7

USAGE

7.1 INSTALLATION

The complete Mikrokosmos suite is divided in multiple parts:

1. the **Mikrokosmos interpreter**, written in Haskell;
2. the **Jupyter kernel**, written in Python;
3. the **CodeMirror Lexer**, written in Javascript;
4. the **Mikrokosmos libraries**, written in the Mikrokosmos language;
5. the **Mikrokosmos-js** compilation, which can be used in web browsers.

These parts will be detailed on the following sections. A system that already satisfies all dependencies (Stack, Pip and Jupyter), can install Mikrokosmos using the following script, which is detailed on this section

```
# Mikrokosmos interpreter
stack install mikrokosmos
# Jupyter kernel for Mikrokosmos
sudo pip install imikrokosmos
# Libraries
git clone https://github.com/M42/mikrokosmos-lib.git ~/.mikrokosmos
```

The **Mikrokosmos interpreter** is listed in the central Haskell package archive, *Hackage* [1]. The packaging of Mikrokosmos has been done using the **cabal** tool; and the configuration of the package can be read in the file `mikrokosmos.cabal` on the Mikrokosmos code. As a result, Mikrokosmos can be installed using the **cabal** and **stack** Haskell package managers. That is,

```
# With cabal
cabal install mikrokosmos
```

1 : Hackage can be accesed in: http://hackage.haskell.org/ and the Mikrokosmos package can be found in https://hackage.haskell.org/package/mikrokosmos

```
# With stack
stack install mikrokosmos
```

The **Mikrokosmos Jupyter kernel** is listed in the central Python package archive. Jupyter is a dependency of this kernel, which only can be used in conjunction with it. It can be installed with the `pip` package manager as

```
sudo pip install imikrokosmos
```

and the installation can be checked by listing the available Jupyter kernels with

```
jupyter kernelspec list
```

The **Mikrokosmos libraries** can be downloaded directly from its GitHub repository. [2] They have to be placed under `~/.mikrokosmos` if we want them to be locally available or under `/usr/lib/mikrokosmos` if we want them to be globally available.

```
git clone https://github.com/M42/mikrokosmos-lib.git ~/.mikrokosmos
```

The following script installs the complete Mikrokosmos suite on a fresh system. It has been tested under `Ubuntu 16.04.3 LTS (Xenial Xerus)`.

```
# 1. Installs Stack, the Haskell package manager
wget -qO- https://get.haskellstack.org | sh
STACK=$(which stack)

# 2. Installs the ncurses library, used by the console interface
sudo apt install libncurses5-dev libncursesw5-dev

# 3. Installs the Mikrokosmos interpreter using Stack
$STACK setup
$STACK install mikrokosmos

# 4. Installs the Mikrokosmos standard libraries
sudo apt install git
git clone https://github.com/M42/mikrokosmos-lib.git ~/.mikrokosmos

# 5. Installs the IMikrokosmos kernel for Jupyter
sudo apt install python3-pip
sudo -H pip install --upgrade pip
sudo -H pip install jupyter
sudo -H pip install imikrokosmos
```

2 : The repository can be accessed in: https://github.com/M42/mikrokosmos-lib.git

7.2 MIKROKOSMOS INTERPRETER

Once installed, the Mikrokosmos λ interpreter can be opened from the terminal with the `mikrokosmos` command. It will enter a *read-eval-print loop* where λ-expressions and interpreter commands can be evaluated.

```
$> mikrokosmos
Welcome to the Mikrokosmos Lambda Interpreter!
Version 0.5.0. GNU General Public License Version 3.
mikro> _
```

The interpreter evaluates every line as a lambda expression. Examples on the use of the interpreter can be read on the following sections. Apart from the evaluation of expressions, the interpreter accepts the following commands

- `:quit` and `:restart`, stop the interpreter;
- `:verbose` activates *verbose mode*;
- `:ski` activates *SKI mode*;
- `:types` changes between untyped and simply typed λ-calculus;
- `:color` deactivates colored output;
- `:load` loads a library.

Figure 2 is an example session on the mikrokosmos interpreter.

