CS320

Evaluation of WAE

Sukyoung Ryu

March 11, 2019

Homework

- To help you understand PL better.
- To see whether you're following well.
- Start early!
- Utilize office hours and the Piazza board.
- Tests for a given template language may not work for your new language.

WAE: Concrete Syntax

```
<WAE> ::= <num>
        | {+ <WAE> <WAE>}
        | {- <WAE> <WAE>}
        | {with {<id> <WAE>} <WAE>}
        | <id>
```

WAE: Abstract Syntax

```
trait WAE
case class Num(n: Int) extends WAE
case class Add(l: WAE, r: WAE) extends WAE
case class Sub(l: WAE, r: WAE) extends WAE
case class With(x: String, i: WAE, b: WAE) extends WAE
case class Id(x: String) extends WAE
```

Evaluation

```
interp(WAE("{with {x 1}      []
             {with {y 2}
               {+ 100 {+ 99 {+ 98 ··· {+ y x} ···}"))
⇒
interp(WAE("{with {y 2}      [x=1]
             {+ 100 {+ 99 {+ 98 ··· {+ y x} ···}"))
⇒
interp(WAE("{+ 100 {+ 99 {+ 98 ··· {+ y x}···}")) [y=2 x=1]
⇒ ... ⇒
interp(WAE("y"))             [y=2 x=1]
```

Evaluation

```
interp(WAE("{with {x 1}      []
             {with {y 2}
               {+ 100 {+ 99 {+ 98 ··· {+ y x} ···}"))
⇒
interp(WAE("{with {y 2}      [x=1]
             {+ 100 {+ 99 {+ 98 ··· {+ y x} ···}"))
⇒
interp(WAE("{+ 100 {+ 99 {+ 98 ··· {+ y x}···}")) [y=2 x=1]
⇒ ... ⇒
interp(WAE("y"))             [y=2 x=1]
```

Evaluation

```
interp(WAE("{with {x 1}      []
             {with {y 2}
               {+ 100 {+ 99 {+ 98 ··· {+ y x} ···}"))
⇒
interp(WAE("{with {y 2}      [x=1]
             {+ 100 {+ 99 {+ 98 ··· {+ y x} ···}"))
⇒
interp(WAE("{+ 100 {+ 99 {+ 98 ··· {+ y x}···}")) [y=2 x=1]
⇒ ... ⇒
interp(WAE("y"))             [y=2 x=1]
```

Evaluation

```
interp(WAE("{with {x 1}      []
             {with {y 2}
               {+ 100 {+ 99 {+ 98 ··· {+ y x} ···}"))
⇒
interp(WAE("{with {y 2}      [x=1]
             {+ 100 {+ 99 {+ 98 ··· {+ y x} ···}"))
⇒
interp(WAE("{+ 100 {+ 99 {+ 98 ··· {+ y x}···}")) [y=2 x=1]
⇒ ... ⇒
interp(WAE("y"))             [y=2 x=1]
```

Evaluation

```
interp(WAE("{with {x 1}          []
             {with {x 2}
               x}}"))
```

$\Rightarrow$

```
interp(WAE("{with {x 2}          [x=1]
             x}"))
```

$\Rightarrow$

```
interp(WAE("x"))                 [x=2]
```

A new mapping replaces any old value.

Evaluation

```
interp(WAE("{with {x 1}          []
             {with {x 2}
               x}}"))
```

$\Rightarrow$

```
interp(WAE("{with {x 2}          [x=1]
             x}"))
```

$\Rightarrow$

```
interp(WAE("x"))                 [x=2]
```

A new mapping replaces any old value.

Evaluation

```
interp(WAE("{with {x 1}          []
             {with {x 2}
               x}}"))
```

$\Rightarrow$

```
interp(WAE("{with {x 2}          [x=1]
             x}"))
```

$\Rightarrow$

```
interp(WAE("x"))                 [x=2]
```

A new mapping replaces any old value.

