How to Be Both Lazy and Reasonable

Tim Chevalier

Scholarship Skills

March 17, 2008

Talk Overview

- You want to reason about your programs.
- Programming in a lazy functional language helps.
- Implementing those languages is hard.

Reasoning About Programs

f (someComplicatedFunction())

```
f(int x) {
    if todayIsMonday()
        1
    else
        x
}
```

Can we replace this with 1?

Evaluation Strategies

Strict evaluation

Evaluate function arguments to **values**

Pass **values** to functions

Lazy evaluation (call-by-name)

Represent arguments as **thunks** (delayed computations)

Pass **thunks** to functions

Roadmap

- Reasoning is good. ✓
- Shall we be lazy or strict?
- Laziness has consequences
  - Reasoning about programs: easy
  - Reasoning about implementation: hard
  - Can we do better?

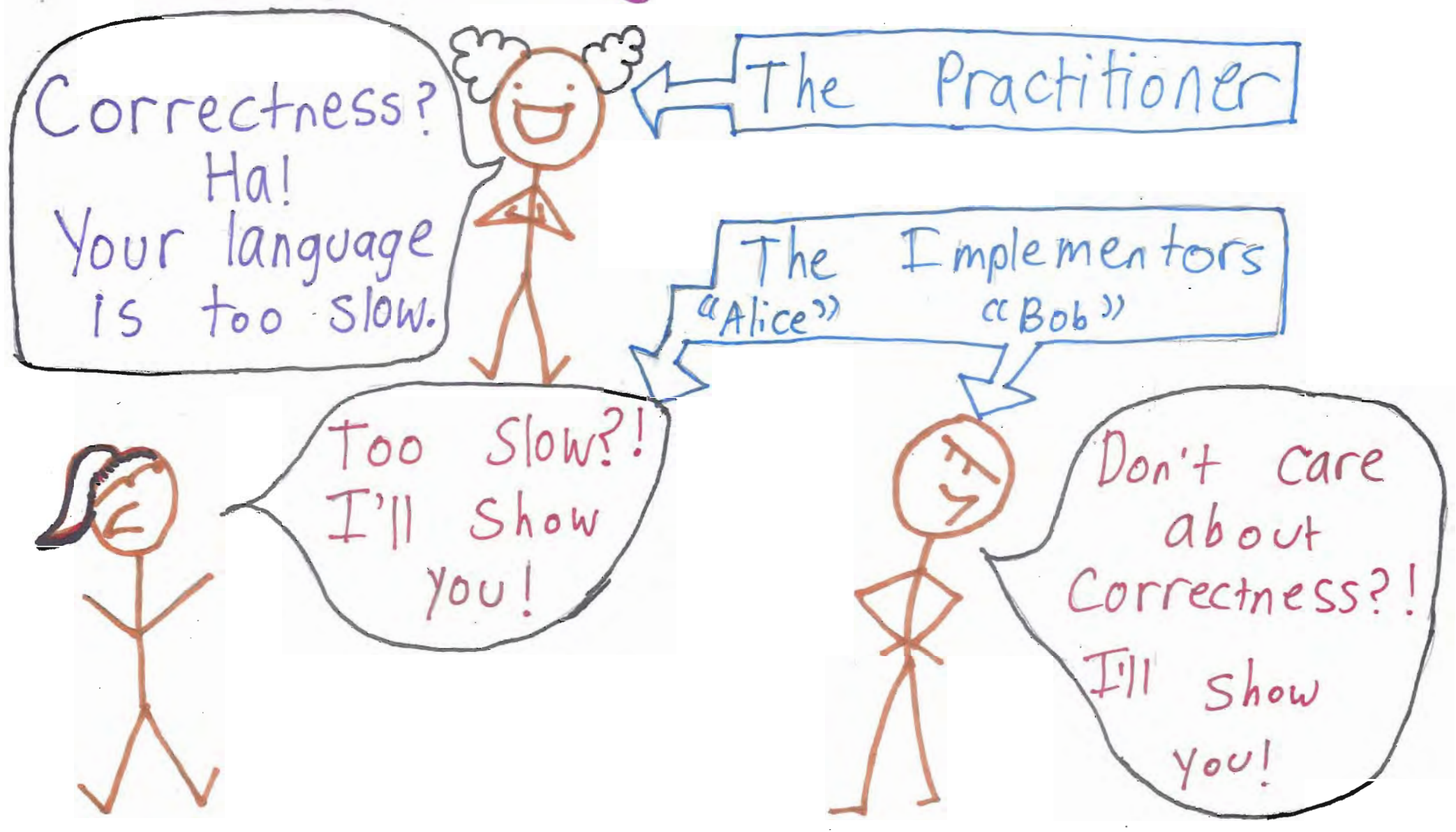
The Dilemma

Strict language	Lazy language
Close to hardware	Abstracted from hardware
Good performance	Bad performance
Difficult reasoning	Easy reasoning

Can we have the best of both worlds?

