Tracing OCaml Programs

Darius Foo, Wei-Ngan Chin

National University of Singapore

OCaml 2022

Debugging in OCaml today

#trace	ocamldebug	printf
✓ Getting an overview	✓ Reverse execution	✓ Control over output ✓ *Accessible*
✗ Too much output ✗ Bytecode toplevel ✗ Needs inputs	✗ Hard to get an overview ✗ Code evaluation	✗ Modifying source ✗ Inserting printers

Can a combined tool mitigate the downsides of each approach?

Type-aware record-and-replay debugging

- *Instrument* program to collect events
  - e.g. function calls and returns, with arguments and return value
- Run program and *record* an execution trace
- *Extract* information from trace

Instrumentation

```
let cons x xs = x :: xs
```

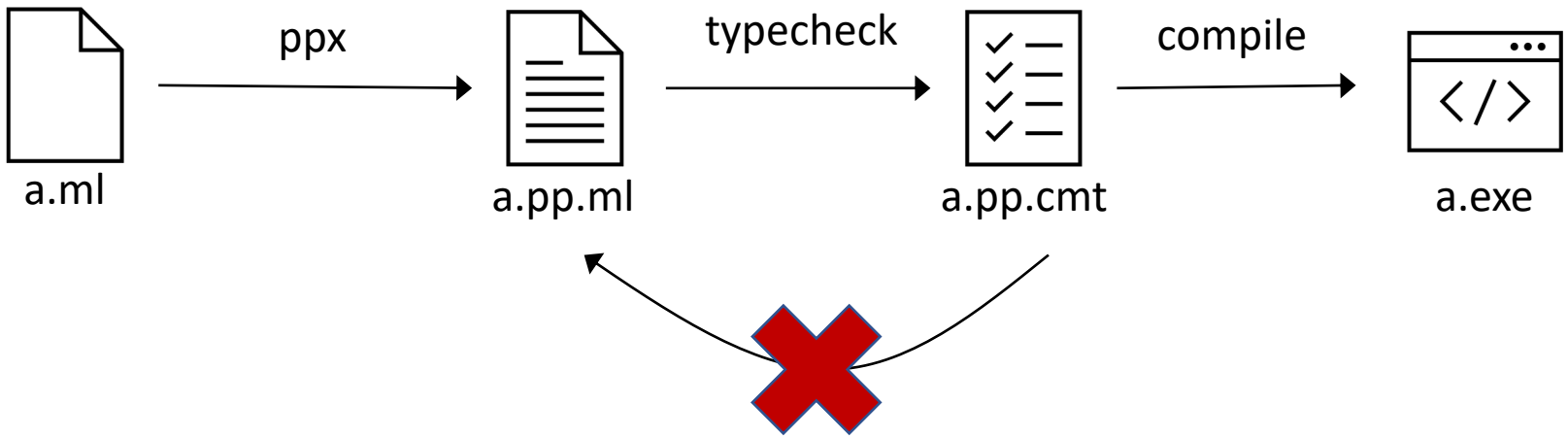
```
let cons x xs =
  record_start "cons" [show x; show xs];
  let res = x :: xs in
  record_end "cons" (show res);
  res
```

Instrumentation

```
let rec fact n =
  if n = 0 then 1 else n * fact (n - 1)
```

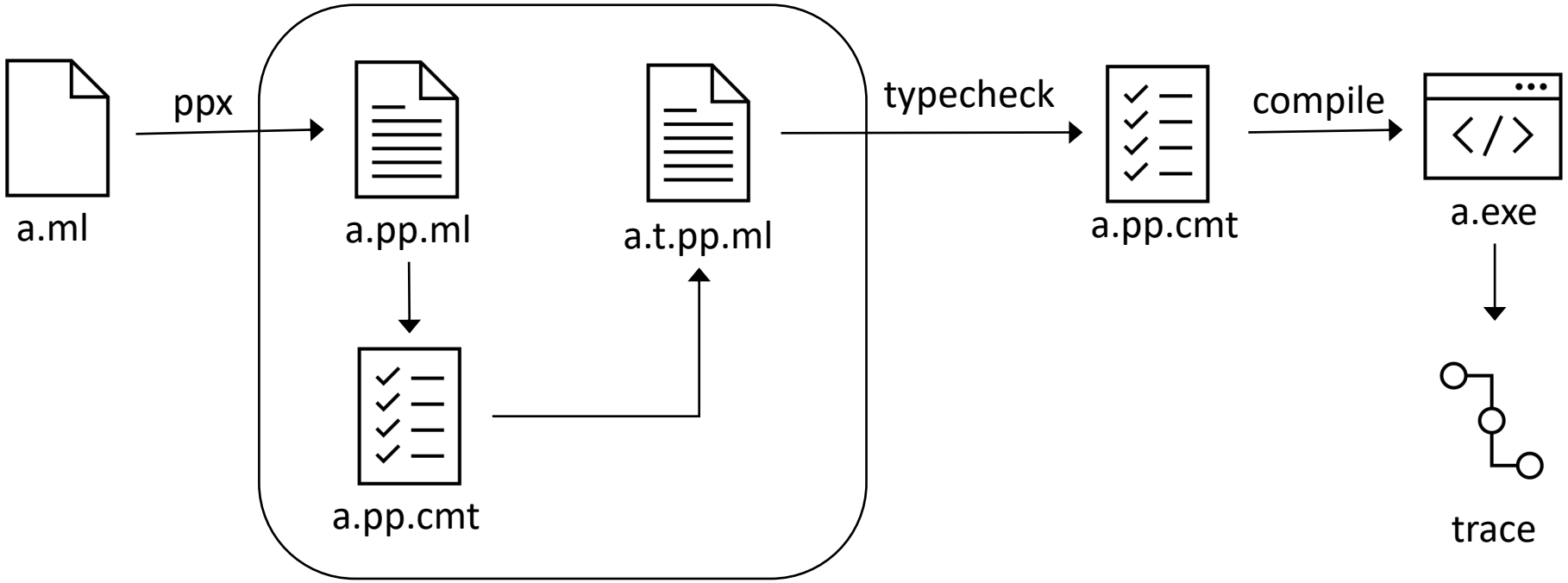
```
let fact n =
  let fact_ self n =
    if n = 0 then 1 else n * self (n – 1)
  in let rec aux n =
    record_start "fact" [show n];
    let res = fact_ aux in
    record_end "fact" (show res);
    res
  in aux n
```

