Ciências ULisboa | Informática

Compiler Techniques 2024/25
AGUDA M2: Printing the AST
Deliver by April 14, 2025

For the second milestone (M2) of the compiler we shall print the abstract syntax tree (AST) of syntactically correct AGUDA programs, and issue an error when the source is not syntactically correct.

M2 is delivered as a docker container, so that I may run your code on my machine, regardless of the programming language and the versions of the software you use.

You are expected to

- Write a parser for the AGUDA programming language
- Write a compiler that reads a source file (a `.agu` file) from the command line and prints the AST in indented form (see example below).
- Write a short how-to report . Possible formats: `md`, `txt`, `pdf`.

Requirements:

- You compiler should pass as many tests as possible. Tests are taken from `https://git.alunos.di.fc.ul.pt/tcomp000/aguda-testing`.
- If you use LR technology then:
  - The parser should contain no reduce-reduce conflicts
  - A small number of shift-reduce conflicts are tolerated. Use judiciously the `left`, `right`, `nonassoc` clauses, for operators, symbols and keywords. In addition, factor your grammar if needed
- Errors: your compiler must clearly distinguish lexical from syntactic errors. In addition it must print the line and column of each error

The how-to report should include:

- How to build your compiler
- How to run a particular test
- How to run the whole test suite (valid and invalid programs)
- How to interpret the testing output (how many tests passed, which failed)
- If your parser does not pass all tests, explain why
- Name you report `aguda-M2.md` (similarly if you chose a different format); place it in the top folder of your deliverable

For example, given the following source code:

```
                        let
-- What a nice program
                                            x :



Int -- An equals sign follows
          =
                                           while true


             do unit
      ;2
```

your compiler should print something similar to

```
let x : Int =
  while true do
    unit ;
  2
```

Note on the textual representation of the AST:

- There are no precise rules for the format of the output; in particular it need *not* be a syntactically correct AGUDA program
- The textual representation may not preserve the precedence of arithmetic operations (it is perfectly fine to print arithmetic expressions without parenthesis)
- But, the code after each top level declaration (**`let`**) should appear in a new line, indented. The same applies to **`while`**, **`if`**-**`then`** and **`if`**-**`then`**-**`else`**. Furthermore, in `exp1 ; exp2`, the two expressions should appear in separated lines