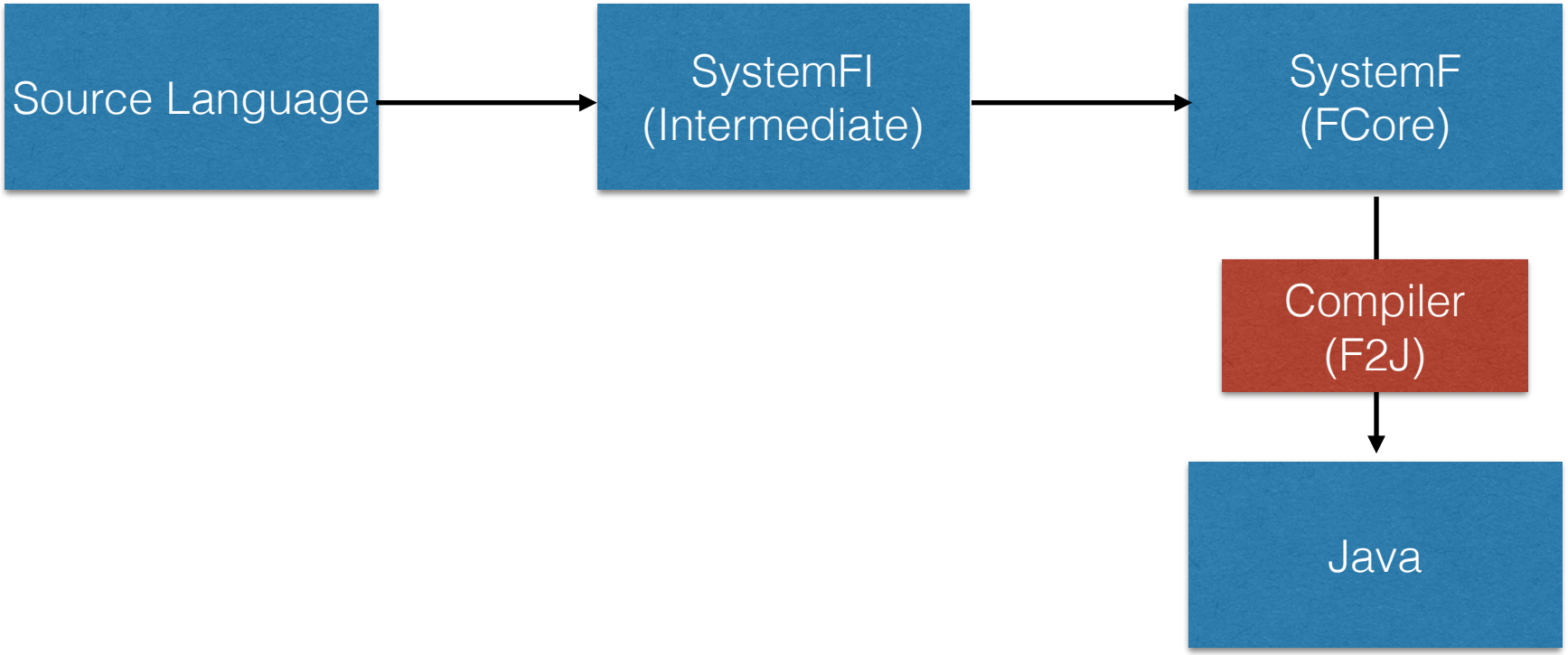
Compiler Infrastructure for F2J

Student: Boya Peng
Supervisor: Dr. Bruno Oliveira

FCore

- A research language with two main goals:
  A. Investigating new compilation strategies for FL in the JVM
  B. Investigating new language designs for modularity and extensibility of software
- FCore is based on System F, a well-known minimal core language for functional programming

Overview


Source Language
SystemFI
(Intermediate)
SystemF
(FCore)
Compiler
(F2J)
Java

Work Accomplished

- REPL (Read-Eval-Print-Loop) for FCore
- Testing tool for efficient regression testing for FCore

