## PROGRAMMING ENVIRONMENT

## 8.1 CABAL, STACK AND HADDOCK

The Mikrokosmos documentation as a Haskell library is included in its own code. It uses **Haddock**, a tool that generates documentation from annotated Haskell code; it is the *de facto* standard for Haskell software.

Dependencies and packaging details for Mikrokosmos are specified in a file distributed with the source code called mikrokosmos.cabal. It is used by the package managers **stack** and **cabal** to provide the necessary libraries even if they are not available systemwide. The **stack** tool is also used to package the software, which is uploaded to *Hackage*.

## 8.2 TESTING

**Tasty** is the Haskell testing framework of our choice for this project. It allows the user to create a comprehensive test suite combining multiple types of tests. The Mikrokosmos code is tested using the following techniques

- **unit tests**, in which individual core functions are tested independently of the rest of the application;
- property-based testing, in which multiple test cases are created automatically in order to verfiy that a specified property always holds;
- **golden tests**, a special case of unit tests in which the expected results of an IO action, as described on a file, are checked to match the actual ones.

We are using the **HUnit** library for unit tests. It tests particular cases of type inference, unification and parsing. The following is an example of unit test, as found in tests.hs. It checks that the type inference of the identity term is correct.

```
-- Checks that the type of λx.x is exactly A → A
testCase "Identity type inference" $
  typeinference (Lambda (Var 1)) @?= Just (Arrow (Tvar 0) (Tvar 0))
```

We are using the **QuickCheck** library for property-based tests. It tests transformation properties of lambda expressions. In the following example, it tests that any De Bruijn expression keeps its meaning when translated into a  $\lambda$ -term.

```
-- Tests if translation preserves meaning

QC.testProperty "Expression -> named -> expression" $

\expr -> toBruijn emptyContext (nameExp expr) == expr
```

We are using the **tasty-golden** package for golden tests. Mikrokosmos can be passed a file as an argument to interpret it and show only the results. This feature is used to create a golden test in which the interpreter is asked to provide the correct interpretation of a given file. This file is called testing.mkr, and contains library definitions and multiple tests. Its expected output is testing.golden. For example, the following Mikrokosmos code can be found on the file

```
:types on
caseof (inr 3) (plus 2) (mult 2)
```

and the expected output is

```
-- types: on 
-- \lambda a.\lambda b.(a (a (a (a (a b)))))) \Rightarrow 6 :: (A \rightarrow A) \rightarrow A \rightarrow A
```

## 8.3 VERSION CONTROL AND CONTINUOUS INTEGRATION

Mikrokosmos uses **git** as its version control system and the code, which is licensed under GPLv<sub>3</sub>, can be publicly accessed on the following GitHub repository:

```
https://github.com/M42/mikrokosmos
```

Development takes place on the development git branch and permanent changes are released into the master branch. Some more minor repositories have been used in the development; they directly depend on the main one

```
    https://github.com/m42/mikrokosmos-js
    https://github.com/M42/ivputos_mikroko
```

- https://github.com/M42/jupyter-mikrokosmos
- https://github.com/M42/mikrokosmos-lib

The code uses the **Travis CI** continuous integration system to run tests and check that the software builds correctly after each change and in a reproducible way on a fresh Linux installation provided by the service.