Minor Research Report 3

15/08/2023

Student: TRAN, Thanh Cong
Supervisor: Nao Hirokawa

I. Overview

- Objectives:
  - Implement support for function & call stack in the IMP Intepreter
  - Implement unification function in Typescript
- Progress: Done
- Source code:
  - IMP Intepreter: https://github.com/thanhtcptit/typescript-mr/blob/main/imp_interpreter/run.ts
  - Unification: https://github.com/thanhtcptit/imp-typescript/blob/main/code_samples/unification.ts

II. IMP Intepreter Implementation

1. Function definition parser

Structure	Example

SYNTAX “func” + VARIABLE + SYNTAX “(“ + Optional(Repeat(VARIABLE, “,”)) + SYNTAX “)” + SYNTAX “{“ + Optional(BlockParser) + SYNTAX “return” + (Arithmetic expression | Logic expression) + SYNTAX “}”	func add(x, y) {
	  return x + y
	};

2. Function call parser

Structure	Example
VARIABLE + SYNTAX “(“ + Repeat(Arithmetic expression | Logic expression, “,”) + SYNTAX “)”	z := add(x, y);

3. Frame
   - Purpose: Serve as an environment for a function call
   - Each frame has a Map<string, value> to store the variables used in the current function execution
4. Call Stack
   - Purpose: Manage a stack of frames for function execution, allocating new frame to the top of the stack when a function is called, and deallocating the top frame when a function is finished

III. Test programs

- Arithmetic and logic operator

IMP program	Interpreter's environment

func main() {
  a := (1 + (2 - 6)) + 3;
  b := (a + 1) - 2;
  c := a >= 0 || b >= 0;
 	
  if c
  d := !c
  end;
 	
  return 0
}
	a: 0
	b: -1
	c: true
	d: false

- Greatest common divisor

IMP program	Interpreter's environment

func main() {
  x := 128;
  y := 72;
 	
  while y != 0
  if y > x
  tmp := x;
  x := y;
  y := tmp
  else
  x := x - y
  end
  end;
 	
  return 0
}
	x: 8
	y: 0
	tmp: 0

- sum([1, n])

IMP program	Interpreter's environment

func add(x, y) {
  return x + y
};
 	
func main() {
  n := 10;
  sum := 0;
 	
  while n > 0
  sum := add(sum, n);
  n := n - 1
  end;
 	
  return 0
}
	n: 0
	sum: 55

- Find the N Fibonacci number

IMP program	Interpreter's environment

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

- Unification (Typescript)

Test inputs	Outputs
- {add(0, x), add(0, s(x)} - {add(0, x), add(x, s(y)} - {add(x, s(y), add(s(y), s(z))}	- Failed - Failed - {x -> s(z), y -> z}