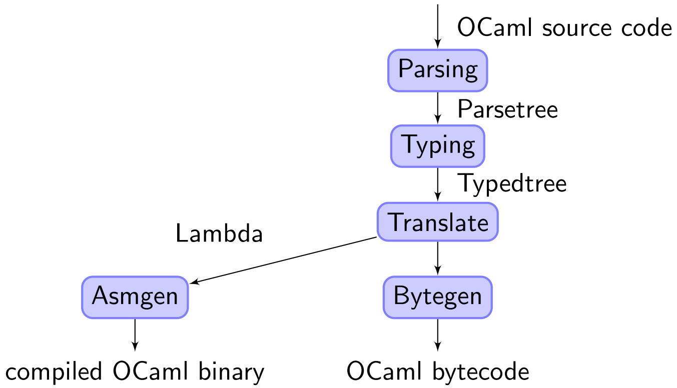
An Observable OCaml

Compiling OCaml into C

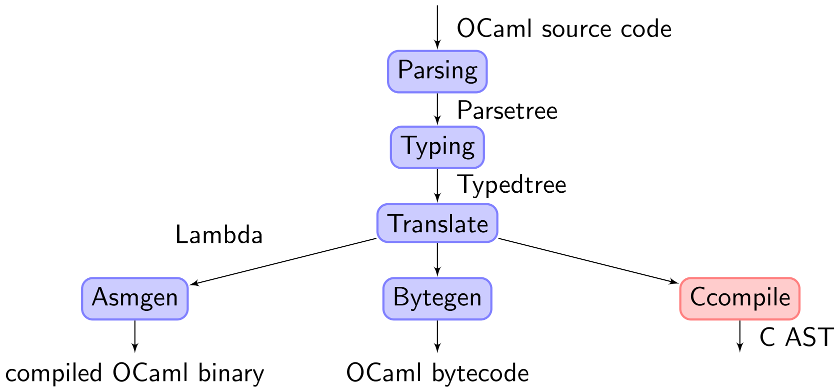
Tianlin Zhang

5th February, 2018

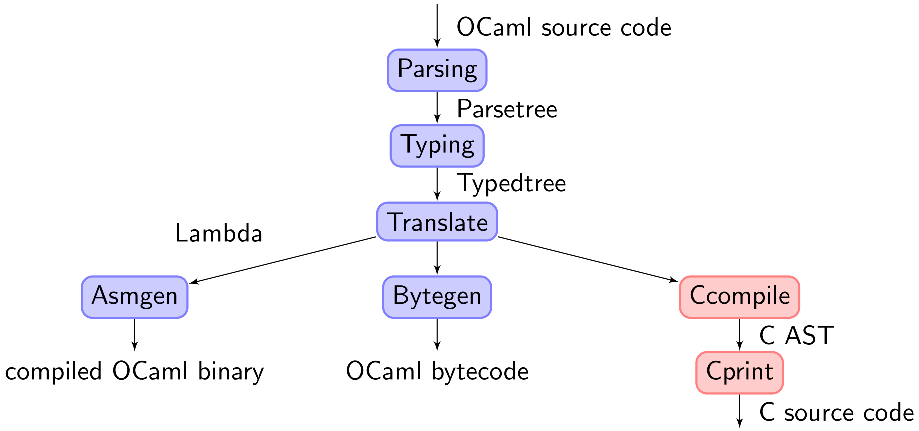
Structure of the project



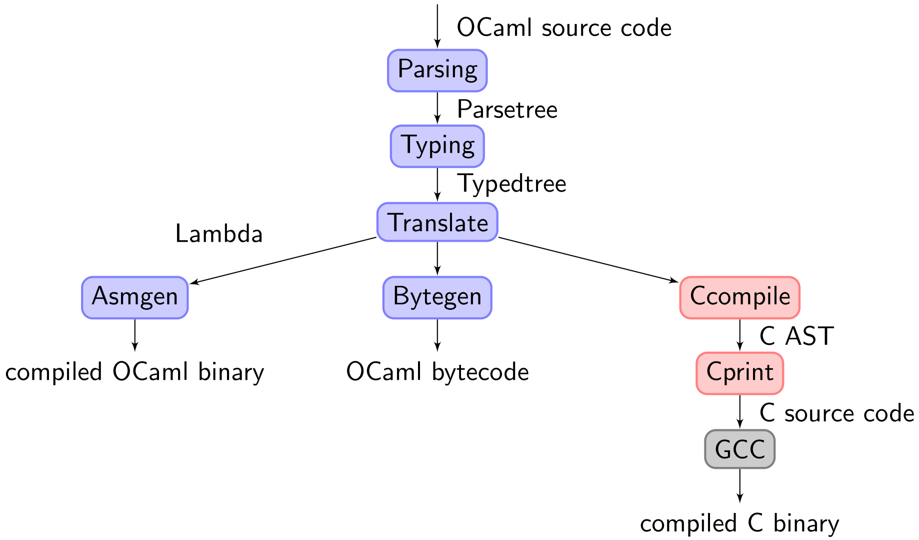
Structure of the project



Structure of the project



Structure of the project



Completed work so far

Core project (a subset of OCaml containing the most commonly used features) is complete, including:

Completed work so far

Core project (a subset of OCaml containing the most commonly used features) is complete, including:

- obtaining types from the Lambda IR
- writing a representation of a subset of C in OCaml

Completed work so far

Core project (a subset of OCaml containing the most commonly used features) is complete, including:

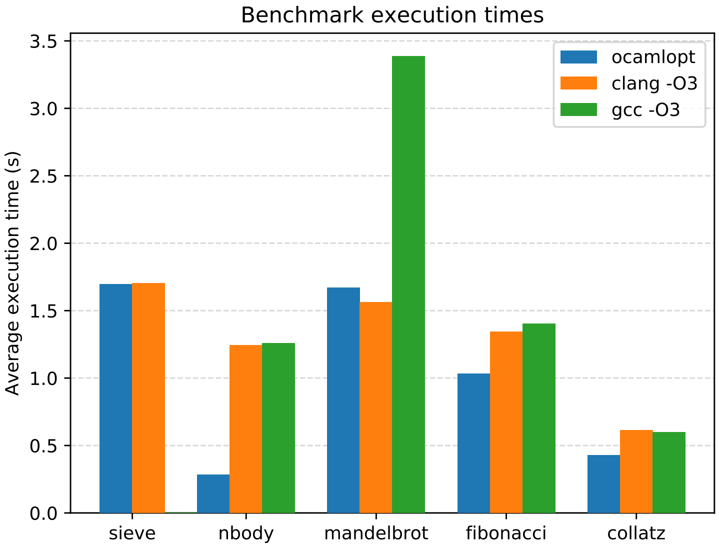
- obtaining types from the Lambda IR
- writing a representation of a subset of C in OCaml
