CMSC 430, Feb 18th 2020

Extort

First things first

First things first

- Assignment #2

First things first

- Assignment #2
  - Thanks to those of you that turned it in!

First things first

- Assignment #2
  - Thanks to those of you that turned it in!
  - Hoping to get grading done by the end of the week.

First things first

- Assignment #2
  - Thanks to those of you that turned it in!
  - Hoping to get grading done by the end of the week.
- Two issues:

First things first

- Assignment #2
  - Thanks to those of you that turned it in!
  - Hoping to get grading done by the end of the week.
- Two issues:
  - One of my TAs is going to disambiguate github ID <-> UID. If you have concerns about that, contact me ASAP

First things first

- Assignment #2
  - Thanks to those of you that turned it in!
  - Hoping to get grading done by the end of the week.
- Two issues:
  - One of my TAs is going to disambiguate github ID <-> UID. If you have concerns about that, contact me ASAP
  - Without ELMS/Canvas, how can I best communicate grades?

We’ve been Duped

We've been Duped

- On Thursday we saw that even though we have two types and a semantics for our programs, there's all sorts of undefined behaviour.

We've been Duped

- On Thursday we saw that even though we have two types and a semantics for our programs, there's all sorts of undefined behaviour.
- One strange consequence of this was that our interpreter and compiler behaved differently!

We've been Duped

- On Thursday we saw that even though we have two types and a semantics for our programs, there's all sorts of undefined behaviour.
- One strange consequence of this was that our interpreter and compiler behaved differently!
  - Why?

Addressing the error of our ways

Addressing the error of our ways

- Recap from last time:

Addressing the error of our ways

- Recap from last time:

```
(add1 #f)
```

Addressing the error of our ways

- Recap from last time:

Addressing the error of our ways

- Recap from last time:

```
(zero? #f)
```

Addressing the error of our ways

- Recap from last time:

Addressing the error of our ways

- Recap from last time:

```
(if (zero? #f) 1 2)
```

Addressing the error of our ways

- Recap from last time:

```
(if (zero? #f) 1 2)
```

- Previously, these were undefined

Addressing the error of our ways

- Recap from last time:

```
(if (zero? #f) 1 2)
```

- Previously, these were undefined
  - In our interpreter we would get a failure because of the errors from the underlying Racket execution

Addressing the error of our ways

- Recap from last time:

```
(if (zero? #f) 1 2)
```

- Previously, these were undefined
  - In our interpreter we would get a failure because of the errors from the underlying Racket execution
  - In our compiler we'd get junk

Extort

Extort

- Our language **`extort`** the same as **`dupe`** *except* we address errors explicitely

Extort's AST

Extort's AST

- No changes:

Extort's AST

- No changes:
  - **`e ::= ... | if e e e | zero? e`**

Extort's AST

- No changes:
  - **`e ::= ... | if e e e | zero? e`**
  - Why don't we need to change the AST?

C’est man-tick

C’est man-tick

- Type mismatches in **dupe** were undefined behavior

C’est man-tick

- Type mismatches in **dupe** were undefined behavior
  - Do we have to make them defined?

C’est man-tick

- Type mismatches in **dupe** were undefined behavior
  - Do we have to make them defined?
  - What are the pros/cons?

Errors Rule

Errors Rule

- Let’s add some, knowing that it’s not strictly necessary

Errors Rule

- Let's add some, knowing that it's not strictly necessary
- Our semantics now relate programs to *answers* instead of values

Errors Rule

- Let's add some, knowing that it's not strictly necessary
- Our semantics now relate programs to *answers* instead of values
  - *answers* are either *values* (as before), or *errors*

Errors Rule

- Let's add some, knowing that it's not strictly necessary
- Our semantics now relate programs to *answers* instead of values
  - *answers* are either *values* (as before), or *errors*
- We'll just show the new rules, none of the others have changed.

C’est man-tick

C’est man-tick

- Where can errors occur (currently)?