Sort of ...

Defending Laziness



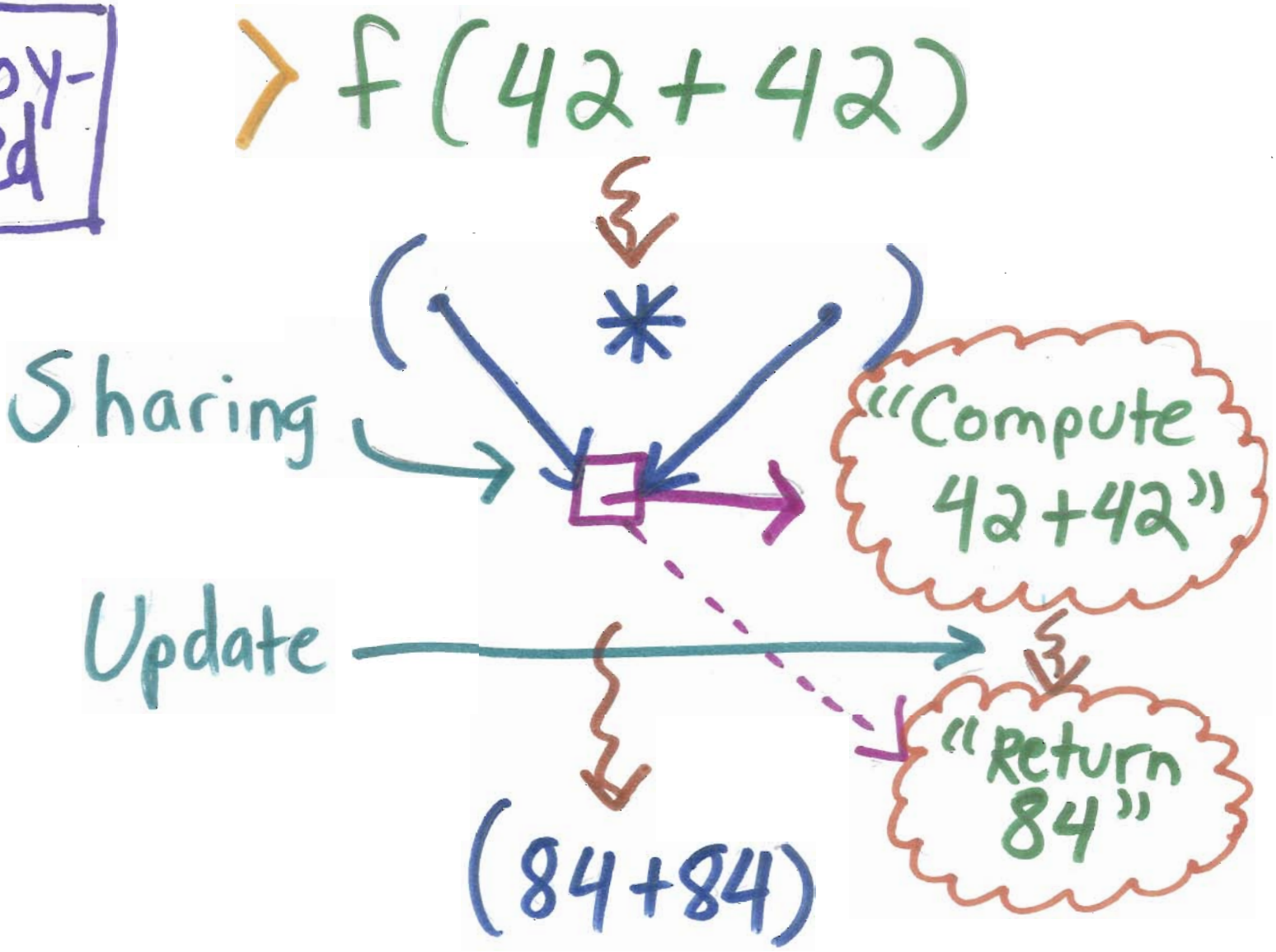
The Trouble With Laziness

```
f x = x*x

> f (42 + 42)

  (42 + 42) * (42 + 42)
```