Evaluation

```
interp(WAE("{with {x 1}          []
             {with {x 2}
               x}}"))
```

$\Rightarrow$

```
interp(WAE("{with {x 2}          [x=1]
             x}"))
```

$\Rightarrow$

```
interp(WAE("x"))                 [x=2]
```

A new mapping replaces any old value.

Evaluation

```
interp(WAE("{with {x 1}          []
              {+ {with {x 2} x}
                 x}}"))
```

$\Rightarrow$

```
interp(WAE("{+ {with {x 2} x} [x=1]
              x}"))
```

$\Rightarrow$

```
interp(WAE("{with {x 2} x}")) [x=1] +
interp(WAE("x")) [x=1]
```

$\Rightarrow$

```
interp(WAE("x")) [x=2] + interp(WAE("x")) [x=1]
```

$\Rightarrow$ `2 + 1`

Evaluation

```
interp(WAE("{with {x 1}          []
              {+ {with {x 2} x}
                 x}}"))
```

$\Rightarrow$

```
interp(WAE("{+ {with {x 2} x} [x=1]
               x}"))
```

$\Rightarrow$

```
interp(WAE("{with {x 2} x}")) [x=1] +
interp(WAE("x")) [x=1]
```

$\Rightarrow$

```
interp(WAE("x")) [x=2] + interp(WAE("x")) [x=1]
```

$\Rightarrow$ `2 + 1`

Evaluation

```
interp(WAE("{with {x 1}          []
              {+ {with {x 2} x}
                 x}}"))
```

$\Rightarrow$

```
interp(WAE("{+ {with {x 2} x} [x=1]
               x}"))
```

$\Rightarrow$

```
interp(WAE("{with {x 2} x}")) [x=1] +
interp(WAE("x")) [x=1]
```

$\Rightarrow$

```
interp(WAE("x")) [x=2] + interp(WAE("x")) [x=1]
```

$\Rightarrow$ `2 + 1`

Evaluation

```
interp(WAE("{with {x 1}          []
              {+ {with {x 2} x}
                 x}}"))
```

$\Rightarrow$

```
interp(WAE("{+ {with {x 2} x} [x=1]
              x}"))
```

$\Rightarrow$

```
interp(WAE("{with {x 2} x}")) [x=1] +
interp(WAE("x")) [x=1]
```

$\Rightarrow$

```
interp(WAE("x")) [x=2] + interp(WAE("x")) [x=1]
```

$\Rightarrow$ 2 + 1

Evaluation

```
interp(WAE("{with {x 1}          []
              {+ {with {x 2} x}
                 x}}"))
```

$\Rightarrow$

```
interp(WAE("{+ {with {x 2} x} [x=1]
              x}"))
```

$\Rightarrow$

```
interp(WAE("{with {x 2} x}")) [x=1] +
interp(WAE("x")) [x=1]
```

$\Rightarrow$

```
interp(WAE("x")) [x=2] + interp(WAE("x")) [x=1]
```

$\Rightarrow$ `2 + 1`

Evaluation with Environment

Change

```
// interp : WAE => Int
```

to

```
// interp : (WAE, Env) => Int
```

```
type Env = Map[String, Int]

val x: Env = Map()
val y: Env = Map("a" -> 0, "b" -> 1)
y + ("b" -> 2)        // Map("a" -> 0, "b" -> 2)
y + ("d" -> 42)       // Map("a" -> 0, "b" -> "1", "d" -> 42)
y - "a"               // Map("b" -> 1)
y.get("a")            // Some(0)
y.get("c")            // None
y.getOrElse("a", 42) // 0
y.getOrElse("c", 42) // 42
```

