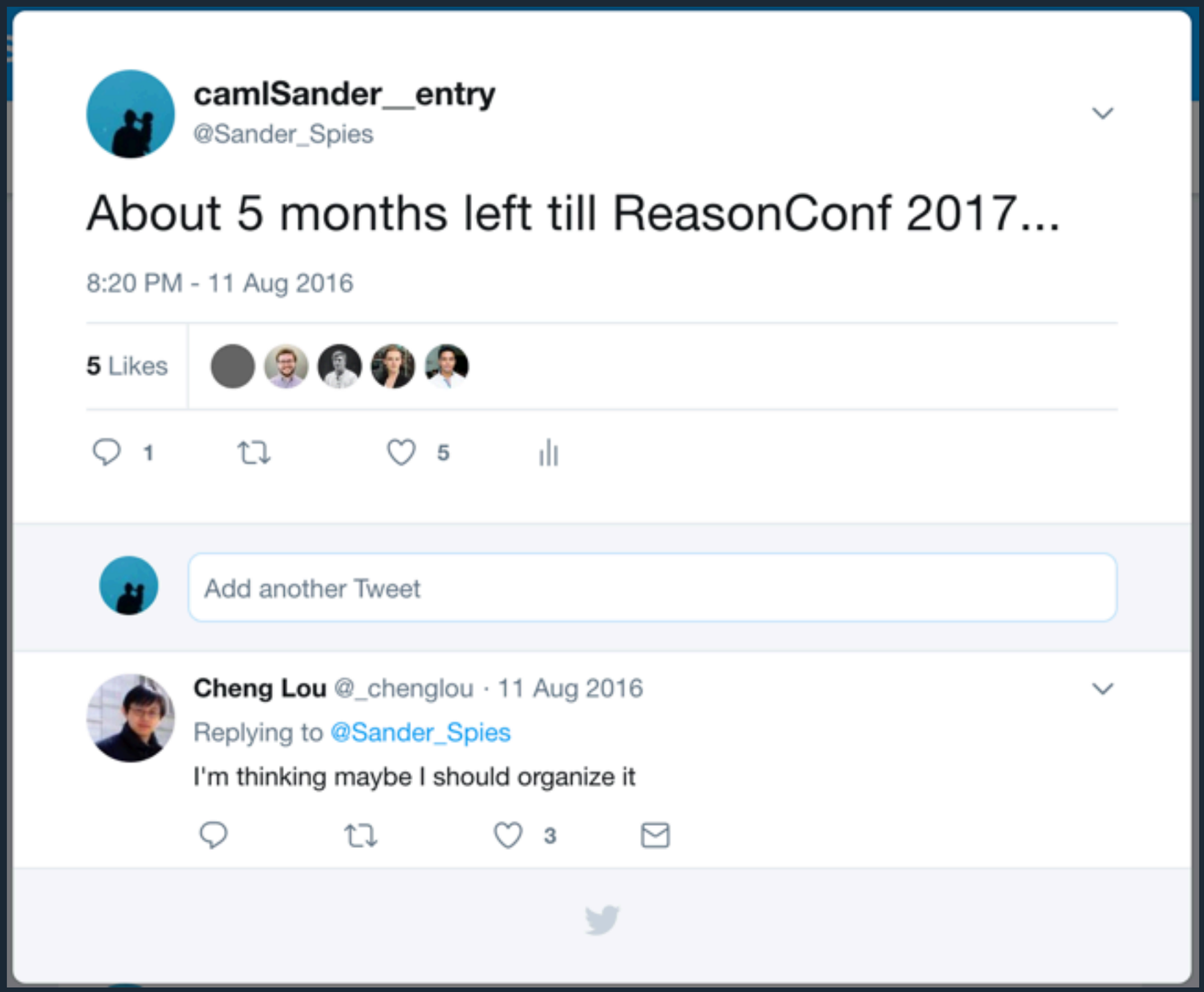
Down the WebAssembly Rabbit Hole

@Sander_Spies


camlSander__entry
@Sander_Spies
About 5 months left till ReasonConf 2017...
8:20 PM - 11 Aug 2016
5 Likes
1
5
Add another Tweet
Cheng Lou @_chenglou · 11 Aug 2016
Replying to @Sander_Spies
I'm thinking maybe I should organize it
3

