An OOPS Compiler Written in Scala

Martin Ring

Studiengang Informatik
Universität Bremen
October 18, 2011

This document describes a compiler for the OOPS programming language. OOPS is a 'fictional' language introduced in the course "Übersetzer" (Compiler) at the University of Bremen. No language specification exists, but the language is defined through an existing reference compiler written in Java.

In the course students are required to extend the reference compiler with several features. The scala based compiler offers an alternative base, on which these features can be implemented, which utilizes functional concepts like parser combinators, algebraic data structures and transformation monads.

1 Lexical Analysis

Because we want to use the parser combinators included in the standard Scala library we need to build a lexer that is compatible with the parser combinators. Luckyly this is very easy. First we define our tokens:

```scala
/*
 * Defines the tokens that are distinguished in the OOPS language.
 */
trait OOPSTokens extends Tokens {
  /* Represents a keyword (Reserved word or delimiter) */
  case class Keyword(chars: String) extends Token  {
    override def toString = Lexical.delimiters(chars)
  }
  /* Represents a number literal */
  case class Number(chars: String) extends Token {
    require(!chars.exists(x => !('0' to '9').contains(x)))
    val value = chars.toInt
    override def toString = "NUMBER " + chars
  }
  /* Represents an identifier */
  case class Identifier(chars: String) extends Token {
    override def toString = "IDENTIFIER " + chars
  }
}
```

Listing 1: Tokens.scala

2 Syntactical Analysis

3 Context Analysis

4 Code Generation