Minor Research Report 5

25/09/2023

Student: TRAN, Thanh Cong
Supervisor: Nao Hirokawa

I. Overview

- Objectives:
  - Implement polymorphism type inference system
- Progress: Done
- Source code: https://github.com/thanhtcptit/imp-typescript/blob/main/imp_interpreter/typing.ts

II. Implementation

1. Type terms

Term	**Form**
Type Variable	α
Type Arrow	α → α
Type Application	α α
Type Constructor	Int, Bool, Unit

2. Type environment (Γ)

Variable	**Type**
0, 1, 2, …	Int
true, **false**	Bool
+, **-**	Int → Int → Int
>, **>=**, **<**, **<=**	Int → Int → Bool
==, **!=**	α → α → Bool
&&, **||**	Bool → Bool → Bool
!	Bool → Bool
:=	α → α → Unit
if	Bool → Unit → Unit
if-else	Bool → Unit → Unit → Unit
while	Bool → Unit → Unit
;	Unit → α → α
return	α → α

3. Type expressions

Expression (e)	**Form**	**Example**
Variable	x	x
Integer	0, 1, 2, …	1
Boolean	**true**, **false**	**true**
λ-abstraction	λx. e	λx. x
Application	e1 e2	λx. λy. (+) x y
Let	let t = e1 in e2	let t = λx. (+) x 1 in t 1

4. Type inference rules

Expression	**Deduction form**
Variable	$\dfrac{\mathbf{\Gamma}(x) = \alpha}{\mathbf{\Gamma} \vdash x : \alpha}$
Integer	$\dfrac{}{\mathbf{\Gamma} \vdash 1 : Int}$
Boolean	$\dfrac{}{\mathbf{\Gamma} \vdash true : Bool}$
λ-abstraction	$\dfrac{\mathbf{\Gamma} \cup \{x : \alpha\} \vdash e : \beta}{\mathbf{\Gamma} \vdash \lambda x.\, e : \alpha \rightarrow \beta}$
Application	$\dfrac{\mathbf{\Gamma} \vdash e_1 : \alpha \rightarrow \beta \quad \mathbf{\Gamma} \vdash e_2 : \alpha}{\mathbf{\Gamma} \vdash e_1\, e_2 : \beta}$
Let	$\dfrac{\mathbf{\Gamma} \vdash e_1 : \alpha \quad \mathbf{\Gamma} \cup \{x : \alpha\} \vdash e_2 : \beta}{\mathbf{\Gamma} \vdash let\ x = e_1\ in\ e_2 : \beta}$

5. Type constraints

Expression	**Constraint**
Variable	$\dfrac{\mathbf{\Gamma}(x) = \bot}{\mathbf{\Gamma} \vdash x : a}\ [a \approx \alpha]$
	$\dfrac{\mathbf{\Gamma}(x) = \tau}{\mathbf{\Gamma} \vdash x : a}\ [a \approx \tau']$

λ-abstraction	$\dfrac{\Gamma \cup \{x : b\} \vdash e : c}{\Gamma \vdash \lambda x.\, e : a} \; [a \approx b \to c]$
Application	$\dfrac{\Gamma \vdash e_1 : b \quad \Gamma \vdash e_2 : c}{\Gamma \vdash e_1\, e_2 : a} \; [b \approx c \to a]$
Let	$\dfrac{\Gamma \vdash e_1 : b \quad \Gamma \cup \{x : b\} \vdash e_2 : c}{\Gamma \vdash let\ x = e_1\ in\ e_2 : a} \; [a \approx c]$

III. Test programs

1. Find the n-th fibonacci number

IMP Code	Type inference

func fib(n) {
    if n <= 1
        r := n
    else
        r := fib(n - 1) + fib(n - 2)
    end;
    return r
};
 	
func main() {
    n := fib(9);
    return 0
}
	let fib = λn. ((; (((if-else ((<= n) 1)) ((:= r) n)) ((:= r) ((+ (fib ((- n) 1))) (fib ((- n) 2)))))) (return r))
	in ((:= n) (fib 9))
	
	fib: Int -> Int
	n: Int
	
	n: 34

2. Greatest common divisor

IMP Code	Type inference

func gcd(x, y) {
    while y != 0
        if y > x
            tmp := x;
            x := y;
            y := tmp
        else
            x := x - y
        end
    end;
    return x
};
 	
func main() {
	let gcd = λx λy ((; ((while ((!= y) 0)) (((if-else ((> y) x)) ((; ((; ((:= tmp) x)) ((:= x) y))) ((:= y) tmp))) ((:= x) ((- x) y))))) (return x))
	in ((:= r) ((gcd 128) 72))
	
	gcd: Int -> Int -> Int
	r: Int
	
	r: 8

r := gcd(128, 72);
return 0
}	

3. First

<table>
<tr><th>IMP Code</th><th>Type inference</th></tr>
<tr><td rowspan="3">func fst(x, y) {
    return x
};

func main() {
    f1 := fst(0, 1);
    f2 := fst(true, 1);
    return 0
}</td><td>let fst = λx λy (return x)
in ((; ((:= f1) ((fst 0) 1))) ((:= f2) ((fst true) 1)))</td></tr>
<tr><td>fst: α -> β -> α
f1: Int
f2: Bool</td></tr>
<tr><td>f1: 0
f2: true</td></tr>
</table>

4. Compare

<table>
<tr><th>IMP Code</th><th>Type inference</th></tr>
<tr><td rowspan="3">func compare(x, y) {
    return x == y
};

func main() {
    r1 := compare(1, 2);
    r2 := compare(true, true);
    return 0
}</td><td>let compare = λx λy (return ((== x) y))
in ((; ((:= r1) ((compare 1) 2))) ((:= r2) ((compare true) true)))</td></tr>
<tr><td>compare: α -> α -> Bool
r1: Bool
r2: Bool</td></tr>
<tr><td>r1: false
r2: true</td></tr>
</table>