Teaching Concepts of Programming Languages in Scala with WebLab

Teaching Approaches

There are various ways to teach programming language concepts.

- Study real world languages
- Study formal semantics
- Implement definitional interpreters

Concepts of Programming Languages

Every week students learn new programming language concepts, and implement them in Scala as part of a Scheme-like language.

WebLab

CONCEPTS OF PROGRAMMING LANGUAGES / ASSIGNMENT 1.04.0 1 2 3 4

> Basic Interpreter ✔ Completed

Example

<table>
<thead>
<tr>
<th>Problem</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>(not true)</td>
<td>def parse(sexp: SExp) = match { case Num(n) = NumExt(n) case Sym(&quot;false&quot;) = FalseExt() case SList(list: List[SExp]) =&gt; IfExt(parse(c), parse(t), parse(e)) case SNum(n) = NumExt(n) }</td>
</tr>
<tr>
<td>(not 2)</td>
<td>test(&quot;test 2&quot;)</td>
</tr>
<tr>
<td>(not (not true))</td>
<td>test(&quot;test (not operator)&quot;)</td>
</tr>
<tr>
<td>FalseC(), TrueExt()</td>
<td>interp(desugar(UnOpExt(&quot;not&quot;, TrueExt())))</td>
</tr>
<tr>
<td>UnOpExt(&quot;not&quot;, TrueExt())</td>
<td>interp(desugar(UnOpExt(&quot;not&quot;, TrueExt())))</td>
</tr>
<tr>
<td>BoolV(false)</td>
<td>interp(desugar(parse(&quot;false&quot;)))</td>
</tr>
</tbody>
</table>

Reference Implementation

```scala
trait NumInterpreter extends Interpreter {
  override def interp(e: ExprC) = e match {
    case Num(n) = NumExt(n)
    case PlusC(l, r) = NumV(interp(l) + interp(r))
    case _ = super.interp(e)
  }
}
```

Our reference implementation uses Scala’s mixin trait composition to override and extend each week’s implementation, for high reuse and low maintenance.

Test Definition DSL

```scala
Cluster("not operator", List(
  Pos("not true"),
  Pos("not false"),
  Neg("not true"),
  Neg("not false")
))
```

Our DSL in Scala allows us to generate new test cases efficiently.

Specification Tests

```scala
class TestSpec extends FunSuite {
  test("not operator") { 
    expect(NumExt("not", TrueExt()))
    interp("not true")
  }
}
```

The generated specification test suite is used for grading, but students use it also to get immediate feedback on their progress.

Specification Meta-tests

```scala
test("Students should test for duplicate let-binding") { 
  testParser(new Week3Interpreter {
    override def checkDelegateBinding[vars: List[String]] = { 
      // Do not check for duplicates
    }
  })
}
```

To test the student’s test suite, we override a single aspect of our reference implementation to be faulty, and verify that the student’s test suite catches this.

Test Generator

The test generator generates a ScalaTest test suite from our test definition DSL, asserting that the student implementation produces the same results as our reference implementation.

Shriram Krishnamurthi, and Joe Gibbons Politz.
“Programming and Programming Languages.”
http://papl.cs.brown.edu/2015/

Tim van der Lippe, Thomas Smith, Daniël Pelsmaeker, and Eelco Visser.
“*A Scalable Infrastructure for Teaching Concepts of Programming Languages in Scala with WebLab.*”