Late Typing

for Loosely Coupled Recursion

Ravi Chugh

JavaScript

```
var tick = function (n) {
  return n > 0 ? "tick " + tock(n) : "";
};

var tock = function (n) {
  return "tock " + tick(n-1);
};

tick(2);   // "tick tock tick tock "
```

λ -Calc + Refs

```
let (tick,tock) = (ref undef, ref undef)

tick := \n.
  n > 0 ? "tick " + !tock n : ""


tock := \n.
  "tock " + !tick (n-1)


!tick 2   // "tick tock tick tock "
```

λ -Calc + Refs

```
let (tick,tock) = (ref undef, ref undef)

tick := \n.
  n > 0 ? "tick " + !tock n : ""


tock := \n.
  "tock " + !tick (n-1)


!tick 2   // "tick tock tick tock "
```

Simply-Typed λ-Calc + Refs

```
val tick : ref (int -> str)
let tick = ref (\_. "")

val tock : ref (int -> str)
let tock = ref (\_. "")

tick := \n. n > 0 ? "tick " + !tock n : ""

tock := \n. "tock " + !tick (n-1)

!tick 2   // "tick tock tick tock "
```

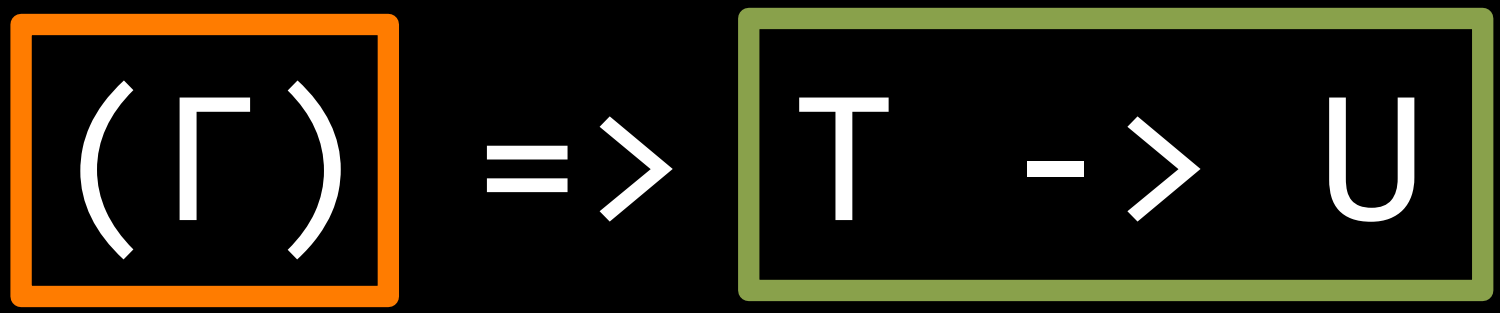
Simply-Typed λ-Calc + Refs

```
val tick : ref (int -> str)
let tick = ref (\_. "")

val tock : ref (int -> str)
let tock = ref (\_. "")

tick := \n. n > 0 ? "tick " + !tock n : ""

tock := \n. "tock " + !tick (n-1)

!tick 2   // "tick tock tick tock "
```

Function Types

```
T -> U
```

"Open" Function Types

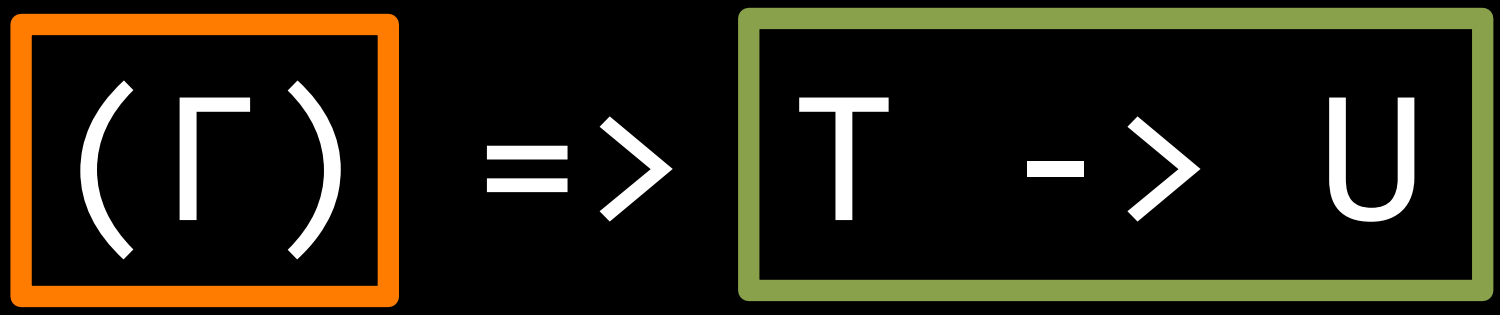


Rely **Guarantee**

```
let tick = ref undefined
let tock = ref undefined

val f_tick : (*tock: int -> str) => int -> str
let f_tick n = n > 0 ? "tick " + !tock n : ""

val f_tock : (*tick: int -> str) => int -> str
let f_tock n = "tock " + !tick (n-1)

tick := f_tick

 // Rely: { *tick: int -> str , *tock: int -> str }
 // Guar: { *tick: int -> str , *tock: undefined  }

!tick 2
  // ill-typed; run-time crash!
```

```
let tick = ref undefined
let tock = ref undefined

val f_tick : (*tock: int -> str) => int -> str
let f_tick n = n > 0 ? "tick " + !tock n : ""

val f_tock : (*tick: int -> str) => int -> str
let f_tock n = "tock " + !tick (n-1)

tick := f_tick
tock := f_tock

 // Rely: { *tick: int -> str , *tock: int -> str }
 // Guar: { *tick: int -> str , *tock: int -> str }

!tick 2
  // well-typed; "tick tock tick tock "
```

"Open" Function Types



Rely **Guarantee**

"Late" Checking at Calls

λ-Calc + Recds

```
let tick this n =
  n > 0 ? "tick " + this.tock this n : ""


let tock this n -
  "tock " + this.tick this (n-1)

let obj = { tick = tick ; tock = tock }




obj.tick obj 2 // "tick tock tick tock "
```

λ-Calc + Recds

```
tick : (ς: {tock: ς->int->str}) => ς -> int -> str
let tick this n =
  n > 0 ? "tick " + this.tock this n : ""

tock : (ς: {tick: ς->int->str}) => ς -> int -> str
let tock this n -
  "tock " + this.tick this (n-1)

let obj = { tick = tick ; tock = tock }

 // Rely: { tick: ς->int->str, tock: ς->int->str }
 // Guar: { tick: ς->int->str, tock: ς->int->str }

obj.tick obj 2 // "tick tock tick tock "
```

Late Typing

for Loosely Coupled Recursion

```
(Γ) => T -> U
```