Compiling OCaml to C, and observing values with `liballocs`

Cheng Sun

CST Part II Project

14th February 2017

What's the problem?

```
(* rev : 'a list -> 'a list *)
let rev xs =
    let rec loop accum = function
        | [] -> accum
        | x::xs -> loop (x::accum) xs
    in
    loop [] xs


let _ = rev [1;2;3]
```

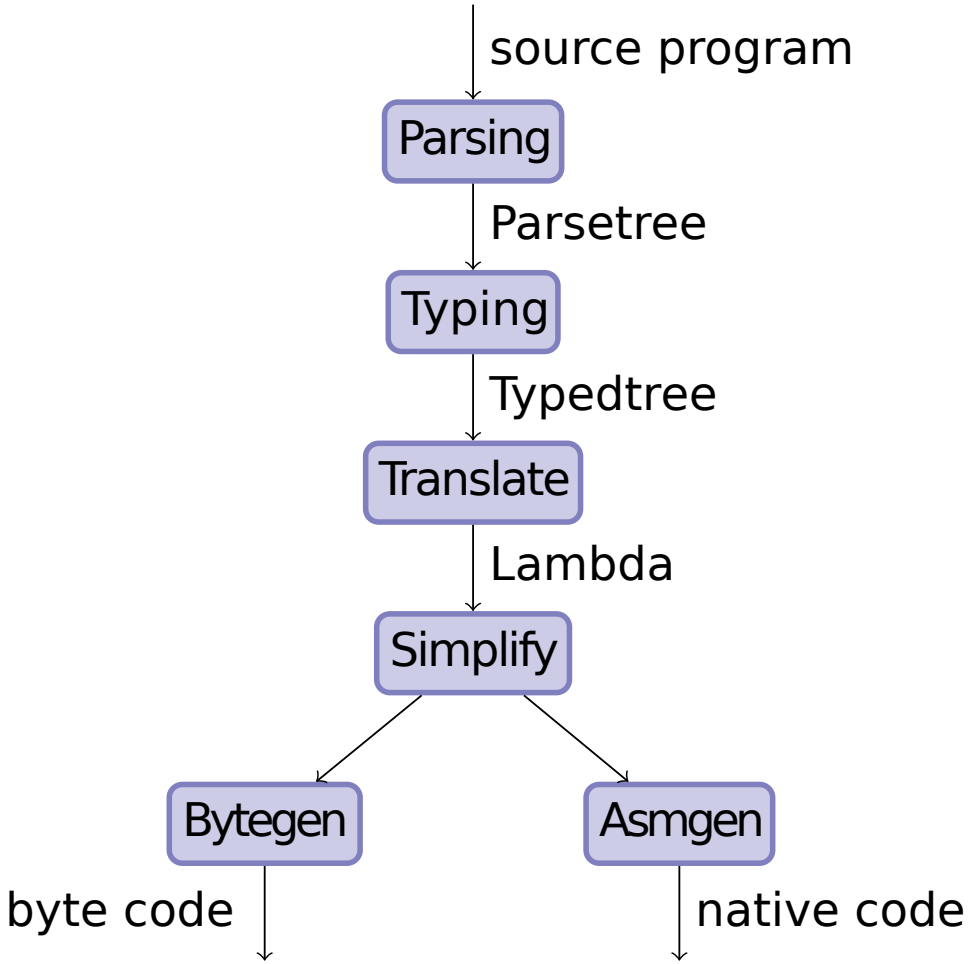
ocamldebug

```
(ocd) break @ Test 5
Loading program... done.
Breakpoint 1 at 6188: file test.ml, line 3,
   characters 18-94
(ocd) run
Time: 14 - pc: 6220 - module Test
Breakpoint: 1
5          | x::xs -> <|b|>loop (x::accum) xs
(ocd) print x
x: 'a = <poly>
```