f2ji

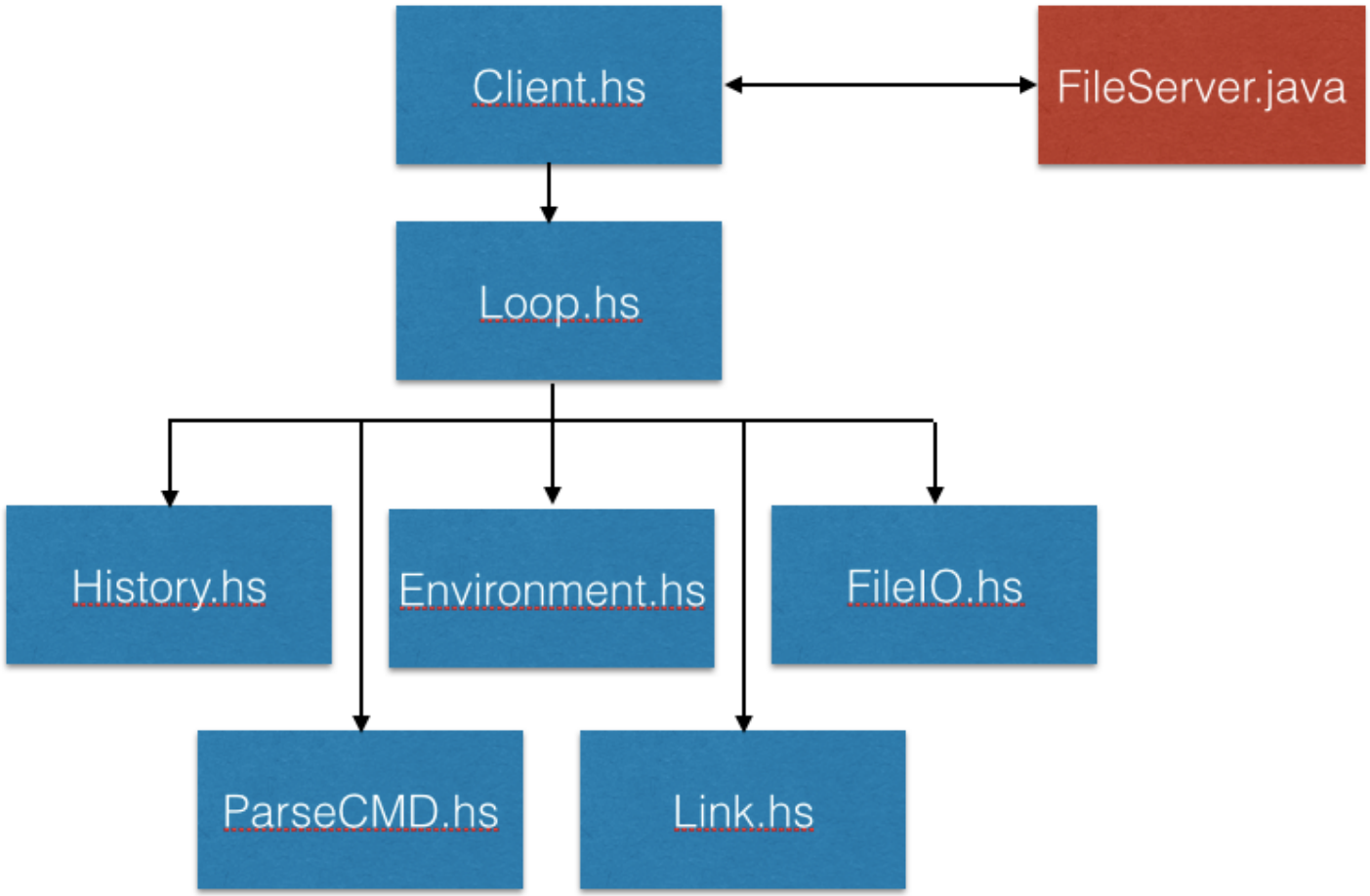
- Interactive interpreter
- Dynamic execution
- REPL environment

```
Emma-2:fcore Emma$ f2ji

Welcome to f2ji!
----------------------------------------
[COMMANDS] [SOURCE FILE/FLAG]
Commands:
:help                   Display help manual
:run <sourceFile>       Compile and run sourceFile
:link <sourceFile> -m <module1> <module2> ...
                        Link sourceFile with modules
:expr <sourceFile>      Show core expression of the file
:let var = expr         Bind expr to var
:type var               Show the type of var
:replay                 Replay all previous user commands
:replay default         Replay commands from default.txt
:clear                  Clear environment
:quit                   Quit f2ji

--- Commands for settings ---
:set method opt         Set compilation options
:set simplify on/off    Turn on/off the simplifier

--- Commands for displaying information ---
:show time on/off       Show/Hide execution time
:show file on/off       Show/Hide source file and .java file contents
:show <sourcefile>      Show file content
:show env               Show current bindings
:show method            Show available compilation options
----------------------------------------
f2ji>
```

Challenges

- Communicate with file server for dynamic compilation
  - File Server: continuously process source files
- Maintain a REPL environment
  - New bindings of variables are added to:
    - Local environment
    - Environment maintained by type server
  - Communicate with type server to query about variable types

f2ji -- Implementation



f2ji -- Commands

:run <sourceFile> -- compile and run sourceFile

:let var = expr -- bind the variable to the expression

:type var -- check the type of the variable

:replay -- re-execute all the previously executed commands

:replay default -- execute commands from default.txt

:set method opt -- set different compilation methods

f2ji -- Commands

:show method -- show all available methods

:show time on/off -- show CPU time after execution

:show file on/off -- show contents of source file and the generated java file

:show <sourceFile> -- show content of sourceFile

:show env -- display bindings in the current environment

:clear -- clear the environment

f2ji -- Example

:set method apply stack

:show method

:run fractals.sf

f2ji -- Example

```
:let x = \(x:Int). x

:let y = 3

:type x

:show env

x y

:replay
```

f2ji -- Example

:show file on

:show time on

:show fibo.sf

:run fibo.sf

Module System

- The current module system and its implementation (by George)

```
module M
  f1 = e1
  f2 = e2
  ...
  fn = en
end
```

```
let M =
  let rec
    f1 = e1
    f2 = e2
    ...
    fn = en
  in
  { f1 = f1, f2 = f2, ..., fn = fn }
in
...
```

Module System -- Linking

In f2ji:

- :link m1.sf -m module1.sf module2.sf
- :run m1c.sf

Module System
-- Future work

- Allow “import Module” in implementation files
- Enable loading modules in f2ji
- Support separate compilation

Thank you!