

# CS320 Programming Languages

## Homework #9: mini-Scala

**Due: 11 December 2019**

The goal of Homework #9 is to implement the interpreter of **mini-Scala**, which is a simplified version of **Scala** including several language features introduced in this course (e.g. recursive functions, variables, lazy evaluations, types, and variants). Implement its type checker and the interpreter:

Programs	$p ::= s^* e$	
Statements	$s ::= \text{val } x:\tau = e$ $\quad \mid \text{lazy val } x:\tau = e$ $\quad \mid \text{var } x:\tau = e$ $\quad \mid \text{def } x([x:\tau]^*):\tau = e$ $\quad \mid \text{trait } t [\text{case class } x(\tau^*)]^+$	value declarations lazy value declarations variable declarations recursive functions traits
Expressions	$e ::= n$ $\quad \mid b$ $\quad \mid e + e$ $\quad \mid e - e$ $\quad \mid e == e$ $\quad \mid e < e$ $\quad \mid x$ $\quad \mid ([x:\tau]^*) \Rightarrow e$ $\quad \mid e (e^*)$ $\quad \mid \{s^* e\}$ $\quad \mid x = e$ $\quad \mid e \text{ match } \{[\text{case } x(x^*) \Rightarrow e]^+\}$ $\quad \mid \text{if } (e) e \text{ else } e$	numerical literals boolean literals numerical additions numerical subtractions numerical equality numerical inequalities identifiers functions applications block expressions assignments pattern matches conditional branches
Types	$\tau ::= \text{int}$ $\quad \mid \text{bool}$ $\quad \mid (\tau^*) \Rightarrow \tau$ $\quad \mid t$	integer types boolean types arrow types type identifiers
Values	$v ::= n$ $\quad \mid b$ $\quad \mid \langle \lambda x^*. e, \sigma \rangle$ $\quad \mid \text{constructorV}(x)$ $\quad \mid \text{variantV}(x, v^*)$ $\quad \mid (e, \sigma)$ $\quad \mid a$	integers boolean values function closures constructors variants expression values addresses

## 1 Typing Rules

Each typing rule is related to the `typecheck` function in `Homework09.scala`.

- $\boxed{p : \tau}$  for programs

$$\frac{\emptyset \vdash s_1 : \Gamma_1 \quad \cdots \quad \Gamma_{n-1} \vdash s_n : \Gamma_n \quad \Gamma_n \vdash e : \tau}{s_1 \cdots s_n e : \tau}$$

- $\boxed{\Gamma \vdash s : \Gamma}$  for statements

$$\frac{\Gamma \vdash \tau \quad \Gamma \vdash e : \tau}{\Gamma \vdash \text{val } x : \tau = e : \Gamma[x \mapsto \tau]} \quad \frac{\Gamma \vdash \tau \quad \Gamma \vdash e : \tau}{\Gamma \vdash \text{lazy val } x : \tau = e : \Gamma[x \mapsto \tau]} \quad \frac{\Gamma \vdash \tau \quad \Gamma \vdash e : \tau}{\Gamma \vdash \text{var } x : \tau = e : \Gamma[x \mapsto \tau, x : \text{mutable}]}$$

$$\frac{\Gamma \vdash \tau_1 \quad \cdots \quad \Gamma \vdash \tau_n \quad \Gamma \vdash \tau \quad \Gamma' = \Gamma[x \mapsto (\tau_1, \dots, \tau_n) \Rightarrow \tau] \quad \Gamma'[x_1 \mapsto \tau_1, \dots, x_n \mapsto \tau_n] \vdash e : \tau}{\Gamma \vdash \text{def } x(\tau_1, \dots, \tau_n) : \tau = e : \Gamma'}$$

$$\frac{\Gamma' = \Gamma[t = x_1 @ (\tau_{11}, \dots, \tau_{1m_1}) + \dots + x_n @ (\tau_{n1}, \dots, \tau_{nm_n})] \quad \Gamma'' = \Gamma'[x_1 \mapsto (\tau_{11}, \dots, \tau_{1m_1}) \Rightarrow t, \dots, x_n \mapsto (\tau_{n1}, \dots, \tau_{nm_n}) \Rightarrow t] \quad \Gamma'' \vdash \tau_{11} \quad \cdots \quad \Gamma'' \vdash \tau_{nm_n}}{\Gamma \vdash \text{trait } t \text{ case class } x_1(\tau_{11}, \dots, \tau_{1m_1}) \cdots \text{case class } x_n(\tau_{n1}, \dots, \tau_{nm_n}) : \Gamma''}$$