Evaluation with Environment

Change

```
// interp : WAE => Int
```

to

```
// interp : (WAE, Env) => Int
```

```
type Env = Map[String, Int]

val x: Env = Map()
val y: Env = Map("a" -> 0, "b" -> 1)
y + ("b" -> 2)          // Map("a" -> 0, "b" -> 2)
y + ("d" -> 42)         // Map("a" -> 0, "b" -> "1", "d" -> 42)
y - "a"                 // Map("b" -> 1)
y.get("a")              // Some(0)
y.get("c")              // None
y.getOrElse("a", 42) // 0
y.getOrElse("c", 42) // 42
```

interp with Environment

```
interp(WAE("{with {x 1}
              {with {y 2}
                {+ 100 {+ 99 {+ 98 ··· {+ y x} ···}}}}"),
       Map())
```

⇒

```
interp(WAE("{with {y 2}
              {+ 100 {+ 99 {+ 98 ··· {+ y x} ···}}}"),
       (···))
```

interp with Environment

```
interp(WAE("{with {x 1}
              {with {y 2}
                {+ 100 {+ 99 {+ 98 ··· {+ y x} ···}}}}"),
       Map())
⇒
interp(WAE("{with {y 2}
              {+ 100 {+ 99 {+ 98 ··· {+ y x} ···}}}"),
       (···))
```

interp with Environment

```
interp(WAE("{with {x 1}
              {with {y 2}
                {+ 100 {+ 99 {+ 98 ··· {+ y x} ···}}}}"),
       Map())
```

$\Rightarrow$

```
interp(WAE("{with {y 2}
              {+ 100 {+ 99 {+ 98 ··· {+ y x} ···}}}"),
       Map("x" -> 1))
```

$\Rightarrow$

```
interp(WAE("{+ 100 {+ 99 {+ 98 ··· {+ y x} ···}}}"),
       (···))
```

interp with Environment

```
interp(WAE("{with {x 1}
              {with {y 2}
                {+ 100 {+ 99 {+ 98 ··· {+ y x} ···}}}}"),
       Map())
⇒
interp(WAE("{with {y 2}
              {+ 100 {+ 99 {+ 98 ··· {+ y x} ···}}}"),
       Map("x" -> 1))
⇒
interp(WAE("{+ 100 {+ 99 {+ 98 ··· {+ y x} ···}}}"),
       (···))
```

interp with Environment

```
interp(WAE("{with {x 1}
              {with {y 2}
                {+ 100 {+ 99 {+ 98 ··· {+ y x} ···}}}}"),
       Map())
⇒
interp(WAE("{with {y 2}
              {+ 100 {+ 99 {+ 98 ··· {+ y x} ···}}}"),
       Map("x" -> 1))
⇒
interp(WAE("{+ 100 {+ 99 {+ 98 ··· {+ y x} ···}}}"),
       Map("x" -> 1, "y" -> 2))
⇒ ...
⇒
interp(WAE("y"), Map("x" -> 1, "y" -> 2))
```

interp with Environment

```
interp(WAE("{with {x 1}
              {with {y 2}
                {+ 100 {+ 99 {+ 98 ··· {+ y x} ···}}}}"),
       Map())
⇒
interp(WAE("{with {y 2}
              {+ 100 {+ 99 {+ 98 ··· {+ y x} ···}}}"),
       Map("x" -> 1))
⇒
interp(WAE("{+ 100 {+ 99 {+ 98 ··· {+ y x} ···}}}"),
       Map("x" -> 1, "y" -> 2))
⇒ ...
⇒
interp(WAE("y"), Map("x" -> 1, "y" -> 2))
```

WAE Interpreter with Environment

```
// interp : (WAE, Env) => Int
def interp(wae: WAE, env: Env): Int = wae match {
  case Num(n) => n
  case Add(l, r) => interp(l, env) + interp(r, env)
  case Sub(l, r) => interp(l, env) - interp(r, env)
  case With(x, i, b) => ...
  case Id(x) => ...
}
```

WAE Interpreter with Environment

```
// interp : (WAE, Env) => Int
def interp(wae: WAE, env: Env): Int = wae match {
  case Num(n) => n
  case Add(l, r) => interp(l, env) + interp(r, env)
  case Sub(l, r) => interp(l, env) - interp(r, env)
  case With(x, i, b) => ...
  case Id(x) => lookup(x, env)
}
```