Using gdb to debug OCaml programs

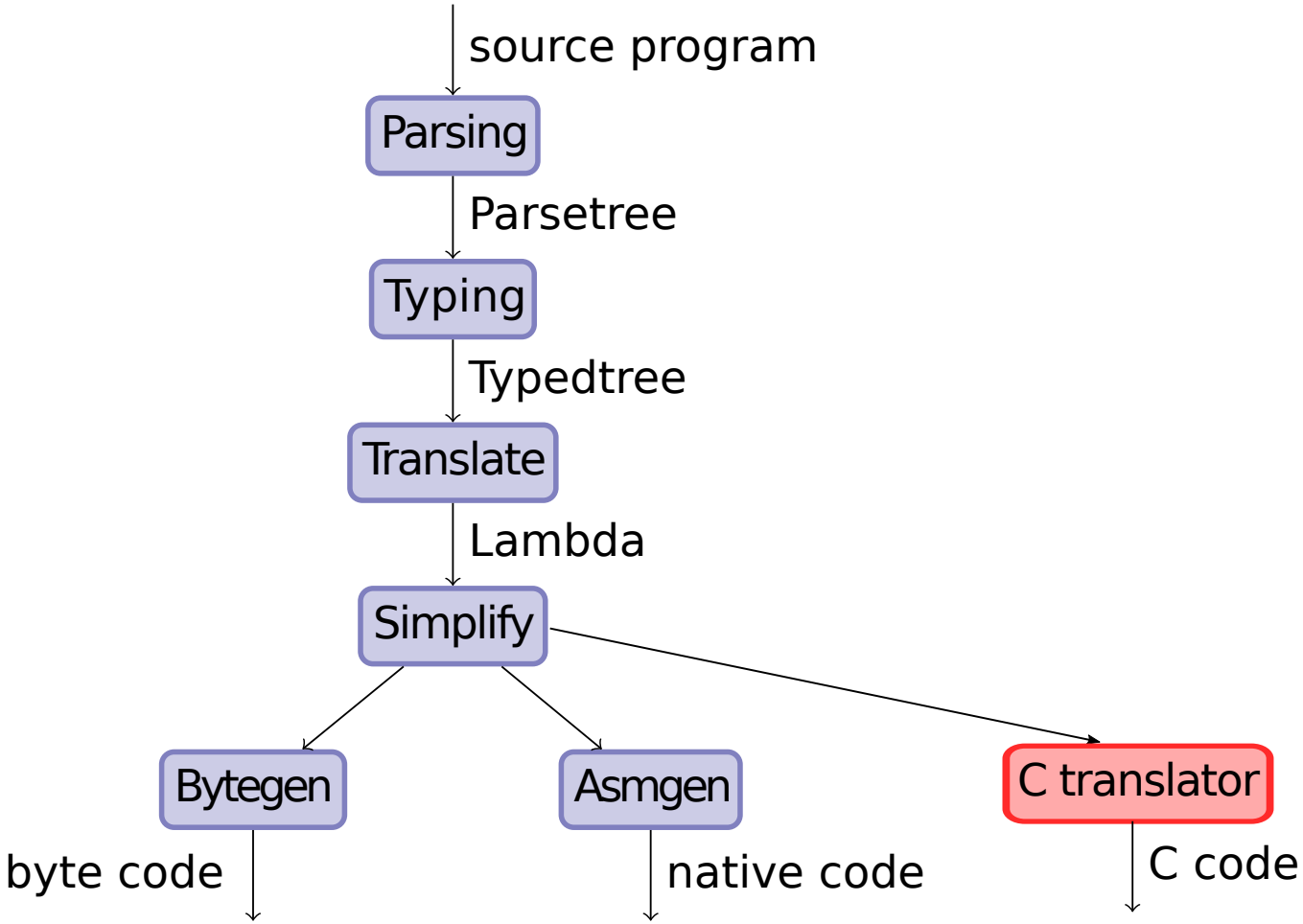
```
(gdb) break test.c:20
Breakpoint 1 at 0x40070d: file ./test.c, line 20.
(gdb) run
Starting program: /home/debian/ocaml/main

Breakpoint 1, loop_1201 (accum_1202=0x0, param_1244
   =0x601030) at ./test.c:20
20      intptr_t* x_1203 = param_1244[0];
(gdb) next
21      intptr_t* __makeblock_1255 = malloc(sizeof(
   intptr_t)*2);
(gdb) call ocaml_liballocs_show(x_1203)
1
```

