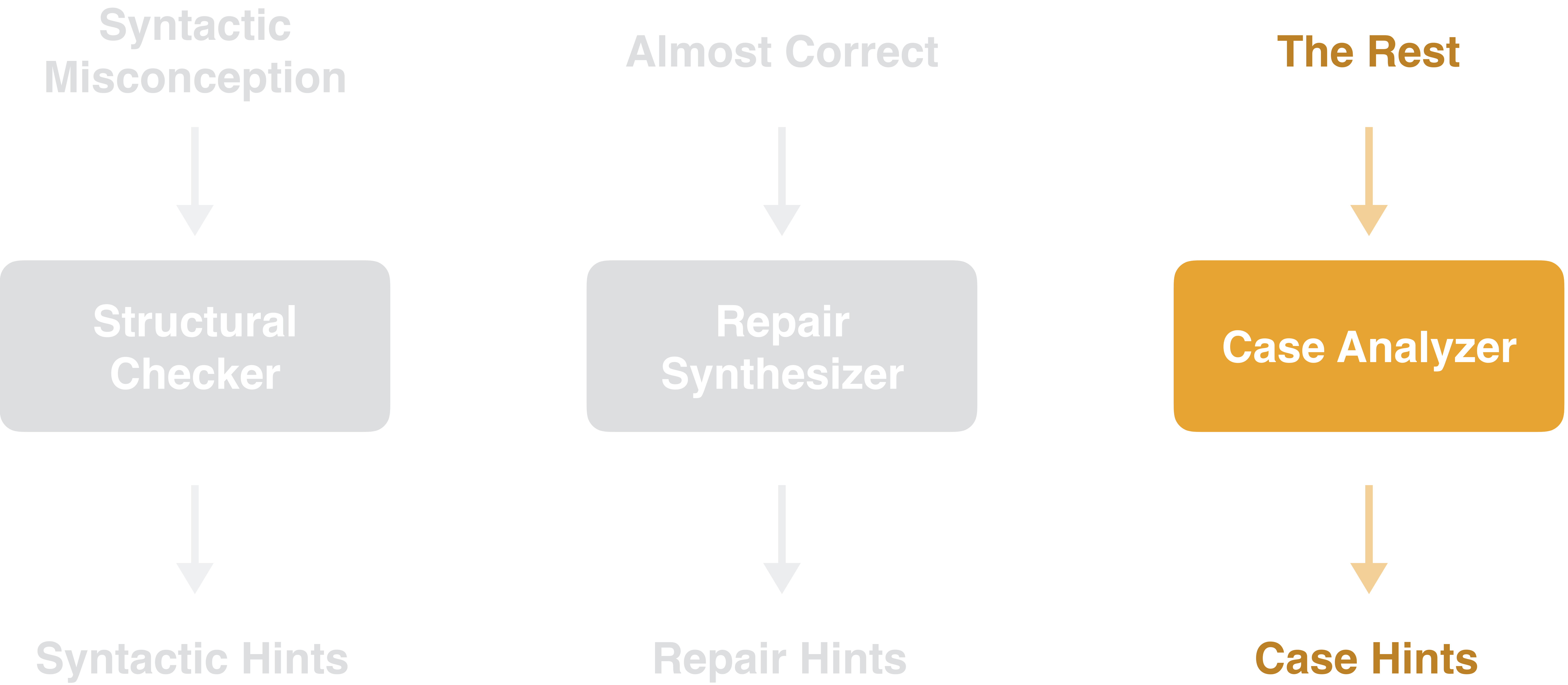
Hard Correctness

Semantically equivalent to the teacher’s solution

- Use **Rosette**, a solver-aided language, embedded in Racket
- Translate code into logical constraints (i.e. SMT)
- Ask SMT to check program equivalence between solution and a mutated program

http://emina.github.io/rosette/

Case Analyzer



Missing Cases —> Case Analyzer

Student's program:

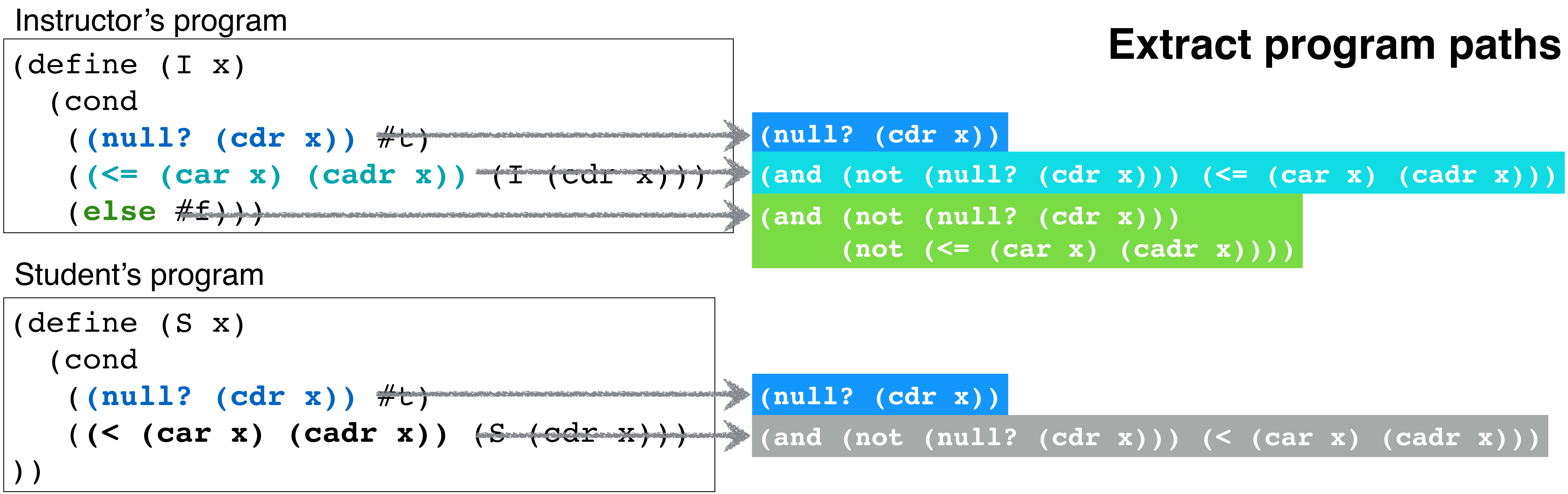
```
(define (S x)
  (cond
   ((null? (cdr x)) #t)
   ((< (car x) (cadr x)) (S (cdr x)))
))
```

Hint:

In your function `(S x)`, what will happen if the **inputs** to the (recursive) function **meet one of the following conditions**?
Does your function **handle these scenarios** correctly?

```
 a. (<= (car x) (cadr x))
 b. (and (not (null? (cdr x)))
         (not (<= (car x) (cadr x))))
```

How Does Case Analyzer Work?



Instructor’s program

```
(define (I x)
  (cond
   ((null? (cdr x)) #t)
   ((<= (car x) (cadr x)) (I (cdr x)))
   (else #f)))
```

Extract program paths

```
(null? (cdr x))
(and (not (null? (cdr x))) (<= (car x) (cadr x)))
(and (not (null? (cdr x)))
     (not (<= (car x) (cadr x))))
```

Student’s program

```
(define (S x)
  (cond
   ((null? (cdr x)) #t)
   ((< (car x) (cadr x)) (S (cdr x)))
))
```