WAE Interpreter with Environment

```
; lookup : (String, Env) => Int
def lookup(name: String, env: Env): Int =
  env.get(name) match {
    case Some(v) => v
    case None => error(s"free identifier: $name")
  }
```

WAE Interpreter with Environment

```
; lookup : (String, Env) => Int
def lookup(name: String, env: Env): Int =
  env.getOrElse(name, error(s"free identifier: $name"))
```

WAE Interpreter with Environment

```
// interp : (WAE, Env) => Int
def interp(wae: WAE, env: Env): Int = wae match {
  case Num(n) => n
  case Add(l, r) => interp(l, env) + interp(r, env)
  case Sub(l, r) => interp(l, env) - interp(r, env)
  case With(x, i, b) => ...
  case Id(x) => lookup(x, env)
}
```

WAE Interpreter with Environment

```
// interp : (WAE, Env) => Int
def interp(wae: WAE, env: Env): Int = wae match {
  case Num(n) => n
  case Add(l, r) => interp(l, env) + interp(r, env)
  case Sub(l, r) => interp(l, env) - interp(r, env)
  case With(x, i, b) => ... interp(i, env) ...
  case Id(x) => lookup(x, env)
}
```

WAE Interpreter with Environment

```
// interp : (WAE, Env) => Int
def interp(wae: WAE, env: Env): Int = wae match {
  case Num(n) => n
  case Add(l, r) => interp(l, env) + interp(r, env)
  case Sub(l, r) => interp(l, env) - interp(r, env)
  case With(x, i, b) => ... env + (x -> interp(i, env)) ...
  case Id(x) => lookup(x, env)
}
```

WAE Interpreter with Environment

```
// interp : (WAE, Env) => Int
def interp(wae: WAE, env: Env): Int = wae match {
  case Num(n) => n
  case Add(l, r) => interp(l, env) + interp(r, env)
  case Sub(l, r) => interp(l, env) - interp(r, env)
  case With(x, i, b) => interp(b, env + (x -> interp(i, env)))
  case Id(x) => lookup(x, env)
}
```

Growing the Language

From AE to WAE

WAE: Concrete Syntax

```
<WAE> ::= <num>
        | {+ <WAE> <WAE>}
        | {- <WAE> <WAE>}
        | {with {<id> <WAE>} <WAE>}
        | <id>
```

WAE: Abstract Syntax

```
trait WAE
case class Num(n: Int) extends WAE
case class Add(l: WAE, r: WAE) extends WAE
case class Sub(l: WAE, r: WAE) extends WAE
case class With(x: String, i: WAE, b: WAE) extends WAE
case class Id(x: String) extends WAE
```

WAE: Interpreter

```
// interp : (WAE, Env) => Int
def interp(wae: WAE, env: Env): Int = wae match {
  case Num(n) => n
  case Add(l, r) => interp(l, env) + interp(r, env)
  case Sub(l, r) => interp(l, env) - interp(r, env)
  case With(x, i, b) => interp(b, env + (x -> interp(i, env)))
  case Id(x) => lookup(x, env)
}
```

Expressions and Environments

$$
\begin{array}{rll}
e ::= & n & \texttt{Num(n)} \\
 & \mid e + e & \texttt{Add(l, r)} \\
 & \mid e - e & \texttt{Sub(l, r)} \\
 & \mid \texttt{val}\ x = e\ \texttt{in}\ e & \texttt{With(x, i, b)} \\
 & \mid x & \texttt{Id(x)}
\end{array}
$$

$$
\begin{array}{rll}
n & \in \mathbb{Z} & \\
x & \in \mathit{Var} & \\
e & \in \mathit{Exp} & \\
\sigma \in \mathit{Env} & = \mathit{Var} \xrightarrow{\text{fin}} \mathbb{Z} & \texttt{type Env = Map[String, Int]} \\
v \in \mathit{Val} & = \mathbb{Z} &
\end{array}
$$