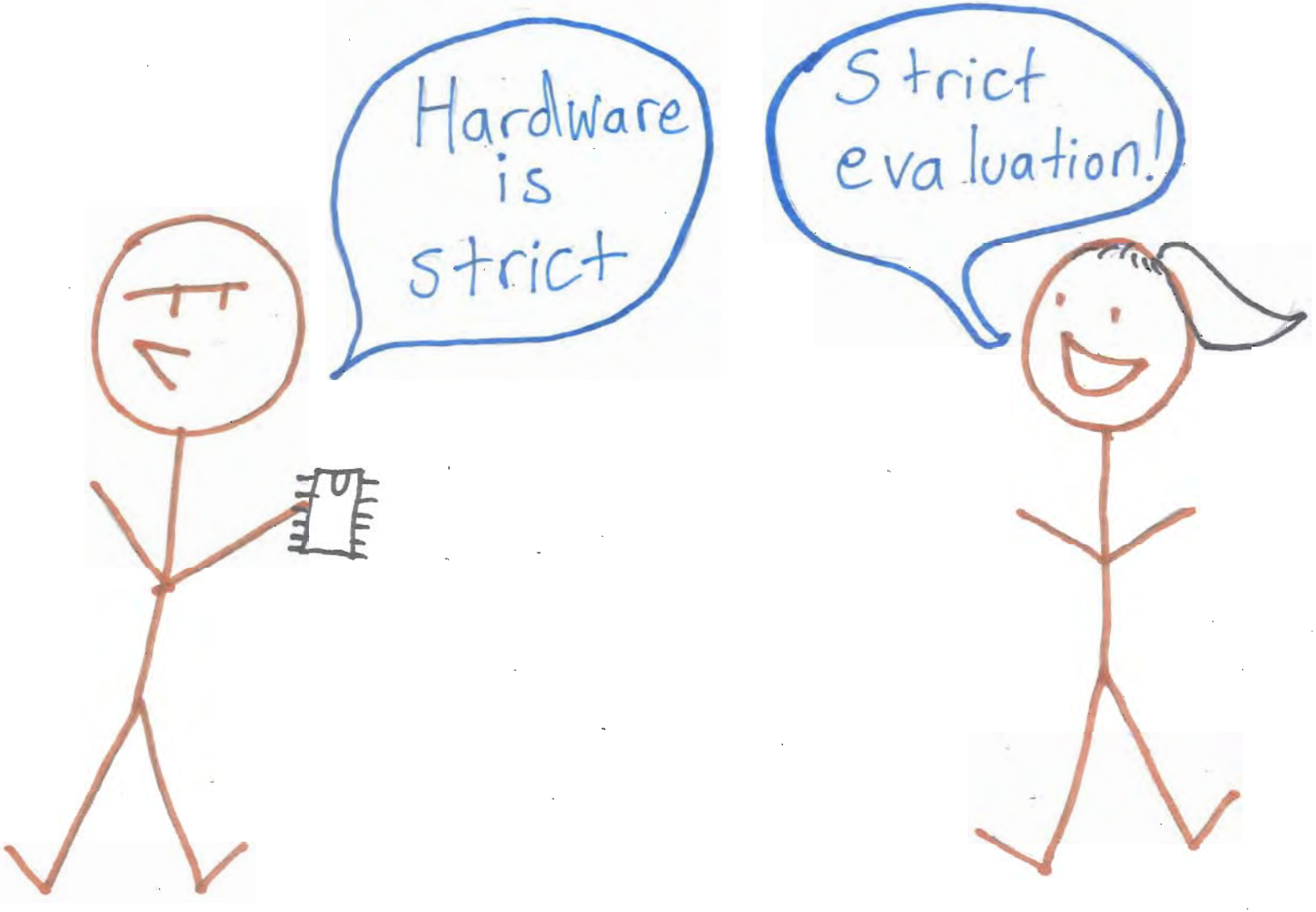
Repeated computation

A Solution and a New Problem