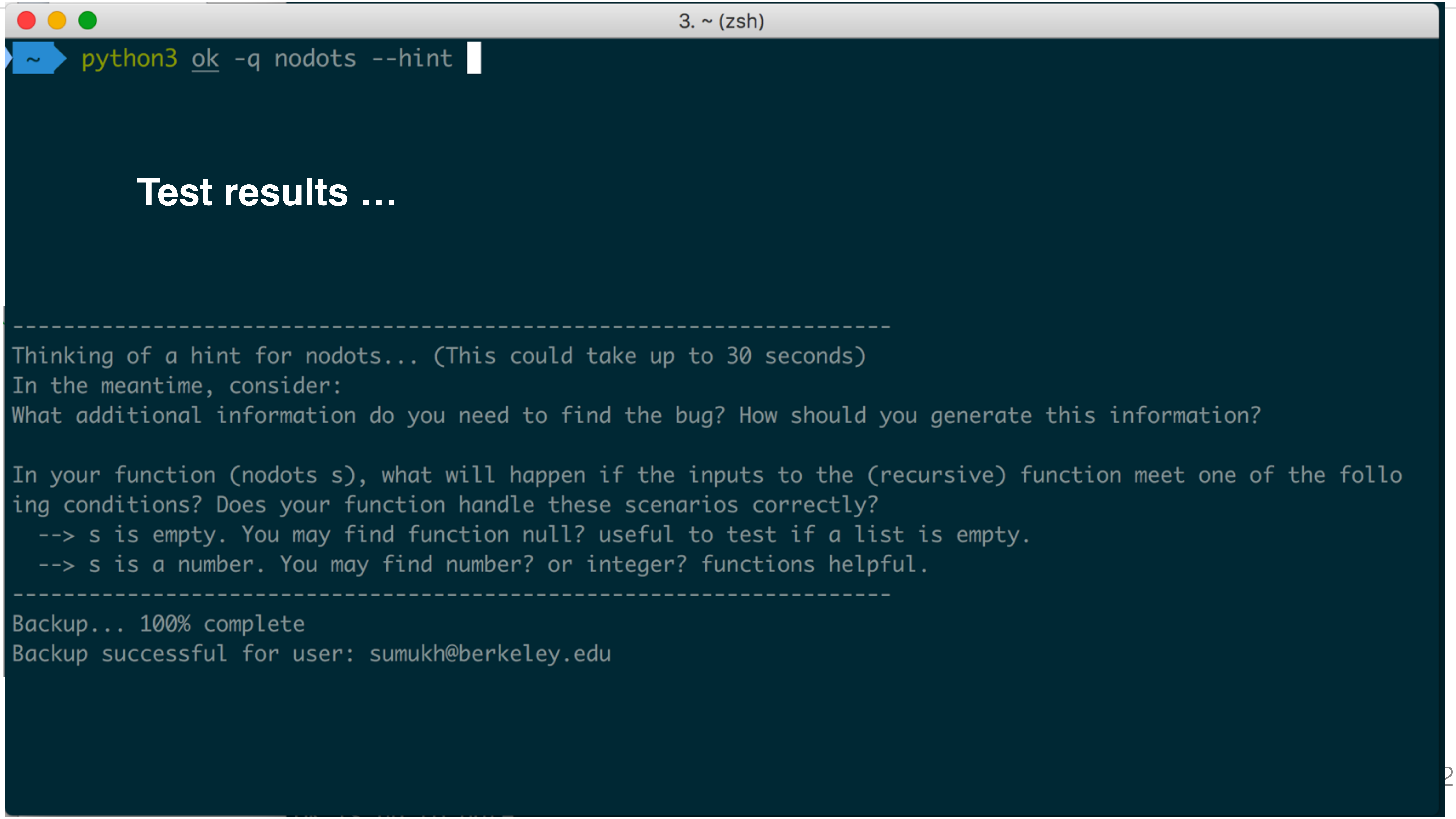
```
(null? (cdr x))
(and (not (null? (cdr x))) (< (car x) (cadr x)))
```

Hint

In your function `(S x)`, what will happen if the inputs to the (recursive) function meet one of the following conditions?
Does your function handle these scenarios correctly?

```
(<= (car x) (cadr x))
(and (not (null? (cdr x)))
     (not (<= (car x) (cadr x))))
```

