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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

2 Operational Semantics

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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")
```