- translation of commonly used language constructs. such as `let`, `let rec`, `if`, `for`, `while`, `ref`, pattern matching etc.
- design of algebraic data structure representation and implementation
- design of closure representation, closure conversion, and function and data structure polymorphism

Completed work so far

Core project (a subset of OCaml containing the most commonly used features) is complete, including:

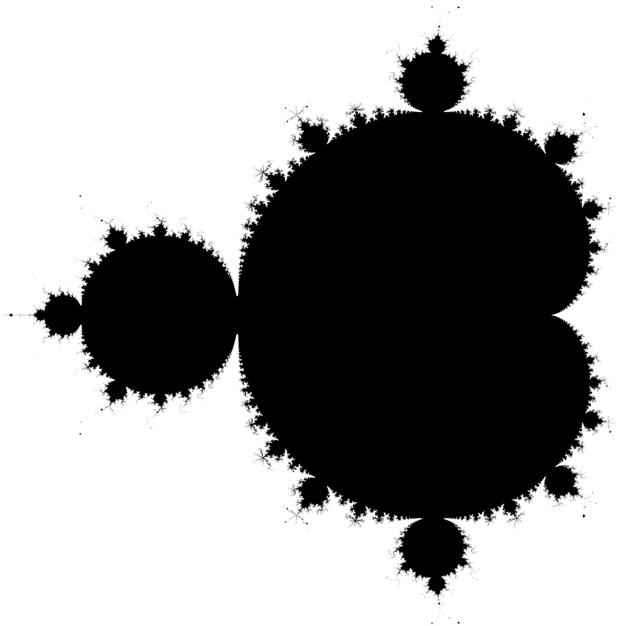
- obtaining types from the Lambda IR
- writing a representation of a subset of C in OCaml
- translation of commonly used language constructs. such as `let`, `let rec`, `if`, `for`, `while`, `ref`, pattern matching etc.
- design of algebraic data structure representation and implementation
- design of closure representation, closure conversion, and function and data structure polymorphism
- library of toy OCaml programs for testing and benchmarking
- some evaluation into performance and observability of compiled output