7.3 JUPYTER KERNEL

The **Jupyter Project** [Jup] is an open source project providing support for interactive scientific computing. Specifically, the Jupyter Notebook provides a web application for creating interactive documents with live code and visualizations.

We have developed a Mikrokosmos kernel for the Jupyter Notebook, allowing the user to write and execute arbitrary Mikrokosmos code on this web application. An example session can be seen on Figure 3.

The implementation is based on the `pexpect` library for Python. It allows direct interaction with any REPL and collects its results. Specifically, the following Python lines represent the central idea of this implementation

```
# Initialization
mikro = pexpect.spawn('mikrokosmos')
mikro.expect('mikro>')

# Interpreter interaction
# Multiple-line support
output = ""
```

Figure 2: Mikrokosmos interpreter session.

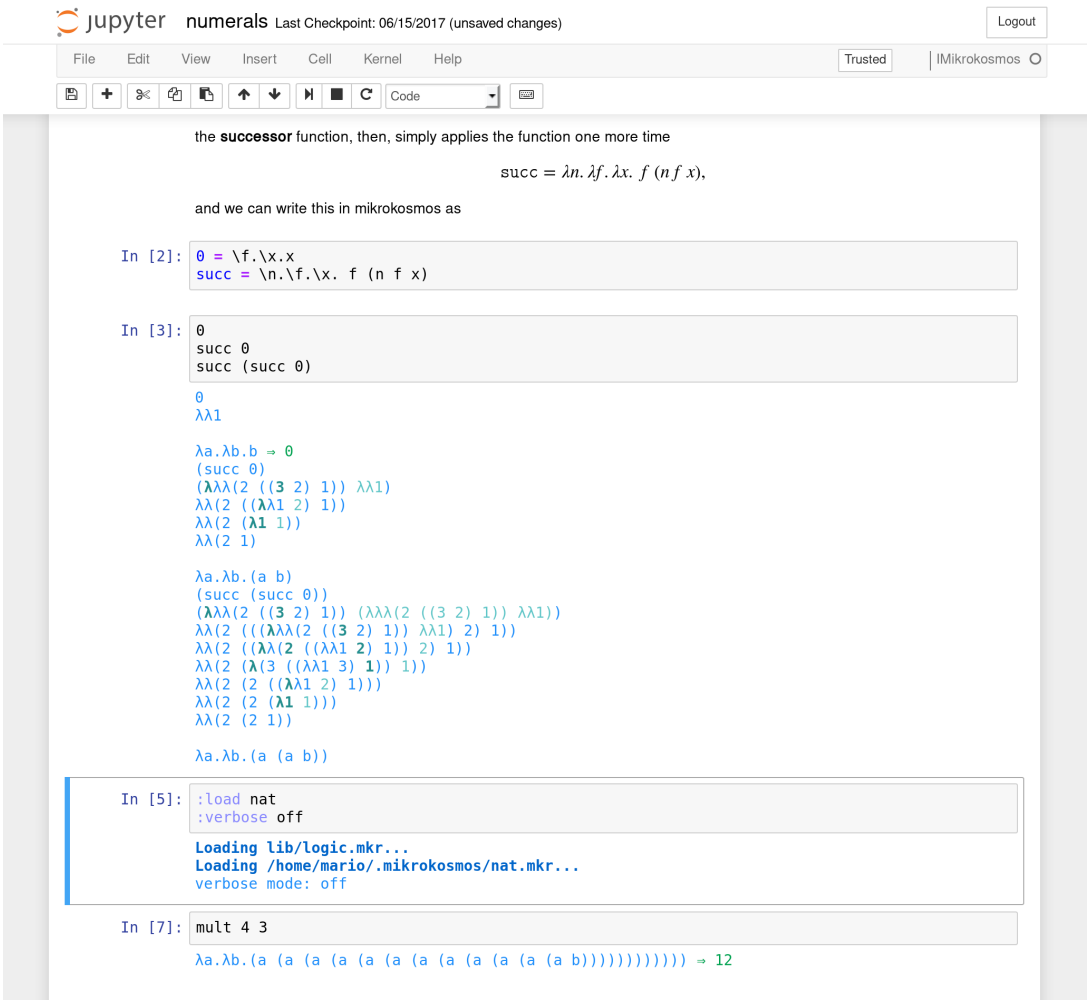

Figure 3: Jupyter notebook Mikrokosmos session.

```
for line in code.split('\n'):
    # Send code to mikrokosmos
    self.mikro.sendline(line)
    self.mikro.expect('mikro> ')

    # Receive and filter output from mikrokosmos
    partialoutput = self.mikro.before
    partialoutput = partialoutput.decode('utf8')
    output = output + partialoutput
```