- $\boxed{\Gamma \vdash e : \tau}$  for expressions

$$\Gamma \vdash n : \text{int} \quad \Gamma \vdash b : \text{bool} \quad \frac{\Gamma \vdash e_1 : \text{int} \quad \Gamma \vdash e_2 : \text{int}}{\Gamma \vdash e_1 + e_2 : \text{int}} \quad \frac{\Gamma \vdash e_1 : \text{int} \quad \Gamma \vdash e_2 : \text{int}}{\Gamma \vdash e_1 - e_2 : \text{int}}$$

$$\frac{\Gamma \vdash e_1 : \text{int} \quad \Gamma \vdash e_2 : \text{int}}{\Gamma \vdash e_1 == e_2 : \text{bool}} \quad \frac{\Gamma \vdash e_1 : \text{int} \quad \Gamma \vdash e_2 : \text{int}}{\Gamma \vdash e_1 < e_2 : \text{bool}} \quad \frac{x \in \text{Domain}(\Gamma)}{\Gamma \vdash x : \Gamma(x)}$$

$$\frac{\Gamma \vdash \tau_1 \quad \cdots \quad \Gamma \vdash \tau_n \quad \Gamma[x_1 \mapsto \tau_1, \dots, x_n \mapsto \tau_n] \vdash e : \tau}{\Gamma \vdash (x_1 : \tau_1, \dots, x_n : \tau_n) \Rightarrow e : (\tau_1, \dots, \tau_n) \Rightarrow \tau}$$

$$\frac{\Gamma \vdash e : (\tau_1, \dots, \tau_n) \Rightarrow \tau \quad \Gamma \vdash e_1 : \tau_1 \quad \cdots \quad \Gamma \vdash e_n : \tau_n}{\Gamma \vdash e(e_1, \dots, e_n) : \tau}$$

$$\frac{\Gamma \vdash s_1 : \Gamma_1 \quad \cdots \quad \Gamma_{n-1} \vdash s_n : \Gamma_n \quad \Gamma_n \vdash e : \tau}{\Gamma \vdash \{s_1 \cdots s_n e\} : \tau} \quad \frac{\Gamma(x) = \tau \quad x : \text{mutable} \in \Gamma \quad \Gamma \vdash e : \tau}{\Gamma \vdash x = e : \tau}$$

$$\frac{\Gamma(t) = x_1 @ (\tau_{11}, \dots, \tau_{1m_1}) + \dots + x_n @ (\tau_{n1}, \dots, \tau_{nm_n}) \quad \Gamma \vdash e : t \quad \Gamma[x_{11} \mapsto \tau_{11}, \dots, x_{1m_1} \mapsto \tau_{1m_1}] \vdash e_1 : \tau \quad \cdots \quad \Gamma[x_{n1} \mapsto \tau_{n1}, \dots, x_{nm_n} \mapsto \tau_{nm_n}] \vdash e_n : \tau}{\Gamma \vdash e \text{ match } \{\text{case } x_1(x_{11}, \dots, x_{1m_1}) \Rightarrow e_1 \cdots \text{case } x_n(x_{n1}, \dots, x_{nm_n}) \Rightarrow e_n\} : \tau}$$

$$\frac{\Gamma \vdash e_0 : \text{bool} \quad \Gamma \vdash e_1 : \tau \quad \Gamma \vdash e_2 : \tau}{\Gamma \vdash \text{if } (e_0) e_1 \text{ else } e_2 : \tau}$$

where  $\boxed{\Gamma \vdash \tau}$  denotes the type validity with the following rules:

$$\Gamma \vdash \text{int} \quad \Gamma \vdash \text{bool} \quad \frac{\Gamma \vdash \tau_1 \quad \cdots \quad \Gamma \vdash \tau_n \quad \Gamma \vdash \tau}{\Gamma \vdash (\tau_1, \dots, \tau_n) \Rightarrow \tau}$$

$$\frac{\Gamma(t) = x_1 @ (\tau_{11}, \dots, \tau_{1m_1}) + \dots + x_n @ (\tau_{n1}, \dots, \tau_{nm_n})}{\Gamma \vdash t}$$

## 2 Operational Semantics

Each semantics rule is related to the `interp` function in `Homework09.scala`.