Benchmarks

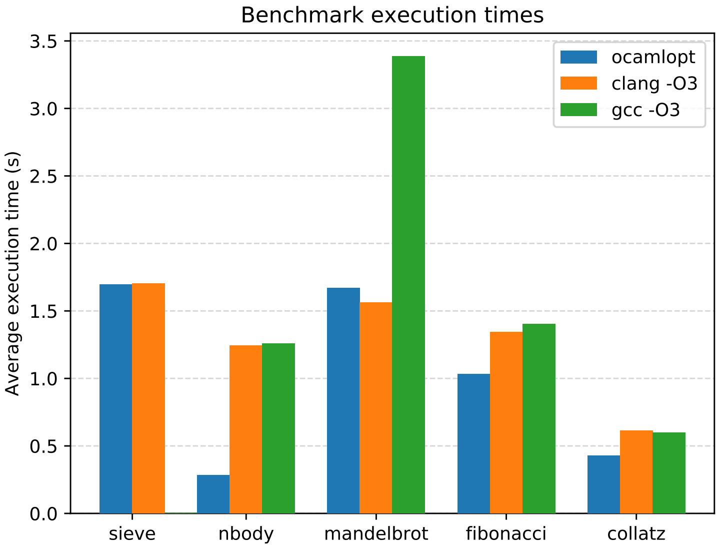


Benchmark execution times
Average execution time (s)
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
ocamlopt
clang -O3
gcc -O3
sieve
nbody
mandelbrot
fibonacci
collatz

Benchmarks



Benchmarks


Benchmark execution times
Average execution time (s)
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
ocamlopt
clang -O3
gcc -O3
sieve
nbody
mandelbrot
fibonacci
collatz

Observability Demo

```
(gdb)
```

Observability Demo

```
(gdb) l
```

Observability Demo

```
(gdb) l
1       let sum xs =
2         let rec go acc = function
3           | x :: xs -> go (acc + x) xs
4           | [] -> acc
5         in go 0 xs
6
7       let a = sum [3]
(gdb)
```

Observability Demo

```
(gdb) l
1       let sum xs =
2         let rec go acc = function
3           | x :: xs -> go (acc + x) xs
4           | [] -> acc
5         in go 0 xs
6
7       let a = sum [3]
(gdb) tb 2
```

Observability Demo

```
(gdb) l
1       let sum xs =
2         let rec go acc = function
3           | x :: xs -> go (acc + x) xs
4           | [] -> acc
5         in go 0 xs
6
7       let a = sum [3]
(gdb) tb 2
Temporary breakpoint 1 at 0x400690: tests/observability/sum.ml:2. (2 locations
(gdb)
```

Observability Demo

```
(gdb) l
1       let sum xs =
2         let rec go acc = function
3           | x :: xs -> go (acc + x) xs
4           | [] -> acc
5         in go 0 xs
6
7       let a = sum [3]
(gdb) tb 2
Temporary breakpoint 1 at 0x400690: tests/observability/sum.ml:2. (2 locations
(gdb) r
```

Observability Demo

```
(gdb) l
1       let sum xs =
2         let rec go acc = function
3           | x :: xs -> go (acc + x) xs
4           | [] -> acc
5         in go 0 xs
6
7       let a = sum [3]
(gdb) tb 2
Temporary breakpoint 1 at 0x400690: tests/observability/sum.ml:2. (2 locations
(gdb) r
Starting program: tests/observability/sum

Temporary breakpoint 1, func_1233 (xs_1205=...) at tests/observability/sum.ml:
2         let rec go acc = function
(gdb)
```

