CS320

Syntax and Semantics

Sukyoung Ryu

March 4, 2019

Hands-On Office Hour Tomorrow

March 5th Tuesday
@ E3-1, Room 1501
7:00PM–9:00PM

Bringing your own laptop is recommended.
Slides will be available at the course website.

Don'ts

- Don't import any other libraries.
- Don't use the `break` statement.
- Don't use the `while` and `for` loops.
- Don't use mutation.

Questions and Answers

```scala
// ... : AE => ...
def ...(ae: AE): ... = ae match {
  case Num(n) => ...
  case Add(l, r) => ...
  case Sub(l, r) => ...
}
```

1. Is `Num` a constructor or a type?
2. Is `match` a function?
3. How is `match` different from `if` and `switch`?
4. Why is `ae.left.num` an error?

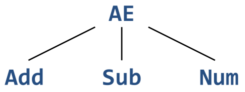
What Is `Num`?

- It is a constructor.

  `Int => AE`

- It is a type.

  `Num <: AE`



`Num(3): Num <: AE`

`Num(3): AE`

Pattern Matching

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

Pattern Matching

```
// interp : AE => Int
def interp(ae: AE): Int =
  if (ae.isInstanceOf[Num])
    ae.asInstanceOf[Num].n
  else if (ae.isInstanceOf[Add]) {
    val add = ae.asInstanceOf[Add]
    interp(add.left) + interp(add.right)
  }
  else {
    val sub = ae.asInstanceOf[Sub]
    interp(sub.left) - interp(sub.right)
  }
interp(ae)
```

Type Checking

```
<AE> ::= <num>
       | {+ <AE> <AE>}
       | {- <AE> <AE>}
```

```scala
trait AE
case class Num(num: Int) extends AE
case class Add(left: AE, right: AE) extends AE
case class Sub(left: AE, right: AE) extends AE

val ae = Add(Num(3), Sub(Num(8), Num(2)))
ae.left.num // error: value num is not a member of AE
```

Programming Language

A *programming language* is defined by

- syntax
  a grammar for programs
- semantics
  rules for evaluating any program to produce a result

Arithmetic Expressions: Concrete Syntax

```
<AE> ::= <num>
       | {+ <AE> <AE>}
       | {- <AE> <AE>}
```

```
{+ 3 {- 8 2}}
```

Arithmetic Expressions: Abstract Syntax

```
<AE> ::= <num>
       | {+ <AE> <AE>}
       | {- <AE> <AE>}
```

```scala
trait AE
case class Num(n: Int) extends AE
case class Add(l: AE, r: AE) extends AE
case class Sub(l: AE, r: AE) extends AE
```

```scala
Add(Num(3), Sub(Num(8), Num(2)))
```

Parser for Arithmetic Expressions

```
// parser for AE
object AE extends RegexParsers {
  lazy val ae: Parser[AE] =
    int                        ^^ { case n     => Num(n)    } |
    wrap("+" ~> ae ~ ae) ^^ { case l ~ r => Add(l, r) } |
    wrap("-" ~> ae ~ ae) ^^ { case l ~ r => Sub(l, r) }
  def apply(str: String): AE =
    parse(ae, str).getOrElse(error(s"bad syntax: $str"))
}
```

- `int`: receiving an integer value
- `str`: receiving a string value
- `~`: concatenation of rules
- `~>`: without receiving the value of the left-hand side
- `wrap`: enclosed by { and }

Parser for Arithmetic Expressions

```
// parser for AE
object AE extends RegexParsers {
  lazy val ae: Parser[AE] =
    int                      ^^ { case n     => Num(n)    } |
    wrap("+" ~> ae ~ ae) ^^ { case l ~ r => Add(l, r) } |
    wrap("-" ~> ae ~ ae) ^^ { case l ~ r => Sub(l, r) }
  def apply(str: String): AE =
    parse(ae, str).getOrElse(error(s"bad syntax: $str"))
}
```

- `int`: receiving an integer value
- `str`: receiving a string value
- `~`: concatenation of rules
- `~>`: without receiving the value of the left-hand side
- `wrap`: enclosed by { and }

Parser for Arithmetic Expressions

```
// parser for AE
object AE extends RegexParsers {
  lazy val ae: Parser[AE] =
    int                       ^^ { case n     => Num(n)    } |
    wrap("+" ~> ae ~ ae) ^^ { case l ~ r => Add(l, r) } |
    wrap("-" ~> ae ~ ae) ^^ { case l ~ r => Sub(l, r) }
  def apply(str: String): AE =
    parse(ae, str).getOrElse(error(s"bad syntax: $str"))
}
```

```
AE("3")
AE("{+ 3 4}")
AE("{+ {- 3 4} 7}")
AE("{- 5 1 2}")
```