- $\boxed{p \Rightarrow v}$  for programs

$$\frac{\emptyset, \emptyset \vdash s_1 \Rightarrow \sigma_1, M_1 \quad \cdots \quad \sigma_{n-1}, M_{n-1} \vdash s_n \Rightarrow \sigma_n, M_n \quad \sigma_n, M_n \vdash e \Rightarrow v, M}{s_1 \cdots s_n e \Rightarrow v}$$

- $\boxed{\sigma, M \vdash s \Rightarrow \sigma, M}$  for statements

$$\frac{\sigma, M \vdash e \Rightarrow v, M'}{\sigma, M \vdash \text{val } x:\tau = e \Rightarrow \sigma[x \mapsto v], M'} \quad \frac{a \notin \text{Domain}(M)}{\sigma, M \vdash \text{lazy val } x:\tau = e \Rightarrow \sigma[x \mapsto a], M[a \mapsto (e, \sigma)]}$$

$$\frac{\sigma, M \vdash e \Rightarrow v, M' \quad a \notin \text{Domain}(M')}{\sigma, M \vdash \text{var } x:\tau = e \Rightarrow \sigma[x \mapsto a], M'[a \mapsto v]} \quad \frac{\sigma' = \sigma[x \mapsto \langle \lambda x_1, \dots, x_n. e, \sigma' \rangle]}{\sigma, M \vdash \text{def } x(x_1:\tau_1, \dots, x_n:\tau_n):\tau = e \Rightarrow \sigma', M}$$

$$\frac{\sigma' = \sigma[x_1 \mapsto \text{constructorV}(x_1), \dots, x_n \mapsto \text{constructorV}(x_n)]}{\sigma, M \vdash \text{trait } t \text{ case class } x_1(\tau_{11}, \dots, \tau_{1m_1}) \cdots \text{case class } x_n(\tau_{n1}, \dots, \tau_{nm_n}) \Rightarrow \sigma', M}$$

- $\boxed{\sigma, M \vdash e \Rightarrow v, M}$  for expressions

$$\sigma, M \vdash n \Rightarrow n, M \quad \sigma, M \vdash b \Rightarrow b, M$$

$$\frac{\sigma, M \vdash e_1 \Rightarrow n_1, M' \quad \sigma, M' \vdash e_2 \Rightarrow n_2, M''}{\sigma, M \vdash e_1 + e_2 \Rightarrow n_1 + n_2, M''} \quad \frac{\sigma, M \vdash e_1 \Rightarrow n_1, M' \quad \sigma, M' \vdash e_2 \Rightarrow n_2, M''}{\sigma, M \vdash e_1 - e_2 \Rightarrow n_1 - n_2, M''}$$

$$\frac{\sigma, M \vdash e_1 \Rightarrow n_1, M' \quad \sigma, M' \vdash e_2 \Rightarrow n_2, M''}{\sigma, M \vdash e_1 = e_2 \Rightarrow n_1 = n_2, M''} \quad \frac{\sigma, M \vdash e_1 \Rightarrow n_1, M' \quad \sigma, M' \vdash e_2 \Rightarrow n_2, M''}{\sigma, M \vdash e_1 < e_2 \Rightarrow n_1 < n_2, M''}$$

$$\frac{\sigma(x) \text{ is not an address}}{\sigma, M \vdash x \Rightarrow \sigma(x), M}$$

$$\frac{\sigma(x) = a \quad M(a) = (e, \sigma') \quad \sigma', M \vdash e \Rightarrow v, M'}{\sigma, M \vdash x \Rightarrow v, M'[a \mapsto v]} \quad \frac{\sigma(x) = a \quad M(a) \text{ is not an expression value}}{\sigma, M \vdash x \Rightarrow M(a), M}$$

$$\sigma, M \vdash (x_1:\tau_1, \dots, x_n:\tau_n) \Rightarrow e \Rightarrow \langle \lambda x_1, \dots, x_n. e, \sigma \rangle, M$$