C’est man-tick

- Where can errors occur (currently)?

$$\frac{}{\boldsymbol{E}[\![(\text{add1 } b),\ \text{err}]\!]}$$

C’est man-tick

- Where can errors occur (currently)?

$$\frac{}{\boldsymbol{E}[\![(\text{add1 } b), \text{err}]\!]}$$

$$\frac{}{\boldsymbol{E}[\![(\text{sub1 } b), \text{err}]\!]}$$

C’est man-tick

- Where can errors occur (currently)?

$$\frac{}{\boldsymbol{E}[\![(\text{add1 } b), \text{err}]\!]}$$

$$\frac{}{\boldsymbol{E}[\![(\text{sub1 } b), \text{err}]\!]}$$

$$\frac{}{\boldsymbol{E}[\![(\text{zero? } b), \text{err}]\!]}$$

C’est man-tick

C’est man-tick

- Is that it?

C’est man-tick

- Is that it?

```
(if (zero? #f) 1 2)
```

C’est man-tick

C’est man-tick

- We also need to propagate errors ’upward’

C’est man-tick

- We also need to propagate errors ’upward’

$$\frac{\boldsymbol{E}[\![e, \text{err}]\!]}{\boldsymbol{E}[\![(\text{zero? } e), \text{err}]\!]}$$

C’est man-tick

- We also need to propagate errors ’upward’

$$\frac{\boldsymbol{E}[\![e, \text{err}]\!]}{\boldsymbol{E}[\![(\text{zero? } e), \text{err}]\!]}$$

$$\frac{\boldsymbol{E}[\![e, \text{err}]\!]}{\boldsymbol{E}[\![(\text{add1 } e), \text{err}]\!]}$$

C’est man-tick

- We also need to propagate errors ’upward’

$$\frac{\boldsymbol{E}[\![e, \text{err}]\!]}{\boldsymbol{E}[\![(\text{zero? } e), \text{err}]\!]}$$

$$\frac{\boldsymbol{E}[\![e, \text{err}]\!]}{\boldsymbol{E}[\![(\text{add1 } e), \text{err}]\!]}$$

$$\frac{\boldsymbol{E}[\![e, \text{err}]\!]}{\boldsymbol{E}[\![(\text{sub1 } e), \text{err}]\!]}$$

C’est man-tick

- We also need to propagate errors ’upward’

$$\frac{\boldsymbol{E}[\![e, \text{err}]\!]}{\boldsymbol{E}[\![(\text{zero? } e), \text{err}]\!]}$$

$$\frac{\boldsymbol{E}[\![e, \text{err}]\!]}{\boldsymbol{E}[\![(\text{add1 } e), \text{err}]\!]}$$

$$\frac{\boldsymbol{E}[\![e, \text{err}]\!]}{\boldsymbol{E}[\![(\text{sub1 } e), \text{err}]\!]}$$

$$\frac{\boldsymbol{E}[\![e, \text{err}]\!]}{\boldsymbol{E}[\![(\text{if } e\ e_0\ e_1), \text{err}]\!]}$$

Rules are the easy part

Rules are the easy part

- How can our implementations match these rules?

Let’s look at the interpreter

We'll do that in the terminal, as it's starting to get a bit too cumbersome

Let's experiment

```
extort> (require "extort_interp.rkt")
```

Now the compiler.

Now the compiler.

- What needs to change, if anything?

Now the compiler.

- What needs to change, if anything?
- What should the error message be?

Runtime errors

Runtime errors

- Things need to happen in the RTS and compiler.

Runtime errors

- Things need to happen in the RTS and compiler.
  - Runtime system?

Runtime errors

- Things need to happen in the RTS and compiler.
  - Runtime system?
  - Compiler?

Let’s take a look at the RTS and compiler

Assignment 3

- ls live

Assignment 3

- Is live
- Due next Tuesday.

Assignment 3

- Is live
- Due next Tuesday.
  - Please tell your fellow students to check the webpage periodically
  - If there are any issues that might make you unable to do the assignment on time, *talk to me*