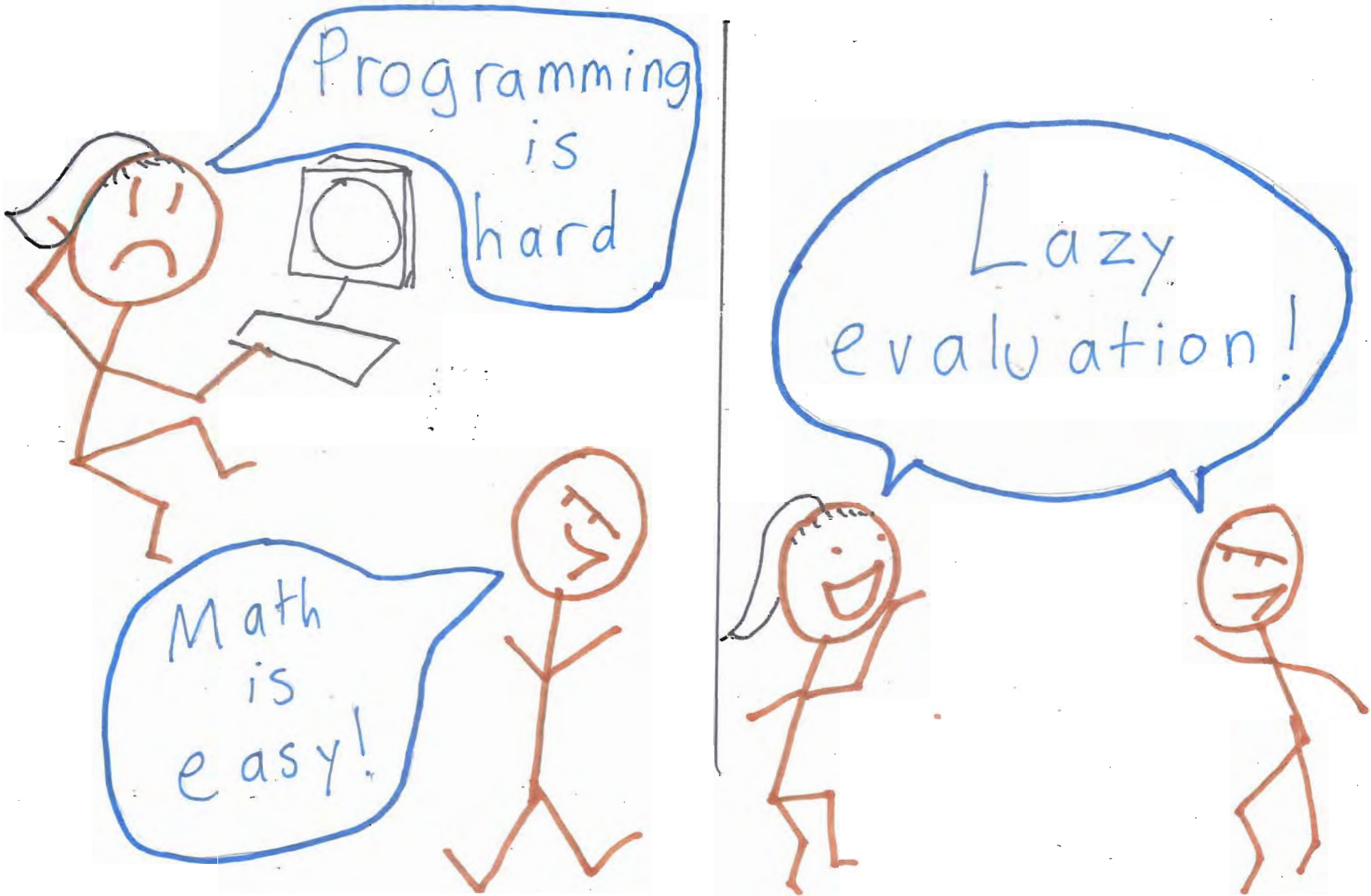
Call-by-need



In Short...

