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

1   Typing Rules

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

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

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

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

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

$$\frac{\begin{array}{c}\Gamma \vdash \tau_1 \qquad \cdots \qquad \Gamma \vdash \tau_n \qquad \Gamma \vdash \tau \\ \Gamma' = \Gamma[x \mapsto (\tau_1, \cdots, \tau_n)\ \mathtt{=>}\ \tau] \qquad \Gamma'[x_1 \mapsto \tau_1, \cdots, x_n \mapsto \tau_n] \vdash e : \tau\end{array}}{\Gamma \vdash \mathtt{def}\ x(x_1{:}\tau_1, \cdots, x_n{:}\tau_n){:}\tau\ \mathtt{=}\ e : \Gamma'}$$

$$\frac{\begin{array}{c}\Gamma' = \Gamma[t = x_1@(\tau_{11}, \cdots, \tau_{1m_1}) + \cdots + x_n@(\tau_{n1}, \cdots, \tau_{nm_n})] \\ \Gamma'' = \Gamma'[x_1 : (\tau_{11}, \cdots, \tau_{1m_1})\ \mathtt{=>}\ t, \cdots, x_n : (\tau_{n1}, \cdots, \tau_{nm_n})\ \mathtt{=>}\ t] \\ \Gamma'' \vdash \tau_{11} \qquad \cdots \qquad \Gamma'' \vdash \tau_{nm_n}\end{array}}{\Gamma \vdash \mathtt{trait}\ t\ \mathtt{case\ class}\ x_1(\tau_{11}, \cdots, \tau_{1m_1}) \cdots \mathtt{case\ class}\ x_n(\tau_{n1}, \cdots, \tau_{nm_n}) : \Gamma''}$$

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

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

$$\frac{\Gamma \vdash e_1 : \mathtt{int} \qquad \Gamma \vdash e_2 : \mathtt{int}}{\Gamma \vdash e_1\ \mathtt{==}\ e_2 : \mathtt{bool}} \qquad \frac{\Gamma \vdash e_1 : \mathtt{int} \qquad \Gamma \vdash e_2 : \mathtt{int}}{\Gamma \vdash e_1\ \mathtt{<}\ e_2 : \mathtt{bool}} \qquad \frac{x \in \mathrm{Domain}(\Gamma)}{\Gamma \vdash x : \Gamma(x)}$$

$$\frac{\Gamma \vdash \tau_1 \qquad \cdots \qquad \Gamma \vdash \tau_n \qquad \Gamma[x_1 \mapsto \tau_1, \cdots, x_n \mapsto \tau_n] \vdash e : \tau}{\Gamma \vdash (x_1{:}\tau_1, \cdots, x_n{:}\tau_n)\ \mathtt{=>}\ e : (\tau_1, \cdots, \tau_n)\ \mathtt{=>}\ \tau}$$

$$\frac{\Gamma \vdash e : (\tau_1, \cdots, \tau_n)\ \mathtt{=>}\ \tau \qquad \Gamma \vdash e_1 : \tau_1 \qquad \cdots \qquad \Gamma \vdash e_n : \tau_n}{\Gamma \vdash e(e_1, \cdots, e_n) : \tau}$$

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

$$\frac{\begin{array}{c}\Gamma(t) = x_1@(\tau_{11}, \cdots, \tau_{1m_1}) + \cdots + x_n@(\tau_{n1}, \cdots, \tau_{nm_n}) \\ \Gamma \vdash e : t \qquad \Gamma[x_{11} \mapsto \tau_{11}, \cdots, x_{1m_1} \mapsto \tau_{1m_1}] \vdash e_1 : \tau \qquad \cdots \qquad \Gamma[x_{n1} \mapsto \tau_{n1}, \cdots, x_{nm_n} \mapsto \tau_{nm_n}] \vdash e_n : \tau\end{array}}{\Gamma \vdash e\ \mathtt{match}\ \{\mathtt{case}\ x_1(x_{11}, \cdots, x_{1m_1})\ \mathtt{=>}\ e_1 \cdots \mathtt{case}\ x_n(x_{n1}, \cdots, x_{nm_n})\ \mathtt{=>}\ e_n\} : \tau}$$

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

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

$$\Gamma \vdash \mathtt{int} \qquad \Gamma \vdash \mathtt{bool} \qquad \frac{\Gamma \vdash \tau_1 \qquad \cdots \qquad \Gamma \vdash \tau_n \qquad \Gamma \vdash \tau}{\Gamma \vdash (\tau_1, \cdots, \tau_n)\ \mathtt{=>}\ \tau}$$

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

2 Operational Semantics

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\sigma, M \vdash (x_1{:}\tau_1, \cdots, x_n{:}\tau_n)\ \mathtt{=>}\ e \Rightarrow \langle \lambda x_1, \cdots, x_n.\ e, \sigma \rangle, M$$

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

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

3

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

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

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

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

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")
```