Compilers - Assignment 2 - Parser

Ayush Sekhari (12185)
Vasu Sharma (12785)
Margaux Dorido

March 12, 2015

1 Design Specifications

We have the following language specifications for our Lexer:
Source Language: ADA
Target Language: MIPS
Implementation Language: Python
Pythons PLY library was used to build the Lexer and Parser.

2 Building the Lexer

The Lexer has been implemented in python and can be executed as:

1. Change directory to the directory containing lexer.py
2. python ./lexer.py ada_program_name.adb example: python ./lexer.py test1.adb

2.1 Lexer Constraints

Our Lexer doesnt handle the following cases:

1. Operator overloading is disallowed by our Lexer due to ambiguity in classi- fying the redefined operator as string or identifier as operator overloading requires the operator to be put in double quotes which makes it impossible to distinguish it from other strings.
2. Our Lexer doesnt allow user to use reserved types as identifier names for example I cant have an identifier named String or Integer etc. This has been done as it creates the ambiguity as whether to identify as a data type or an identifier.
3. Ada has some non-standard data types too which are defined within certain libraries. We are not creating separate token names for these non- standard data types. We have however handled standard data types like Integer, String etc.
4. In the representation of numbers in the exponential form, our Lexer returns the value of a number in exponential form in the normal numeric decimal representation. This leads to overflow errors in case the exponential number is too large. example 19E200 is out of bounds of any data type in Ada and hence causes an overflow causing a garbage value to be returned. We havent handled such issues.

3 Building the Parser

They have been commented out in the file `reserved_tokens.py`

The Parser has been implemented in python and can be executed as:

1. Change directory to the directory containing parser.py
2. ./parser.py ada_program_name.adb example: ./parser.py test1.adb
3. A few test files have been listed in the test folder
4. The parse tree can be generated using the following command
   `dot -Tpng file_name.dot -o Image_name.png`

3.1 Parser Contraints

1. We have removed the token Ada_Keyword from the lexer, We would ensure that this token newly identified as an IDENTIFIER has the value ”ada” in the semantic analysis stage.
2. We needed to remove a few token from our lexer definitions which were difficult to handle as we were not certain of their applications and meanings while building the parser and writing the language grammar. Most of them were attributes.
3. We have removed tokens for the INTEGER_TYPE and FLOAT_TYPE. They are identified as IDENTIFIERS for now.