$$\frac{\sigma, M \vdash e_0 \Rightarrow \langle \lambda x_1, \dots, x_n. e, \sigma' \rangle, M_0 \quad \sigma, M_0 \vdash e_1 \Rightarrow v_1, M_1 \quad \cdots \quad \sigma, M_{n-1} \vdash e_n \Rightarrow v_n, M_n \quad \sigma'[x_1 \mapsto v_1, \dots, x_n \mapsto v_n], M_n \vdash e \Rightarrow v, M'}{\sigma, M \vdash e_0(e_1, \dots, e_n) \Rightarrow v, M'}$$

$$\frac{\sigma, M \vdash e_0 \Rightarrow \text{constructorV}(x), M_0 \quad \sigma, M_0 \vdash e_1 \Rightarrow v_1, M_1 \quad \cdots \quad \sigma, M_{n-1} \vdash e_n \Rightarrow v_n, M_n}{\sigma, M \vdash e_0(e_1, \dots, e_n) \Rightarrow \text{variantV}(x, v_1, \dots, v_n), M_n}$$

$$\frac{\sigma, M \vdash s_1 \Rightarrow \sigma_1, M_1 \quad \dots \quad \sigma_{n-1}, M_{n-1} \vdash s_n \Rightarrow \sigma_n, M_n \quad \sigma_n, M_n \vdash e \Rightarrow v, M'}{\sigma, M \vdash \{s_1 \dots s_n e\} \Rightarrow v, M'}$$

$$\frac{\sigma(x) = a \quad \sigma, M \vdash e \Rightarrow v, M'}{\sigma, M \vdash x = e \Rightarrow v, M'[a \mapsto v]}$$

$$\frac{\sigma, M \vdash e \Rightarrow \text{variantV}(x_i, v_1, \dots, v_{m_i}), M' \quad \sigma[x_{i1} \mapsto v_1, \dots, x_{im_i} \mapsto v_{m_i}], M' \vdash e_i \Rightarrow v, M''}{\sigma, M \vdash e \text{ match } \{\text{case } x_1(x_{11}, \dots, x_{1m_1}) \Rightarrow e_1 \dots \text{case } x_n(x_{n1}, \dots, x_{nm_n}) \Rightarrow e_n\} \Rightarrow v, M''}$$

$$\frac{\sigma, M \vdash e_0 \Rightarrow \text{true}, M' \quad \sigma, M' \vdash e_1 \Rightarrow v, M''}{\sigma, M \vdash \text{if } (e_0) e_1 \text{ else } e_2 \Rightarrow v, M''} \quad \frac{\sigma, M \vdash e_0 \Rightarrow \text{false}, M' \quad \sigma, M' \vdash e_2 \Rightarrow v, M''}{\sigma, M \vdash \text{if } (e_0) e_1 \text{ else } e_2 \Rightarrow v, M''}$$

### 3 Tests

```
test(run("""
  var x: Int = 1
  val y: Int = (x = 3)
  x + y
"""), "6")
test(run("""
  var x: Int = 1
  lazy val y: Int = (x = 3)
  x + y + x
"""), "7")
test(run("""
  var x: Int = 0
  lazy val y: Int = (x = x + 1)
  val z: Int = y + y + y + y
  z"""), "4")
testExc(run("""val x: Int = 42; x = 24"""), "")
test(run("""
  trait AE
  case class Num(Int)
  case class Add(AE, AE)
  case class Sub(AE, AE)

  def interp(e: AE): Int = e match {
    case Num(n) => n
    case Add(l, r) => interp(l) + interp(r)
    case Sub(l, r) => interp(l) - interp(r)
  }

  interp(Add(Num(2), Sub(Num(3), Num(1))))
"""), "4")
test(run("""
  trait Tree
  case class Leaf(Int)
  case class Node(Tree, Tree)

  def max(l: Int, r: Int): Int =
    if (l < r) r else l

  def depth(e: Tree): Int = e match {
    case Leaf(n) => 1
    case Node(l, r) => max(depth(l), depth(r)) + 1
  }
"""))
```

```
}  
  
depth(Node(Node(Leaf(1), Node(Leaf(2), Leaf(3))), Leaf(4)))  
""), "4")
```