Observability Demo

```
(gdb) l
1       let sum xs =
2         let rec go acc = function
3           | x :: xs -> go (acc + x) xs
4           | [] -> acc
5         in go 0 xs
6
7       let a = sum [3]
(gdb) tb 2
Temporary breakpoint 1 at 0x400690: tests/observability/sum.ml:2. (2 locations
(gdb) r
Starting program: tests/observability/sum

Temporary breakpoint 1, func_1233 (xs_1205=...) at tests/observability/sum.ml:
2         let rec go acc = function
(gdb) s
```

Observability Demo

```
(gdb) l
1       let sum xs =
2         let rec go acc = function
3           | x :: xs -> go (acc + x) xs
4           | [] -> acc
5         in go 0 xs
6
7       let a = sum [3]
(gdb) tb 2
Temporary breakpoint 1 at 0x400690: tests/observability/sum.ml:2. (2 locations
(gdb) r
Starting program: tests/observability/sum

Temporary breakpoint 1, func_1233 (xs_1205=...) at tests/observability/sum.ml:
2         let rec go acc = function
(gdb) s
5         in go 0 xs
(gdb)
```

Observability Demo

```
(gdb) l
1       let sum xs =
2         let rec go acc = function
3           | x :: xs -> go (acc + x) xs
4           | [] -> acc
5         in go 0 xs
6
7       let a = sum [3]
(gdb) tb 2
Temporary breakpoint 1 at 0x400690: tests/observability/sum.ml:2. (2 locations
(gdb) r
Starting program: tests/observability/sum

Temporary breakpoint 1, func_1233 (xs_1205=...) at tests/observability/sum.ml:
2         let rec go acc = function
(gdb) s
5         in go 0 xs
(gdb)
local_func_1215 (acc_1207=0, param_1210=..., closure_obj_1214=0x602050) at tes
2         let rec go acc = function
```

Observability Demo

```
(gdb)
```

Observability Demo

```
(gdb)
3             | x :: xs -> go (acc + x) xs
(gdb)
```

Observability Demo

```
(gdb)
3           | x :: xs -> go (acc + x) xs
(gdb) p x_1208
```

Observability Demo

```
(gdb)
3             | x :: xs -> go (acc + x) xs
(gdb) p x_1208
$1 = 3
(gdb)
```

Observability Demo

```
(gdb)
3           | x :: xs -> go (acc + x) xs
(gdb) p x_1208
$1 = 3
(gdb) p acc_1207
```

Observability Demo

```
(gdb)
3           | x :: xs -> go (acc + x) xs
(gdb) p x_1208
$1 = 3
(gdb) p acc_1207
$2 = 0
(gdb)
```

Observability Demo

```
(gdb)
3           | x :: xs -> go (acc + x) xs
(gdb) p x_1208
$1 = 3
(gdb) p acc_1207
$2 = 0
(gdb) p xs_1209
```

Observability Demo

```
(gdb)
3            | x :: xs -> go (acc + x) xs
(gdb) p x_1208
$1 = 3
(gdb) p acc_1207
$2 = 0
(gdb) p xs_1209
$3 = {i = 1, block = 0x1}
(gdb)
```

Observability Demo

```
(gdb)
3           | x :: xs -> go (acc + x) xs
(gdb) p x_1208
$1 = 3
(gdb) p acc_1207
$2 = 0
(gdb) p xs_1209
$3 = {i = 1, block = 0x1}
(gdb) s
```

Observability Demo

```
(gdb)
3           | x :: xs -> go (acc + x) xs
(gdb) p x_1208
$1 = 3
(gdb) p acc_1207
$2 = 0
(gdb) p xs_1209
$3 = {i = 1, block = 0x1}
(gdb) s
local_func_1215 (acc_1207=3, param_1210=..., closure_obj_1214=0x602050) at tes
2         let rec go acc = function
(gdb)
```

Observability Demo

```
(gdb)
3           | x :: xs -> go (acc + x) xs
(gdb) p x_1208
$1 = 3
(gdb) p acc_1207
$2 = 0
(gdb) p xs_1209
$3 = {i = 1, block = 0x1}
(gdb) s
local_func_1215 (acc_1207=3, param_1210=..., closure_obj_1214=0x602050) at tes
2         let rec go acc = function
(gdb)
4           | [] -> acc
(gdb)
```