Asking for Hints

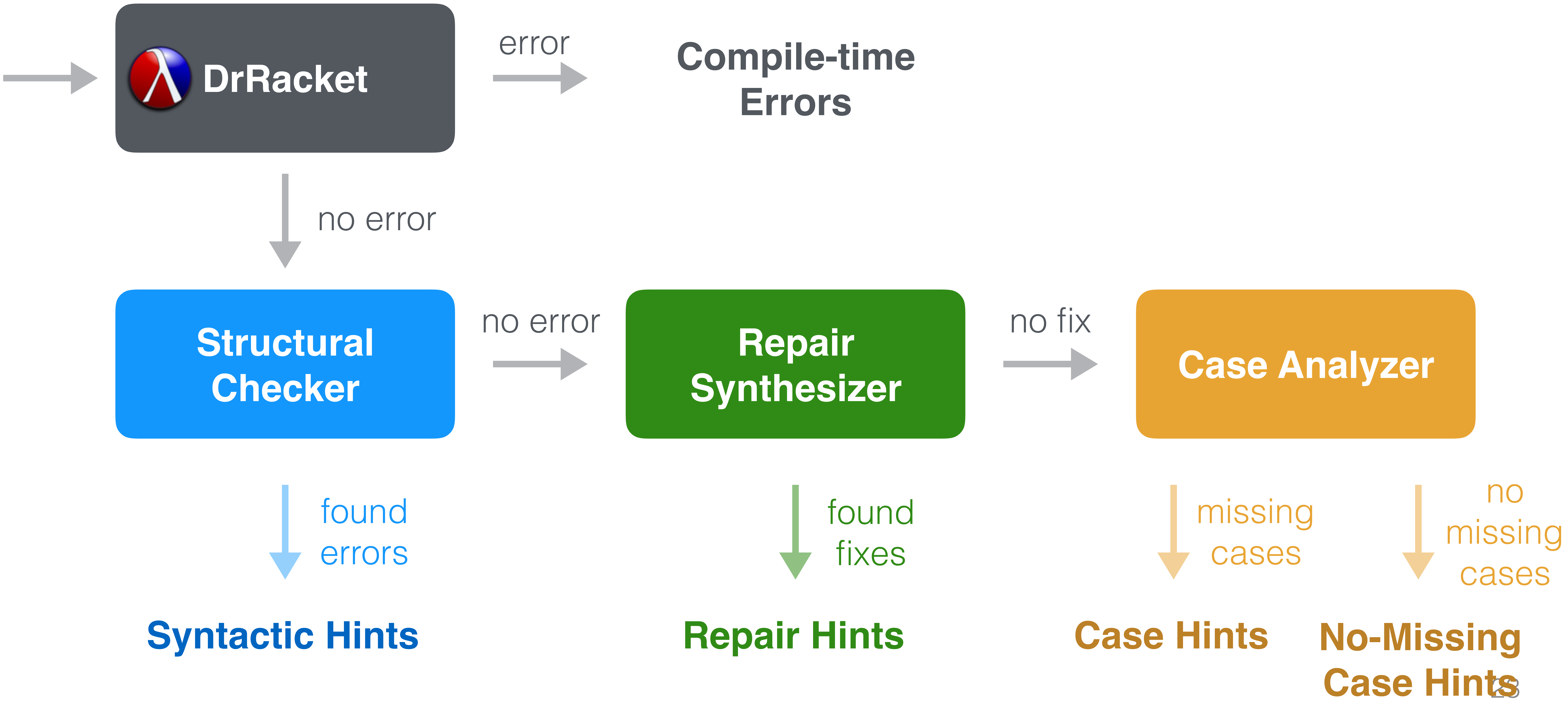


```
~ python3 ok -q nodots --hint
```

Test results …

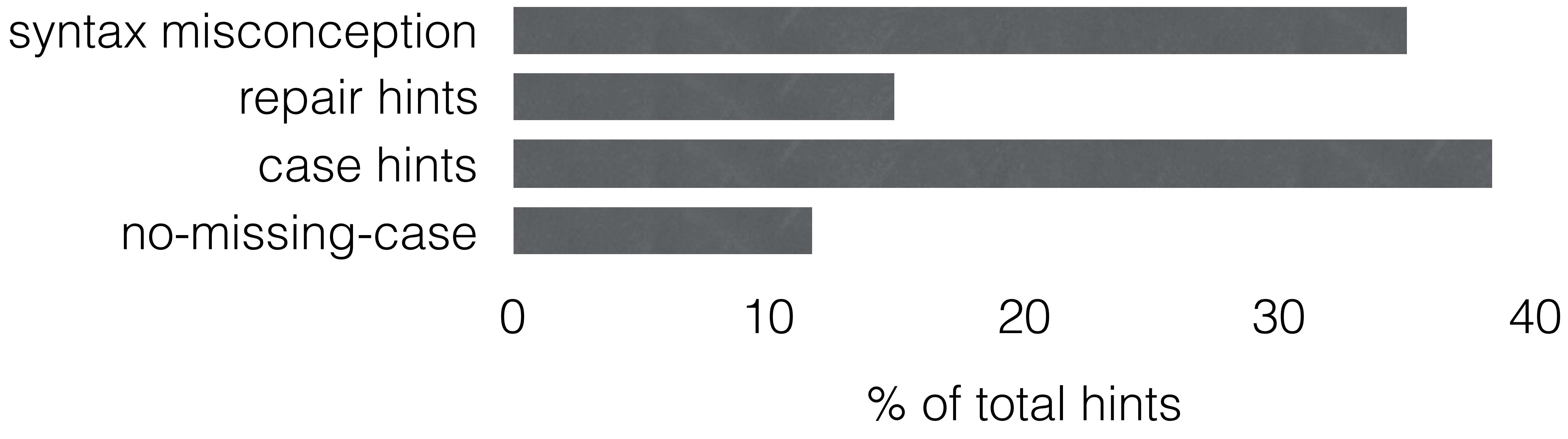
```
---------------------------------------------------------------------
Thinking of a hint for nodots... (This could take up to 30 seconds)
In the meantime, consider:
What additional information do you need to find the bug? How should you generate this information?

In your function (nodots s), what will happen if the inputs to the (recursive) function meet one of the follo
ing conditions? Does your function handle these scenarios correctly?
  --> s is empty. You may find function null? useful to test if a list is empty.
  --> s is a number. You may find number? or integer? functions helpful.
---------------------------------------------------------------------
Backup... 100% complete
Backup successful for user: sumukh@berkeley.edu
```

