User Documentation

LAUNCHING THE COMPILER

1. **Clone the project Repository**

   ```
   git clone https://github.com/nhman2002/Compil-Projet.git
   ```

2. **Compile the project**

make sure that you already in the correct directory

```
cd archive/archive/java
```

Go to jflex directory. In here we will have a mini project which helps us to generate the lexer(Lexer.java in java directory)

```
cd jflex
mvn clean install
```

Went back to the java directory

```
cd ..
make
```

Running the Compiler and Test Files

1. Running Test Files and Generating `.ml` to `.asml` Files

The project includes several test files located in the `tests` directory. These tests cover some functionalities such as type-checking, .asml generation, syntax validation, and code generation.

To execute test files, you can follow these steps:

1. Navigate to the `java` directory Before running commands, change to the java directory for easier access to the mincamlc executable: `cd java`
2. Execute test files

Use the following command to `.ml` file:

```
./mincamlc [../fileLocation/filename.ml]
```

Examples:

```
./mincamlc ../tests/asmlcheck/fib.ml
./mincamlc ../tests/gen-code/ack.ml
```

2. Generating ARM Code from the `.asml` Files:

The project will generate `.s` files from the `.ml` files and write the results to the specified output file

```
cd java
make
./mincamlc -i filepath/input.ml -o filepath/output.txt
```

Example:

```
./mincamlc -i ../tests/typechecking/valid/let_seq.ml -o ../tests/typechecking/valid/let_seq.
```

The above command will produce the ARM equivalent of the specified `.asml` file.

3. Print out the ASML code: The project will print out the ASML code of the input file to the console.

```
./mincamlc -a filepath/input.ml
```

Example:

```
./mincamlc -a ../tests/typechecking/valid/let_seq.ml
```

4. Generating ARM Code from the `.ml` Files:

To generate ARM assembly code from `.ml` files and write `.asml` to ouput file:

```
./mincamlc -i filepath/input.ml -o filepath/output.txt
```

Example

```
./mincamlc -i ../tests/typechecking/valid/let_seq.ml -o ../tests/typechecking/valid/let_seq.
```

4. Run test suite:

```
cd java
make test
```

Developer Documentation

Organization of the archive

```
ARM/     arm source example and compilation with libmincaml
asml/    asml examples
doc/     all the documentation, start with index.hml
java/    MinCaml parser in Java + examples of tree traversal algo, if you do the project in
mincaml/ MinCaml examples
ocaml/   MinCaml parser in OCaml, if you do the project in OCaml
scripts/ put your test scripts and symbolic links there, and add this directory to your path
tests/   put your tests there
tools/   asml intepreter (linux binary)
```

We recommend that you add scripts/ and the dir that contains mincamlc to your PATH.

Dependencies

java-getopt-1.0.13.jar

Make sure that you have java-getopt-1.0.13.jar in your java directory. You can download it from here. The java-getopt-1.0.13.jar file is a Java library that provides functionality for parsing command-line arguments in a manner similar to the getopt function in C. It simplifies handling options (flags) and arguments provided to your Java application.

java-cup-11b.jar and java-cup-11b-runtime.jar

The `java-cup-11b.jar` and `java-cup-11b-runtime.jar` files are part of CUP (Constructor of Useful Parsers), a parser generator for Java. They are used to generate parsers for context-free grammars, similar to tools like Yacc or Bison in C. This file hlep generate parser(Parser.java) for the grammar of the language.

Why Your Project Needs These JARs - `java-cup-11b.jar`: Required for generating the parser code during the build process. - `java-cup-11b-runtime.jar`: Needed at runtime to support the execution of the generated parser.

Install Ocaml to check the functionality of the .ml file

To install ocaml with Unix

```
sudo apt update
sudo apt install opam

opam init
opam switch create 4.14.0
eval $(opam env)
```

Run the main program in ocaml directory

```
cd archive/archive/ocaml
make
./mincamlc filepath/input.ml
```

Example:

```
./mincamlc ../tests/typechecking/valid/let_seq.ml
```

Install Python

```
sudo apt update
sudo apt install python3
sudo dnf install python3
python3 --version
```

References

dev.to. 2025. Introduction to Lexers, Parsers and Interpreters with Chevrotain - DEV Community. [ONLINE] Available at: https://dev.to/codingwithadam/introduction-to-lexers-parsers-and-interpreters-with-chevrotain-5c7b. [Accessed 24 January 2025].

central.sonatype.com. 2025. Maven Central: gnu.getopt:java-getopt. [ONLINE] Available at: https://central.sonatype.com/artifact/gnu.getopt/java-getopt/overview. [Accessed 24 January 2025].

www.jflex.de. 2025. JFlex - changelog. [ONLINE] Available at: https://www.jflex.de/changelog.html. [Accessed 24 January 2025].

repo1.maven.org. 2025. Central Repository: gnu/getopt/java-getopt/1.0.13. [ONLINE] Available at: https://repo1.maven.org/maven2/gnu/getopt/java-getopt/1.0.13/. [Accessed 24 January 2025].

www.skenz.it. 2025. compilers:install_windows [Skenz - How To Wiki]. [ONLINE] Available at: https://www.skenz.it/compilers/install_windows. [Accessed 24 January 2025].

jflex.de. 2025. JFlex Ant Task. [ONLINE] Available at: https://jflex.de/jflex_anttask.html. [Accessed 24 January 2025].

ocaml.org. 2025. OCaml - Lexer and parser generators (ocamllex, ocamlyacc). [ONLINE] Available at: https://ocaml.org/manual/5.3/lexyacc.html. [Accessed 24 January 2025].

ocaml.org. 2025. OCaml - Lexer and parser generators (ocamllex, ocamlyacc). [ONLINE] Available at: https://ocaml.org/manual/5.3/lexyacc.html. [Accessed 24 January 2025].

www.geeksforgeeks.org. 2022. Type Checking in Compiler Design - GeeksforGeeks. [ONLINE] Available at: https://www.geeksforgeeks.org/type-checking-in-compiler-design/. [Accessed 24 January 2025].

www.gnu.org. 2025. LongOpt. [ONLINE] Available at: https://www.gnu.org/software/gnuprologjava/api/gnu
[Accessed 24 January 2025].

stackoverflow.com. 2025. Java GNU GetOpt: Understanding LongOpt's StringBuffer flag - Stack Overflow. [ONLINE] Available at: https://stackoverflow.com/questions/57455418/java-gnu-getopt-understanding-longopts-stringbuffer-flag. [Accessed 24 January 2025].