① (fun (f x y) e)   ② (fun (f (x : T_x)(y : T_y)) e)

[x : A][y : A]e : T   (fun (f (x : Bool)(y : Num))

   Billions of   (f 3 4)

   LOC un-annotated

   These don't TC   can only compile with

                            all tc turned off

---

f_notc :   f :   (check + don't compile)   fn compile-me (f_id : int, x : i64, y : i64)

                          mov rd, f_id-fa-f   let (def, lbl) = get_fun(f_id)
   full un-ann func   ... more args ...   let env = calc_tenv(def, x, y)
                          call compile-me   let tdef = tc_def(def, env)
                          jmp rax   let fastf = compile-def(def)
                                                       instrs-to-asm(GLOBAL-OPS, fastf)

                                                                  return ... label f_fast ...

② solution   ☆ Julia
Compile 2 versions of
each annotated function:
one w/tc   one without

When compiling Call, select
based on types

extend to ①.   ⌐challenging
┌─────────────────────┐
│ infer types for x, y │ from e
└─────────────────────┘
then annotate but also do ② solution

─ mut. rec   ─ isnum, isbool
─ input
─ set! w/type change
─ global vars
─ (m   a   b) (need to be conf.
                 in types of a,b
                 at compile-time.)

─ dead code/loops

solution to ①
           un-annotated
the first time* ↑ a function is
called, tc it with the given
types

Just-in Time

if (typeof x === "number") {
    e   [x:Num] e
}
· TypeScript if splitting

Typed Scheme