A `pip` installable package has been created following the Python Packaging Authority guidelines. [3] This allows the kernel to be installed directly using the `pip` python package manager.

```
sudo -H pip install imikrokosmos
```

7.4 CODEMIRROR LEXER

CodeMirror [4] is a text editor for the browser implemented in Javascript. It is used internally by the Jupyter Notebook.

A CodeMirror lexer for Mikrokosmos has been written. It uses Javascript regular expressions and signals the ocurrence of any kind of operator to CodeMirror. It enables syntax highlighting for Mikrokosmos code on Jupyter Notebooks. It comes bundled with the kernel specification and no additional installation is required.

```
CodeMirror.defineSimpleMode("mikrokosmos", {
    start: [
    // Comments
    {regex: /\#.*/,
    token: "comment"},
    // Interpreter
    {regex: /\:load|\:verbose|\:ski|\:restart|\:types|\:color/,
    token: "atom"},
    // Binding
    {regex: /(.*?)(\s*)(=)(\s*)(.*?)$/,
    token: ["def",null,"operator",null,"variable"]},
    // Operators
    {regex: /[=!]+/,
    token: "operator"},
    ],
```

3 : The PyPA packaging user guide can be found in its official page: https://packaging.python.org/
4 : Documentation for CodeMirror can be found in its official page: https://codemirror.net/

```
    meta: {
        dontIndentStates: ["comment"],
        lineComment: "#"
    }
}
```

7.5 JUPYTERHUB

JupyterHub manages multiple instances of independent single-user Jupyter notebooks. We used it to serve Mikrokosmos notebooks and tutorials to students studying λ-calculus.

In order to install Mikrokosmos on a server and use it as `root` user, we need

- to clone the libraries into `/usr/lib/mikrokosmos`. They should be available system-wide.
- to install the Mikrokosmos interpreter into `/usr/local/bin`. In this case, we chose not to install Mikrokosmos from source, but simply copy the binaries and check the availability of the `ncurses` library.
- to install the Mikrokosmos Jupyter kernel as usual.

Our server used a SSL certificate; and OAuth autentication via GitHub. Mikrokosmos tutorials were installed for every student.

7.6 CALLING MIKROKOSMOS FROM JAVASCRIPT

The GHCjs[5] compiler allows transpiling from Haskell to Javascript. Its foreign function interface allows a Haskell function to be passed as a continuation to a Javascript function.

A particular version of the `Main.hs` module of Mikrokosmos was written in order to provide a `mikrokosmos` function, callable from Javascript. This version includes the standard libraries automatically and reads blocks of texts as independent Mikrokosmos commands. The relevant use of the foreign function interface is showed in the following code

```
foreign import javascript unsafe "mikrokosmos = $1"
    set_mikrokosmos :: Callback a -> IO ()
```

which provides `mikrokosmos` as a Javascript function once the code is transpiled. In particular, the following is an example of how to call Mikrokosmos from Javascript

5 : The GHCjs documentation is available on its web page https://github.com/ghcjs/ghcjs

```
button.onclick = function () {
   editor.save();
   outputcode.getDoc().setValue(mikrokosmos(inputarea.value).mkroutput);
   textAreaAdjust(outputarea);
}
```

A small script has been written in Javascript to help with the task of embedding Mikrokosmos into a web page. It and can be included directly from

`https://m42.github.io/mikrokosmos-js/mikrobox.js`

using GitHub as a CDN. It will convert any HTML script tag written as follows

```
<div class="mikrojs-console">
<script type="text/mikrokosmos">
(λx.x)
... your code
</script>
</div>
```

into a CodeMirror pad where Mikrokosmos can be executed. The Mikrokosmos tutorials are an example of this feature and can be seen on Figure 4.