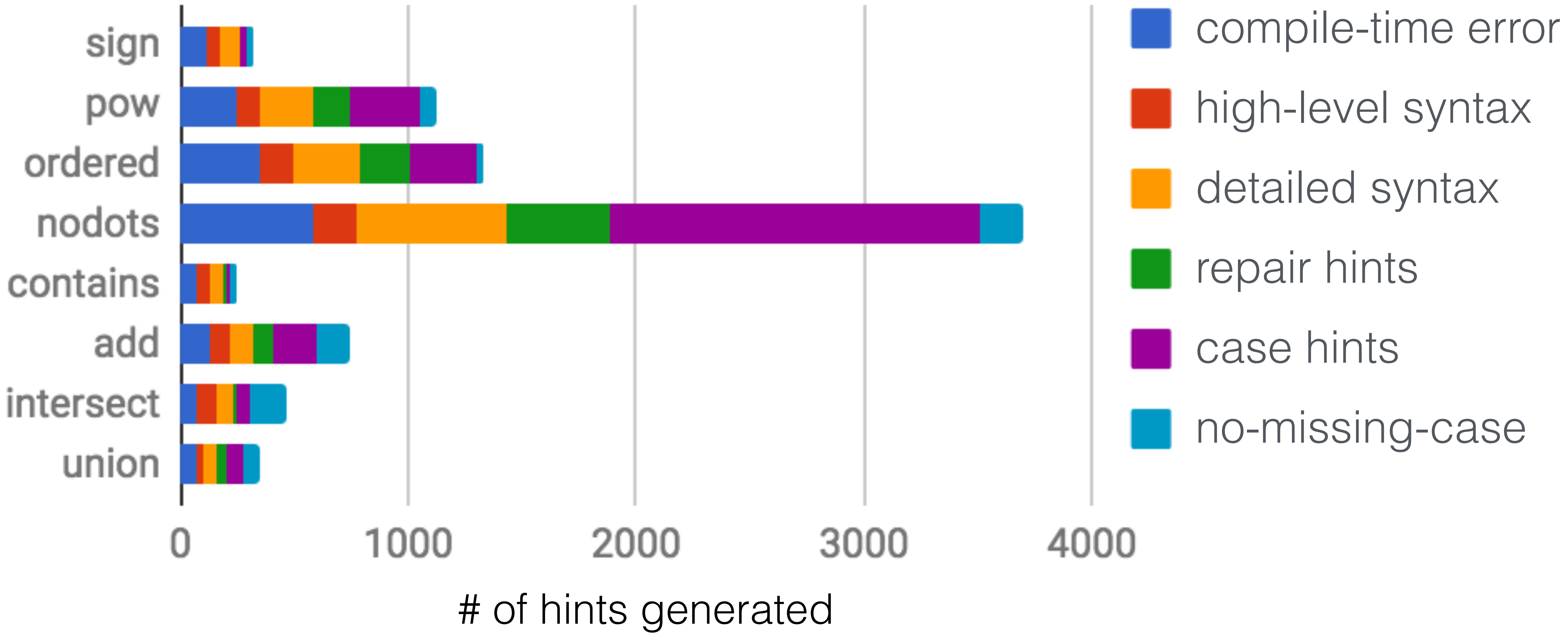
System Implementation



Usage

918 out of 1,485 students asked for hints.





Q1: Did hints help students complete the assignment?

Hints were helpful overall

Compared # of attempts for identical homework across two offerings of the course (one with hints, one without)

- **18% drop** in the number of attempts **when hints are available**
- Statistically significant ($p < 0.001$)
- Students are almost entirely identical demographically

All types of hints were helpful

Syntax misconception hints were extremely effective.

- Students who requested a hint **struggled for 4.7 attempts** (average) on the same error **before receiving hints**.
- Those student **fixed/changed** the error after **2 attempts** (average) **after receiving hints**.

85% of students benefited from **repair hints**.

48% of students benefited from **missing-case/non-missing-case hints**.

Q2: Do students build a dependence?

Do students build a dependence?

Seems like they are not

< 5% of students "abused" the system by asking for more than 8 hints on a question.

Thank you

mangpo@eecs.berkeley.edu

Reference:
High Coverage Hint Generation
for Massive Courses
ITICSE 2017

Some Student Feedback

“Just want to say that the hint function is extremely helpful! It saved me a lot of time and frustration by pointing out something that I would never have thought on my own.”

"It made the homework go faster [because] I didn’t have to wait for office hours or a response on [the online Q&A forum].”