Observability Demo

```
(gdb)
3           | x :: xs -> go (acc + x) xs
(gdb) p x_1208
$1 = 3
(gdb) p acc_1207
$2 = 0
(gdb) p xs_1209
$3 = {i = 1, block = 0x1}
(gdb) s
local_func_1215 (acc_1207=3, param_1210=..., closure_obj_1214=0x602050) at tes
2         let rec go acc = function
(gdb)
4           | [] -> acc
(gdb) finish
```

Observability Demo

```
(gdb)
3           | x :: xs -> go (acc + x) xs
(gdb) p x_1208
$1 = 3
(gdb) p acc_1207
$2 = 0
(gdb) p xs_1209
$3 = {i = 1, block = 0x1}
(gdb) s
local_func_1215 (acc_1207=3, param_1210=..., closure_obj_1214=0x602050) at tes
2         let rec go acc = function
(gdb)
4           | [] -> acc
(gdb) finish
Run till exit from #0  local_func_1215 (acc_1207=3, param_1210=..., closure_ob
0x0000000000400740 in local_func_1215 (acc_1207=0, param_1210=..., closure_obj
3           | x :: xs -> go (acc + x) xs
Value returned is $4 = 3
(gdb)
```

Further work

The project is roughly on schedule, within 1–2 weeks of the timetable.

Further work on the project would include:

Further work

The project is roughly on schedule, within 1–2 weeks of the timetable.

Further work on the project would include:

- Using `liballocs` and `gdb` scripts to improve ‘observability’ and user experience while debugging

Further work

The project is roughly on schedule, within 1–2 weeks of the timetable.

Further work on the project would include:

- Using `liballocs` and `gdb` scripts to improve ‘observability’ and user experience while debugging
- Further evaluation tasks into performance and observability

Further work

The project is roughly on schedule, within 1–2 weeks of the timetable.

Further work on the project would include:

- Using `liballocs` and `gdb` scripts to improve 'observability' and user experience while debugging
- Further evaluation tasks into performance and observability

Extension tasks:

Further work

The project is roughly on schedule, within 1–2 weeks of the timetable.

Further work on the project would include:

- Using `liballocs` and `gdb` scripts to improve ‘observability’ and user experience while debugging
- Further evaluation tasks into performance and observability

Extension tasks:

- improving closure creation performance

Further work

The project is roughly on schedule, within 1–2 weeks of the timetable.

Further work on the project would include:

- Using `liballocs` and `gdb` scripts to improve ‘observability’ and user experience while debugging
- Further evaluation tasks into performance and observability

Extension tasks:

- improving closure creation performance
- compilation of modules, functors and exceptions

Further work

The project is roughly on schedule, within 1–2 weeks of the timetable.

Further work on the project would include:

- Using `liballocs` and `gdb` scripts to improve ‘observability’ and user experience while debugging
- Further evaluation tasks into performance and observability

Extension tasks:

- improving closure creation performance
- compilation of modules, functors and exceptions
- compilation of standard library modules

Some example C code

```
#line 2 "tests/observability/sum.ml"
intptr_t local_func_1215(intptr_t acc_1207,value_type param_1210,
    closure_t closure_obj_1214){closure_t go_1206;go_1206=
    TO_CLOSURE(closure_obj_1214[(TO_INT(*(closure_obj_1214))-1)]);
#line 2 "tests/observability/sum.ml"
intptr_t ifelse_return_1216;if(UNBOX_INT(param_1210)){variable_type
     field_access_1217;field_access_1217=UNBOX_BLOCK(param_1210)
    [2];intptr_t let_return_1218;{{value_type xs_1209;xs_1209=
    TO_VALUE(field_access_1217);variable_type field_access_1219;
    field_access_1219=UNBOX_BLOCK(param_1210)[1];intptr_t
    let_return_1220;{{intptr_t x_1208;x_1208=TO_INT(
    field_access_1219);
#line 3 "tests/observability/sum.ml"

#line 3 "tests/observability/sum.ml"
intptr_t binop_result_1221;binop_result_1221=(acc_1207+x_1208);
    intptr_t temp_1222;temp_1222=binop_result_1221;value_type
    temp_1223;temp_1223=xs_1209;intptr_t apply_result_1224;
    apply_result_1224=((intptr_t(*)(intptr_t,value_type,closure_t))
    ((intptr_t(*)(intptr_t,value_type,closure_t))TO_FUNC(go_1206[(
```