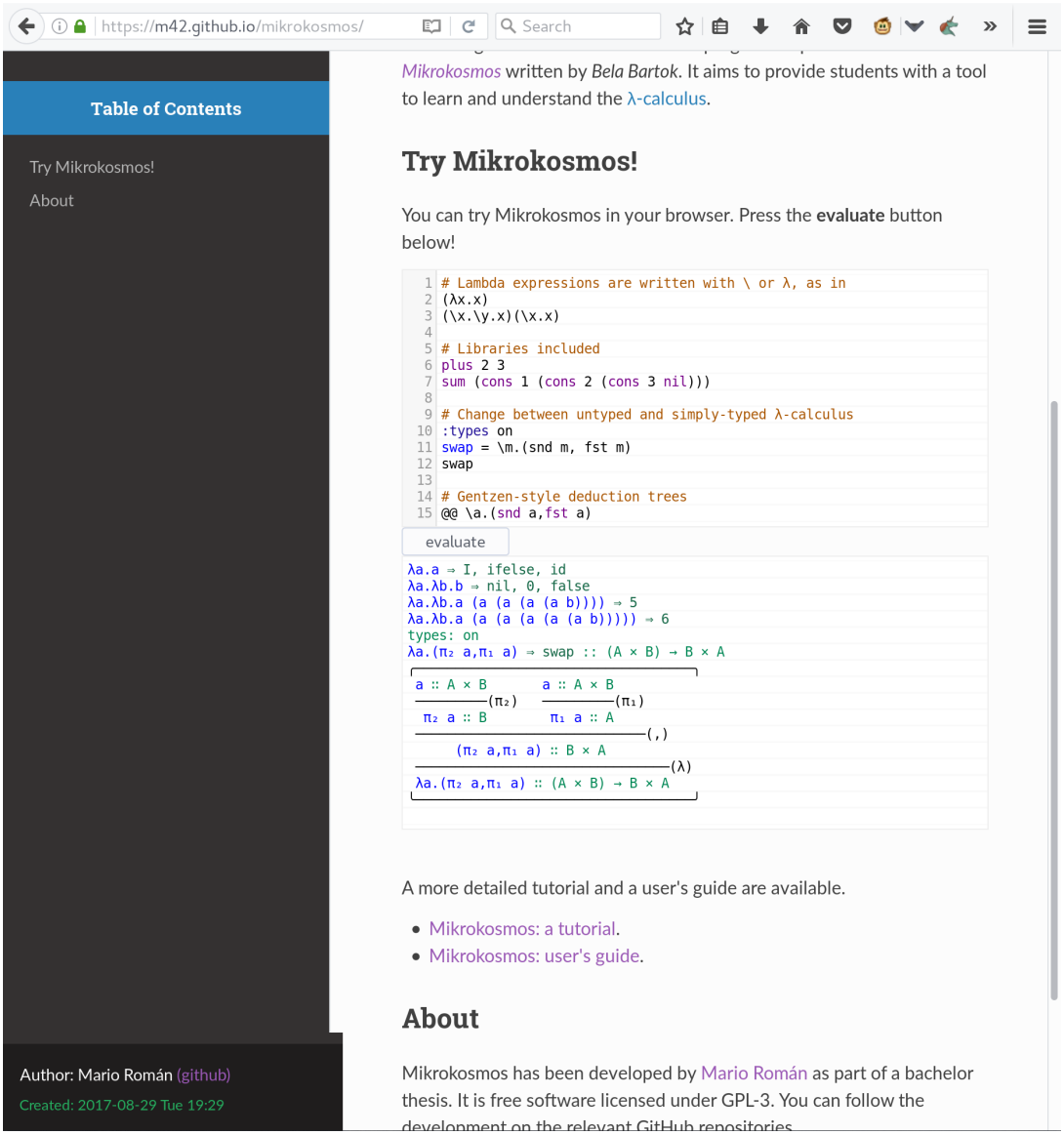

Figure 4: Mikrokosmos embedded into a web page.