JavaScript developer

Started with OCaml at the end of 2015

Not an expert

jordwalke
@jordwalke

Following

Happy to announce that we've just open sourced Reason: a new developer experience for the ML language: facebook.github.io/reason/

1:08 PM - 17 May 2016

Our Rabbit Hole:

Compiling OCaml/Reason to WebAssembly

Where to enter the Rabbit Hole?

Compiler Frontend

```
let example = (a, b) => a + b
```

ocamllex: `LET LIDENT<string> EQUAL …`

ocamlyacc / menhir:

```
let_binding:
    LET let_binding_body { (* add to OCaml AST *) }
```

Abstract Syntax Tree: `Pexp_fun`

see parsing/parsetree.mli

.mly format

```
%{
  OCaml code
%}

tokens

%%

productions

%%
```

Productions

let_binding:

| item_attributes

reduce

item_attributes:

| /* empty */

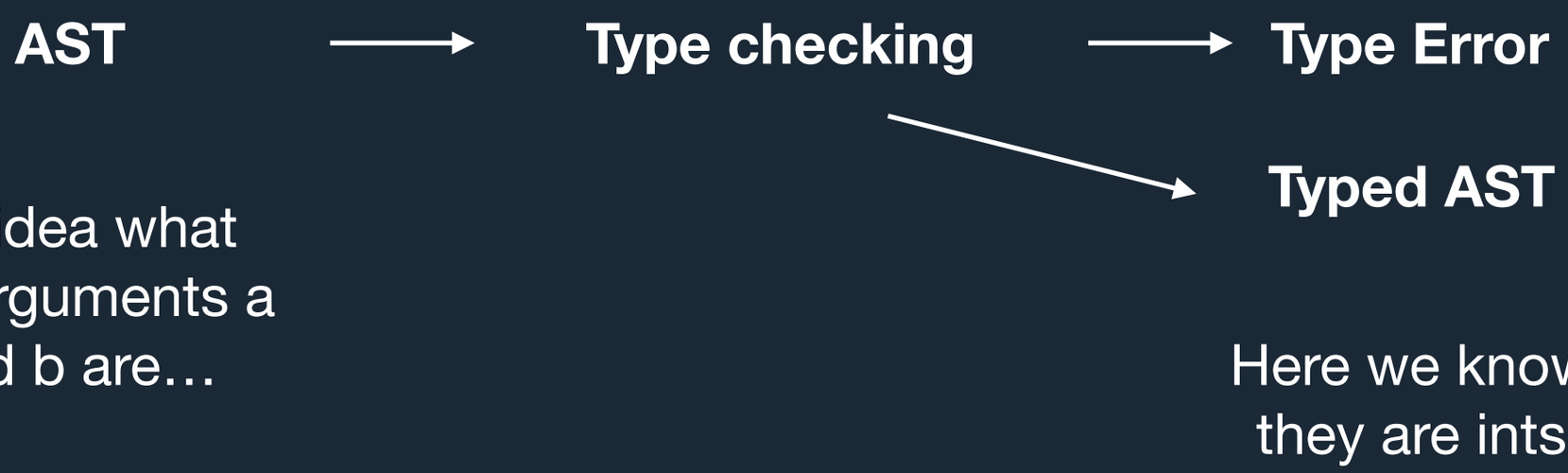
| located_attributes

LET item_extension_sugar? rec_flag let_binding_body

shift

It’s like JSX, but for syntax

(in a really large file)


AST
Type checking
Type Error
Typed AST
No idea what the arguments a and b are…
Here we know they are ints

Compiler Frontend

Compiler Backend

Typed AST → **Lambda IR**

remove higher level
abstractions
(modules, objects, etc.)