Parser for Arithmetic Expressions

```
// parser for AE
object AE extends RegexParsers {
  lazy val ae: Parser[AE] =
    int                       ^^ { case n     => Num(n)    } |
    wrap("+" ~> ae ~ ae) ^^ { case l ~ r => Add(l, r) } |
    wrap("-" ~> ae ~ ae) ^^ { case l ~ r => Sub(l, r) }
  def apply(str: String): AE =
    parse(ae, str).getOrElse(error(s"bad syntax: $str"))
}
```

```
AE("3")
AE("{+ 3 4}")
AE("{+ {- 3 4} 7}")
AE("{- 5 1 2}")
```

Parser for Arithmetic Expressions

```
// parser for AE
object AE extends RegexParsers {
  lazy val ae: Parser[AE] =
    int                    ^^ { case n     => Num(n)    } |
    wrap("+" ~> ae ~ ae) ^^ { case l ~ r => Add(l, r) } |
    wrap("-" ~> ae ~ ae) ^^ { case l ~ r => Sub(l, r) }
  def apply(str: String): AE =
    parse(ae, str).getOrElse(error(s"bad syntax: $str"))
}
```

```
Num(3)
Add(Num(3), Num(4))
Add(Sub(Num(3), Num(4)), Num(7))
Sub(Num(5), Num(1), Num(2))
```

Interpreter for Arithmetic Expressions

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

Interpreter for Arithmetic Expressions

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

test(interp(AE("3")), 3)
test(interp(AE("{+ 3 4}")), 7)
test(interp(AE("{+ {- 3 4} 7}")), 6)
```

Operational Semantics

- A method of defining the meaning of programs by describing the actions carried out during a program's execution.
- Many different styles
  - Evaluator semantics
  - Natural semantics, big-step
  - SOS semantics, small-step
  - Reduction semantics
  - Abstract machine semantics

Courtesy by David Van Horn
https://www.youtube.com/watch?v=TU16mA5-i-g

Operational Semantics for What?

- Specifying a programming language
- Communicating language design ideas
- Validating claims about languages
- Validating claims about type systems, etc
- Proving correctness of a compiler
- $\cdots$

AE: Concrete Syntax

```
<AE> ::= <num>
       | {+ <AE> <AE>}
       | {- <AE> <AE>}
```

AE: Abstract Syntax