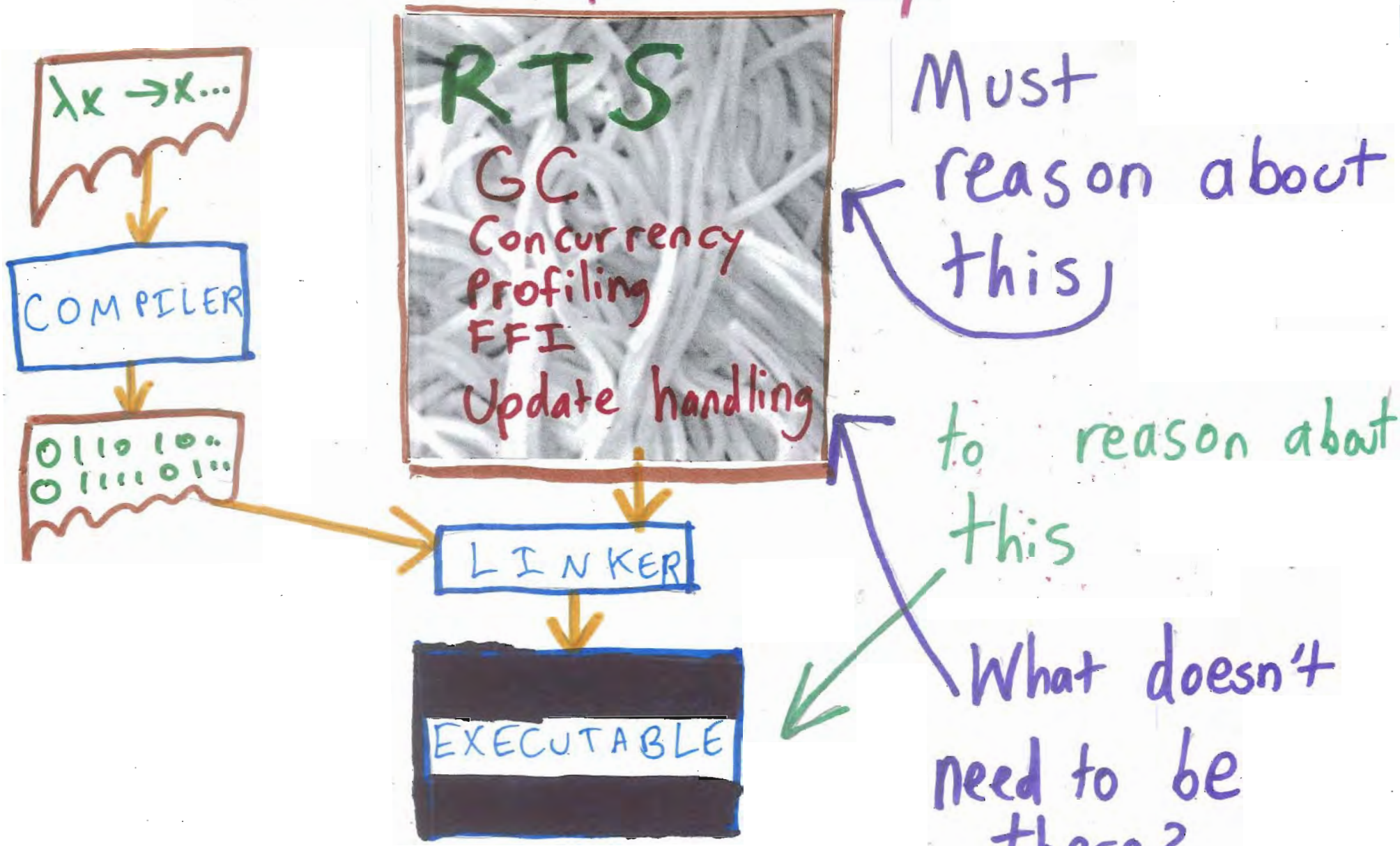


Time Passes



More Problems

Simplicity


λx →x...
COMPILER
0110 10..
0 1111 01..
RTS
GC
Concurrency
Profiling
FFI
Update handling
LINKER
EXECUTABLE
Must reason about this
to reason about this
What doesn't need to be there?