replace types with
runtime memory model

Lambda IR

test.ml

```
let example = (a, b) => a + b
```

```
(setglobal Test!

  (let (example/1002 =
          (function a/1003 b/1004 (+ a/1003 b/1004)))

    (makeblock 0 example/1002)
  )
)
```

Bytecode or Native

Lambda IR

transforms to

Bytecode

runs in

ocamlrun

Lambda IR

transforms to

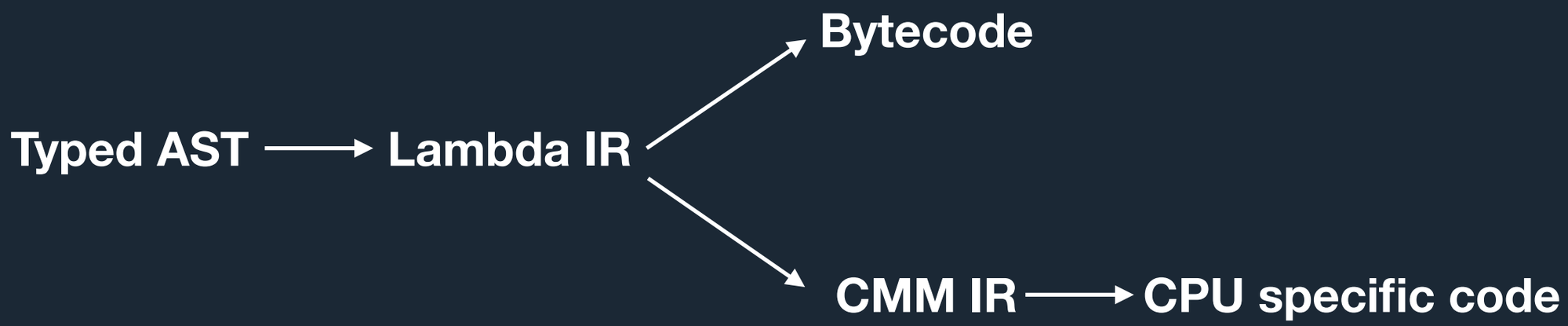
CMM IR

transforms to

Instruction selection
Alloc combining
Register allocation
Linearization
Instruction Scheduling
Native code

Specific to CPU

Compiler Backend



WebAssembly

- Bytecode
- Limited set of instructions
- Security
- Linear memory

Spec implementation is written in OCaml

So how do we get from

```
let example = (a, b) => a + b
```

to

```
(func $example (param $a i32) (param $b i32) (result i32)
  (i32.add
    (get_local $a)
    (get_local $b)
  )
)
```

Possible entry points

Code

AST

Typed AST

Lambda

Bytecode **CMM**

CMM

CMM assumptions

Garbage Collection

Exception Handling

Tail Calls

Stack

Function Example

Function Foo

Function Bar

Function Example Stack Frame

local variables

Memory in use

Heap

“Linear memory is disjoint from code space, the execution stack, and the engine’s data structures; therefore compiled programs cannot corrupt their execution environment“

-

Bringing the Web up to Speed with WebAssembly

Garbage Collection

Stack

Function Example

Function Foo

Function Bar

Need to understand what is in use

Exception handling

Stack

Function Example

Function Foo

Function Bar

Jump in the stack

Tail calls

Stack

Function Example

Function Foo

Function Bar

Reuse the last stack frame

Challenges of compiling OCaml to WASM

No access to the stack

Needed constructs are not available yet

but coming!

The Rabbit Hole

I had no idea what I was getting myself into

Initially it was just me asking questions

...

And more questions

...

And more questions

More people are getting involved

Mozilla

JaneStreet

OCaml Labs

Facebook

Current status

Compile CMM to WASM

Link object files together

Can compile very basic OCaml applications

TODO

Clean up current code

Testsuite

OCaml runtime (GC, memory etc.)

Get it upstream

...

No idea
where the
Rabbit Hole
ends

@Sander_Spies

Thank you!