```
trait AE
case class Num(n: Int) extends AE
case class Add(l: AE, r: AE) extends AE
case class Sub(l: AE, r: AE) extends AE
```

Syntax of AE (1)

$$
\begin{array}{rcl}
n \in \mathbb{Z} & \Rightarrow & n \in \mathcal{A} \\
e_1 \in \mathcal{A} \land e_2 \in \mathcal{A} & \Rightarrow & \{+\ e_1\ e_2\} \in \mathcal{A} \\
e_1 \in \mathcal{A} \land e_2 \in \mathcal{A} & \Rightarrow & \{-\ e_1\ e_2\} \in \mathcal{A}
\end{array}
$$

Syntax of AE (2)

$$\frac{n \in \mathbb{Z}}{n \in \mathcal{A}}$$

$$\frac{e_1 \in \mathcal{A} \qquad e_2 \in \mathcal{A}}{\{+\ e_1\ e_2\} \in \mathcal{A}}$$

$$\frac{e_1 \in \mathcal{A} \qquad e_2 \in \mathcal{A}}{\{-\ e_1\ e_2\} \in \mathcal{A}}$$

Syntax of `AE` (2)

Inference rules

$$\frac{H_1 \qquad H_2 \qquad \cdots \qquad H_n}{C}$$

Syntax of `AE` (2)

Inference rules

$$\frac{H_1 \qquad H_2 \qquad \cdots \qquad H_n}{C}$$

$$H_1 \wedge H_2 \wedge \cdots \wedge H_n \Rightarrow C$$

Syntax of `AE` (2)

Proof of $\{+\ 4\ \{-\ 2\ 1\}\} \in \mathcal{A}$

$$\frac{n \in \mathbb{Z}}{n \in \mathcal{A}} \qquad \frac{e_1 \in \mathcal{A} \qquad e_2 \in \mathcal{A}}{\{+\ e_1\ e_2\} \in \mathcal{A}} \qquad \frac{e_1 \in \mathcal{A} \qquad e_2 \in \mathcal{A}}{\{-\ e_1\ e_2\} \in \mathcal{A}}$$

Syntax of `AE` (2)

Proof of $\{+\ 4\ \{-\ 2\ 1\}\} \in \mathcal{A}$

$$\frac{n \in \mathbb{Z}}{n \in \mathcal{A}} \qquad \frac{e_1 \in \mathcal{A} \qquad e_2 \in \mathcal{A}}{\{+\ e_1\ e_2\} \in \mathcal{A}} \qquad \frac{e_1 \in \mathcal{A} \qquad e_2 \in \mathcal{A}}{\{-\ e_1\ e_2\} \in \mathcal{A}}$$

$$\dfrac{\dfrac{4 \in \mathbb{Z}}{4 \in \mathcal{A}} \qquad \dfrac{\dfrac{2 \in \mathbb{Z}}{2 \in \mathcal{A}} \qquad \dfrac{1 \in \mathbb{Z}}{1 \in \mathcal{A}}}{\{-\ 2\ 1\} \in \mathcal{A}}}{\{+\ 4\ \{-\ 2\ 1\}\} \in \mathcal{A}}$$

Syntax of AE (3)

$$
\begin{array}{llll}
\mathbb{Z} & n & ::= & \cdots \mid -1 \mid 0 \mid 1 \mid \cdots \\
\mathcal{A} & e & ::= & n \\
 & & \mid & \{\texttt{+}\ e\ e\} \\
 & & \mid & \{\texttt{-}\ e\ e\}
\end{array}
$$

Evaluator Semantics of AE

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

Natural Semantics of $\texttt{AE}$

$$\Rightarrow \subseteq \mathcal{A} \times \mathbb{Z}$$

$$\vdash \ \texttt{\{+ 4 \{- 2 1\}\}} \Rightarrow \texttt{5}$$

Natural Semantics of AE

$$\vdash \; n \Rightarrow n$$

$$\frac{\vdash \; e_1 \Rightarrow n_1 \qquad \vdash \; e_2 \Rightarrow n_2}{\vdash \; \{\texttt{+}\; e_1 \; e_2\} \Rightarrow n_1 + n_2}$$

$$\frac{\vdash \; e_1 \Rightarrow n_1 \qquad \vdash \; e_2 \Rightarrow n_2}{\vdash \; \{\texttt{-}\; e_1 \; e_2\} \Rightarrow n_1 - n_2}$$

Natural Semantics of AE

$$\frac{\vdash \; e_1 \Rightarrow n_1 \qquad \vdash \; e_2 \Rightarrow n_2}{\vdash \; \{+ \; e_1 \; e_2\} \Rightarrow n_1 + n_2}$$

$$\frac{\vdash \; e_1 \Rightarrow n_1 \qquad \vdash \; e_2 \Rightarrow n_2 \qquad n_3 = n_1 + n_2}{\vdash \; \{+ \; e_1 \; e_2\} \Rightarrow n_3}$$

Natural Semantics of AE

Proof of $\vdash$ `{+ 4 {- 2 1}}` $\Rightarrow 5$

$$\vdash\ n \Rightarrow n \qquad \frac{\vdash\ e_1 \Rightarrow n_1 \qquad \vdash\ e_2 \Rightarrow n_2}{\vdash\ \{\texttt{+}\ e_1\ e_2\} \Rightarrow n_1 + n_2} \qquad \frac{\vdash\ e_1 \Rightarrow n_1 \qquad \vdash\ e_2 \Rightarrow n_2}{\vdash\ \{\texttt{-}\ e_1\ e_2\} \Rightarrow n_1 - n_2}$$

Natural Semantics of `AE`

Proof of $\vdash$ {+ 4 {- 2 1}} $\Rightarrow 5$

$$\vdash n \Rightarrow n \qquad \frac{\vdash e_1 \Rightarrow n_1 \qquad \vdash e_2 \Rightarrow n_2}{\vdash \{+\ e_1\ e_2\} \Rightarrow n_1 + n_2} \qquad \frac{\vdash e_1 \Rightarrow n_1 \qquad \vdash e_2 \Rightarrow n_2}{\vdash \{-\ e_1\ e_2\} \Rightarrow n_1 - n_2}$$

$$\frac{\vdash 4 \Rightarrow 4 \qquad \dfrac{\vdash 2 \Rightarrow 2 \qquad \vdash 1 \Rightarrow 1}{\vdash \{-\ 2\ 1\} \Rightarrow 1}}{\vdash \{+\ 4\ \{-\ 2\ 1\}\} \Rightarrow 5}$$

Natural Semantics of AE

Proof of $\vdash$ {+ {- 3 4} 7 } $\Rightarrow$?

$$\vdash\ n \Rightarrow n \qquad \frac{\vdash\ e_1 \Rightarrow n_1 \qquad \vdash\ e_2 \Rightarrow n_2}{\vdash\ \{+\ e_1\ e_2\} \Rightarrow n_1 + n_2} \qquad \frac{\vdash\ e_1 \Rightarrow n_1 \qquad \vdash\ e_2 \Rightarrow n_2}{\vdash\ \{-\ e_1\ e_2\} \Rightarrow n_1 - n_2}$$

Natural Semantics of AE

Proof of $\vdash$ {- 5 1 2} $\Rightarrow$?

$$\vdash\ n \Rightarrow n \qquad \frac{\vdash\ e_1 \Rightarrow n_1 \qquad \vdash\ e_2 \Rightarrow n_2}{\vdash\ \{+\ e_1\ e_2\} \Rightarrow n_1 + n_2} \qquad \frac{\vdash\ e_1 \Rightarrow n_1 \qquad \vdash\ e_2 \Rightarrow n_2}{\vdash\ \{-\ e_1\ e_2\} \Rightarrow n_1 - n_2}$$

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