Typical ppx



Typed ppx

(typpx, typedppxlib)



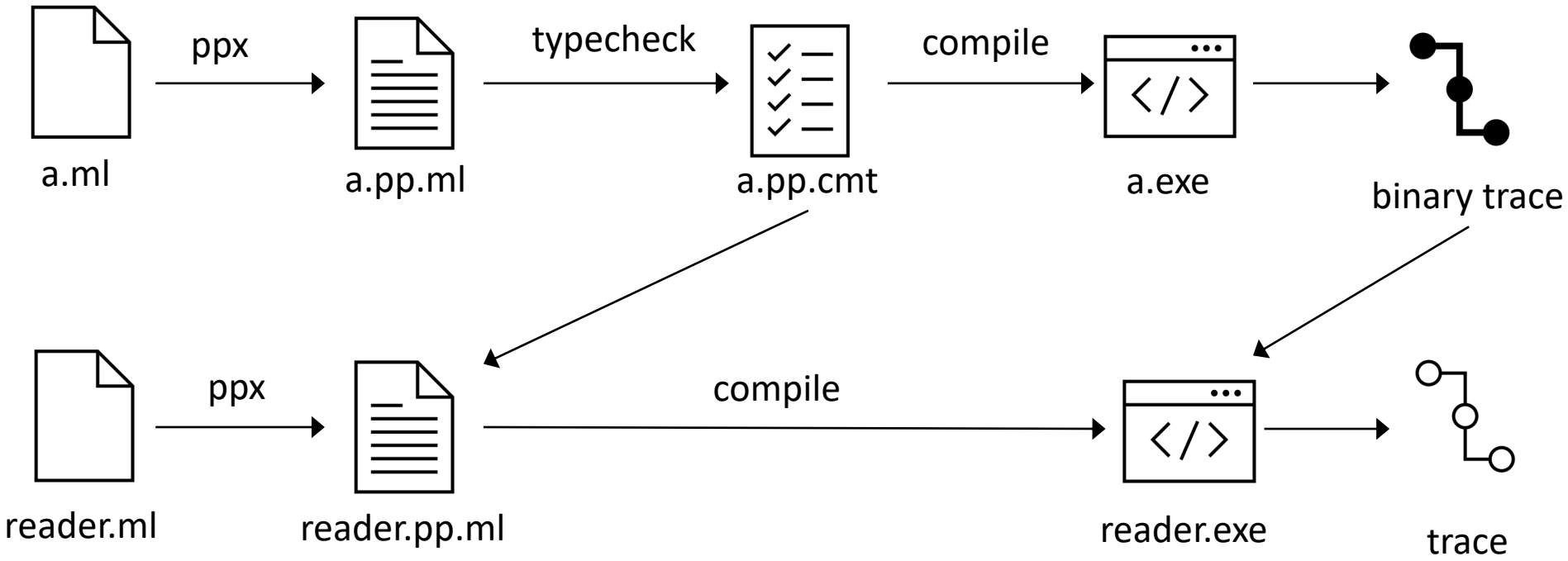
Typed ppx redux

The overhead of typechecking twice is difficult to avoid in the general case.

This is a *specific* case: we can separate the parts of the program which produce recorded values from the parts which consume them.

This allows compilation to be *staged*.

Typed ppx redux



— THE —

Common Trace Format

A FLEXIBLE, HIGH-PERFORMANCE
BINARY TRACE FORMAT

Tradeoffs

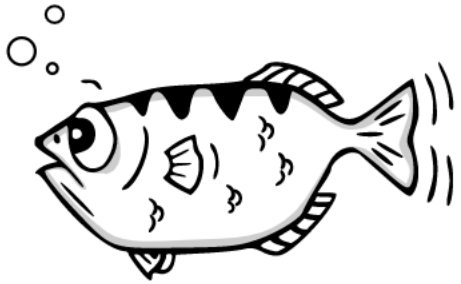
- Makes debugging a build problem instead of a runtime problem
- Code must be recompiled to be instrumented
  - Library code is not instrumented, however we can see its output
- Fragile uses of Typedtree APIs
  - Less than vendoring typechecker
- Staged build a workaround for lack of ad hoc polymorphism?
  - Separating content from schema does lower (runtime, compile) overhead
- Scalability?
  - Lots of configuration for instrumentation process
  - In principle, could be no more expensive than regular printf

Other debugging methods

- Logging, tracing, testing (Runtime Events)
- gdb, rr, dtrace
- ocamli, Furukawa et. al's stepper
- magic-trace

rr

ocamli



- hat

Work in progress

- Try on projects of all kinds and sizes
- Make build integration simpler
- More ways to query traces (backward slicing, evaluate code, …)
- Concurrency

https://github.com/dariusf/ppx_debug

Thank you!