Laziness by Transformation

- Goal:
  - Exploit existing compilers for strict languages
    - Code may be hard to reason about, but compiler isn't!
- Existing Work:
  - Aimed at performance
- Another possible goal:
  - Simplicity

Writing Lazy Code in a Strict Language

á la Scheme:

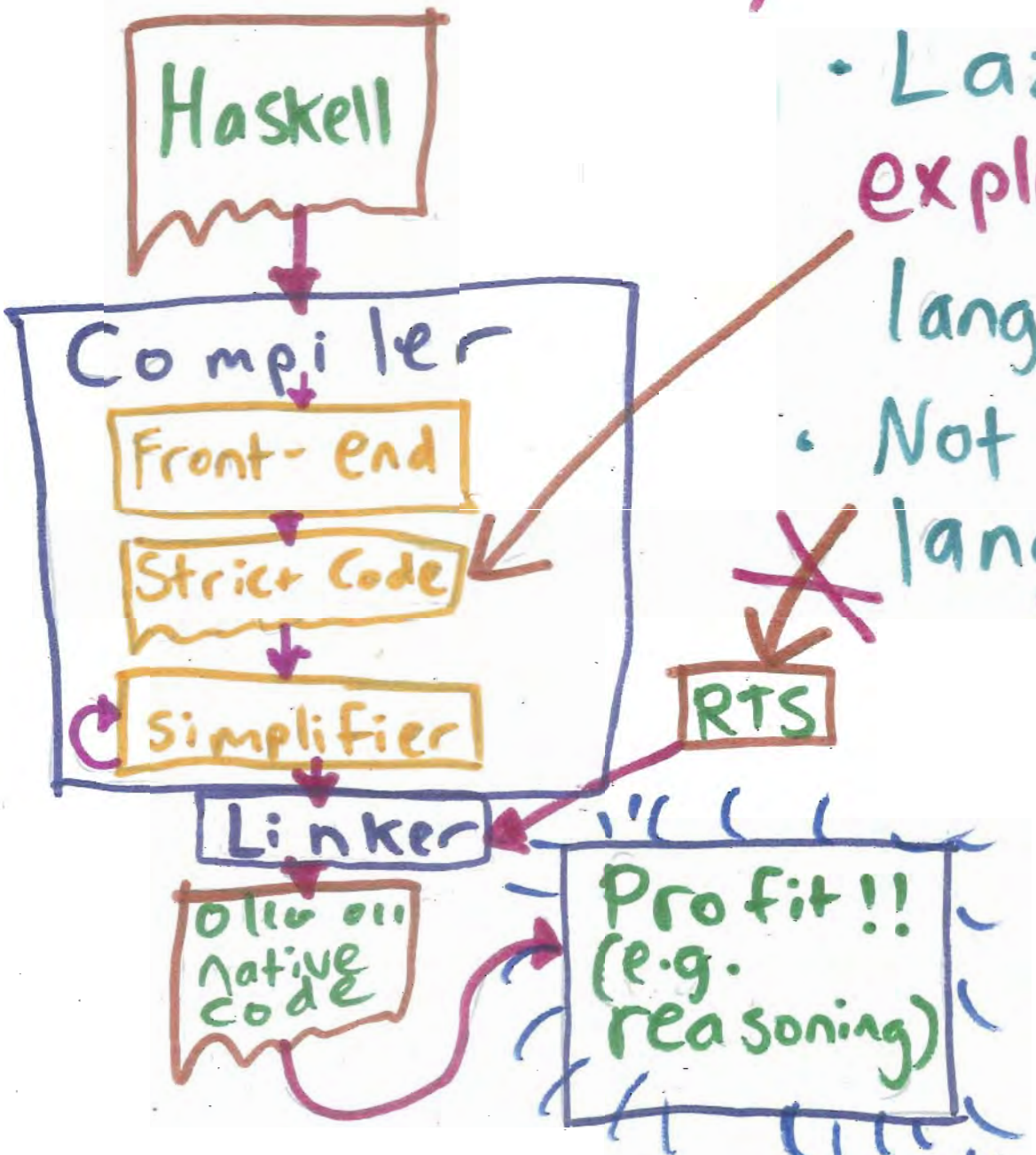
(delay e) ⇝ (λ () → e)

(force e) ⇝ (e ())

But it's not so simple:

- This is call-by-name
- Getting call-by-need is harder.

Compiling By Transformation



- Laziness explicit in intermediate language
- Not implicit in target language

Conclusions

- Laziness is nice
  (Makes reasoning about programs easier)
- Laziness is not so nice
  (Makes reasoning about RTS harder)
- Combine strict language compilers with program transformation
  (And get the best of both worlds)

Thanks to:

Tom Harke

Jim Hook

Rebekah Leslie

Andrew McCreight

Andrew Tolmach

(none of whom endorse the format of this talk)

Questions?