Seminar 10
week 10 (2 December 2019 – 6 December 2019)

Toy language syntax:

Type ::= int
| bool
| string
| Ref Type
| void

Stmt ::= Stmt;Stmt
| Id= Exp
| Type Id
| print(Exp)
| If Exp Then Stmt1 else Stm2
| Nop
| openRFile(Exp)
| readFile(Exp, id)
| closeRFile(Exp)
| new(Id, Exp)
| wH(Id, Exp)
| while Exp Stmt
| fork(Stmt)

Value ::= Number
| True
|False
| String
| (value, Type) //ref value

Exp ::= Value
| id
| rH(Exp)
| Exp1 + Exp2
| Exp1 - Exp2
| Exp1 * Exp2
| Exp1 / Exp2
| Exp1 and Exp2
| Exp1 or Exp2
| Exp1 < Exp2
| Exp1 <= Exp2
| Exp1 == Exp2
| Exp1 != Exp2
| Exp1 > Exp2
| Exp1 >= Exp2

Types

2 : int
id : type (given by the programmer)

Type rules

1: int 2:int
----
1 + 2 : int

is reading as:
-- 1+2 has type int IF 1 has type int and 2 has type int
or
-- IF 1 has type int and 2 has type int THEN 1+2 has type int

1: int v:??
----
1 + v : int

G- type environment, defined as a list of pairs (id:type)

G|-1: int G|-v:int
---------- where G=[v:int]
G|-1 + v : int

Type rules for Values

G|- Number : int

G|-True:bool

G|-False:bool

G|-String:string

G|- val: int

G|-(val,type) : Ref type

Type rules for Expressions

(id:t) is in G

G|- id:t

G|- e1: int G|-e2:int

G|-e1 + e2:int

the same rule for -,*,/

G|- e1: bool G|-e2:bool

G|-e1 and e2:bool
the same rule for or

G|- e1: int G|-e2:int

G|-e1 < e2:bool
the same rule for <=,==,!=,>, >=

G|- e1 :Ref t1

G|- rH(e1): t1

Type rules for Statements

G|- s1:void,G1 G1|- s2:void,G2

G|- s1;s2: void,G2

G|- id:t G|- exp:t

G|- id=exp: void,G

G|- type id : void, G+[(id:type)]

G|- exp:t

G|- print(exp): void,G

G|- e : bool
G|- s1:void,G1
G|- s2:void,G2

G|- if e then s1 else s2 : void,G

G|- nop:void, G

G|- exp:string G|-id:int

G|- readFile(exp,id):void,G

G|- exp:string

G|- openRFile(exp):void, G

G|- exp:string

G|- closeRFile(exp):void, G

G|- exp:t G|- id:Ref t

G|- new(id,exp):void, G

G|- exp:t G|- id:Ref t

G|- wH(id,exp):void, G

G|- exp:bool G|- stmt:void,G1

G|- while exp stmt:void, G

G|- stmt:void,G1

G|- fork(stmt):void, G