Operational Semantics

```
// interp : (WAE, Env) => Int
def interp(wae: WAE, env: Env): Int = wae match {
  case Num(n) => n
  ...
```

$$\sigma \vdash e \Rightarrow v$$

$$\sigma \vdash n \Rightarrow n$$

Operational Semantics

```
// interp : (WAE, Env) => Int
def interp(wae: WAE, env: Env): Int = wae match {
  case Num(n) => n
  ...
```

$$\sigma \vdash e \Rightarrow v$$

$$\sigma \vdash n \Rightarrow n$$

Operational Semantics

```
// interp : (WAE, Env) => Int
def interp(wae: WAE, env: Env): Int = wae match {
  case Add(l, r) => interp(l, env) + interp(r, env)
  ...
```

$$\sigma \vdash e \Rightarrow v$$

$$\frac{\sigma \vdash e_1 \Rightarrow n_1 \quad \sigma \vdash e_2 \Rightarrow n_2}{\sigma \vdash e_1 + e_2 \Rightarrow n_1 + n_2}$$

Operational Semantics

```
// interp : (WAE, Env) => Int
def interp(wae: WAE, env: Env): Int = wae match {
  case Add(l, r) => interp(l, env) + interp(r, env)
  ...
```

$$\sigma \vdash e \Rightarrow v$$

$$\frac{\sigma \vdash e_1 \Rightarrow n_1 \quad \sigma \vdash e_2 \Rightarrow n_2}{\sigma \vdash e_1 + e_2 \Rightarrow n_1 + n_2}$$

Operational Semantics

```
// interp : (WAE, Env) => Int
def interp(wae: WAE, env: Env): Int = wae match {
  case Sub(l, r) => interp(l, env) - interp(r, env)
  ...
```

$$\sigma \vdash e \Rightarrow v$$

$$\frac{\sigma \vdash e_1 \Rightarrow n_1 \quad \sigma \vdash e_2 \Rightarrow n_2}{\sigma \vdash e_1 - e_2 \Rightarrow n_1 - n_2}$$

Operational Semantics

```
// interp : (WAE, Env) => Int
def interp(wae: WAE, env: Env): Int = wae match {
  case Id(x) => lookup(x, env)
  ...
```

$$\sigma \vdash e \Rightarrow v$$

$$\frac{x \in Domain(\sigma)}{\sigma \vdash x \Rightarrow \sigma(x)}$$

Operational Semantics

```
// interp : (WAE, Env) => Int
def interp(wae: WAE, env: Env): Int = wae match {
  case Id(x) => lookup(x, env)
  ...
```

$$\sigma \vdash e \Rightarrow v$$

$$\frac{x \in \mathit{Domain}(\sigma)}{\sigma \vdash x \Rightarrow \sigma(x)}$$

Operational Semantics

```
// interp : (WAE, Env) => Int
def interp(wae: WAE, env: Env): Int = wae match {
  case With(x, i, b) => interp(b, env + (x -> interp(i, env)))
  ...
```

$$\sigma \vdash e \Rightarrow v$$

$$\frac{\sigma \vdash e_1 \Rightarrow v_1 \quad \sigma[x \mapsto v_1] \vdash e_2 \Rightarrow v_2}{\sigma \vdash \texttt{val}\ x = e_1\ \texttt{in}\ e_2 \Rightarrow v_2}$$

Operational Semantics

```
// interp : (WAE, Env) => Int
def interp(wae: WAE, env: Env): Int = wae match {
  case With(x, i, b) => interp(b, env + (x -> interp(i, env)))
  ...
```

$$\sigma \vdash e \Rightarrow v$$

$$\frac{\sigma \vdash e_1 \Rightarrow v_1 \quad \sigma[x \mapsto v_1] \vdash e_2 \Rightarrow v_2}{\sigma \vdash \texttt{val}\ x = e_1\ \texttt{in}\ e_2 \Rightarrow v_2}$$

Sukyoung Ryu
sryu.cs@kaist.ac.kr
http://plrg.kaist.ac.kr