My approach



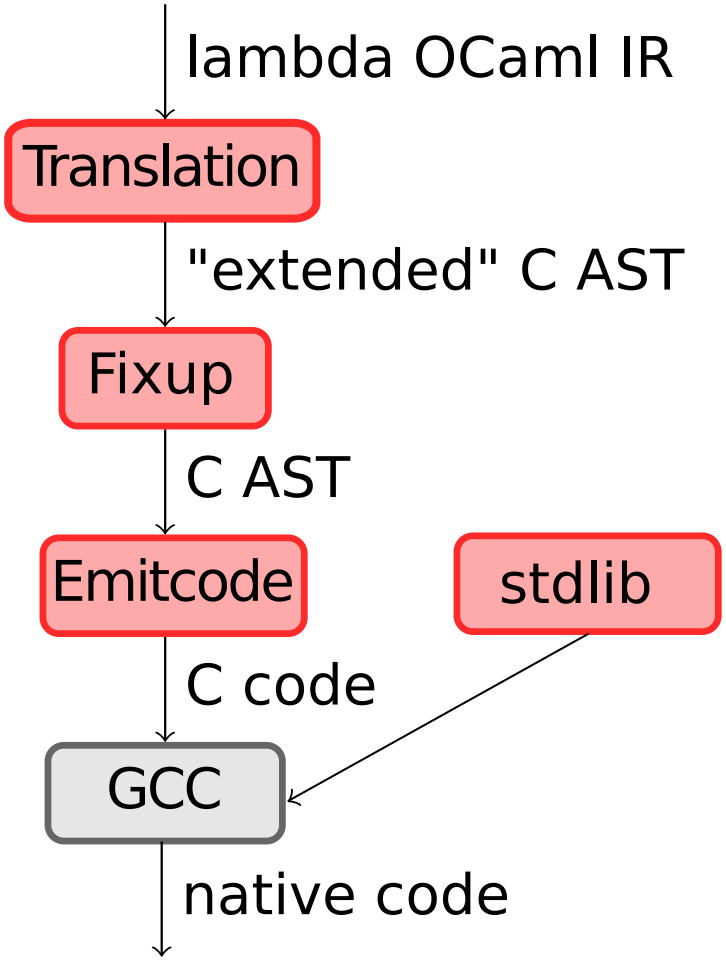
(adapted from Fischbach 2011)

My approach



(adapted from Fischbach 2011)

My approach


lambda OCaml IR
Translation
"extended" C AST
Fixup
C AST
Emitcode
stdlib
C code
GCC
native code

What's done so far?

Enough to get some self-contained OCaml programs running.

1. Basic types
2. Tuples, lists, records, references
3. Polymorphic functions
4. Cross-module interfacing
5. Prototype standard library (`List`, `Printf`)
6. Test suite of self-contained OCaml programs

What's next?

1. Closures
2. Better types representation with `liballocs`
3. Better debugging experience
4. Evaluation

A snippet of C output

```
intptr_t* loop_1201(intptr_t* accum_1202, intptr_t* param_1244){
intptr_t* __deinlined_1256;
if (param_1244) {
intptr_t* xs_1204 = param_1244[1];
intptr_t* x_1203 = param_1244[0];
intptr_t* __makeblock_1255 = malloc(sizeof(intptr_t)*2);
__makeblock_1255[0] = x_1203;
__makeblock_1255[1] = accum_1202;
__deinlined_1256 = ((intptr_t*(*)(intptr_t*,intptr_t*))loop_1201)(
    __makeblock_1255, xs_1204);
} else {
__deinlined_1256 = accum_1202;
}
return __deinlined_1256;
}

intptr_t* rev_1199(intptr_t* xs_1200){
loop_1201;
return ((intptr_t*(*)(intptr_t*,intptr_t*))loop_1201)(((void*)0),
    xs_1200);
}
```