Compiler Infrastructure for F2J

Student: Boya Peng
Supervisor: Dr. Bruno Oliveira

F2J Compiler

- F2J — A research compiler/language with two main goals:

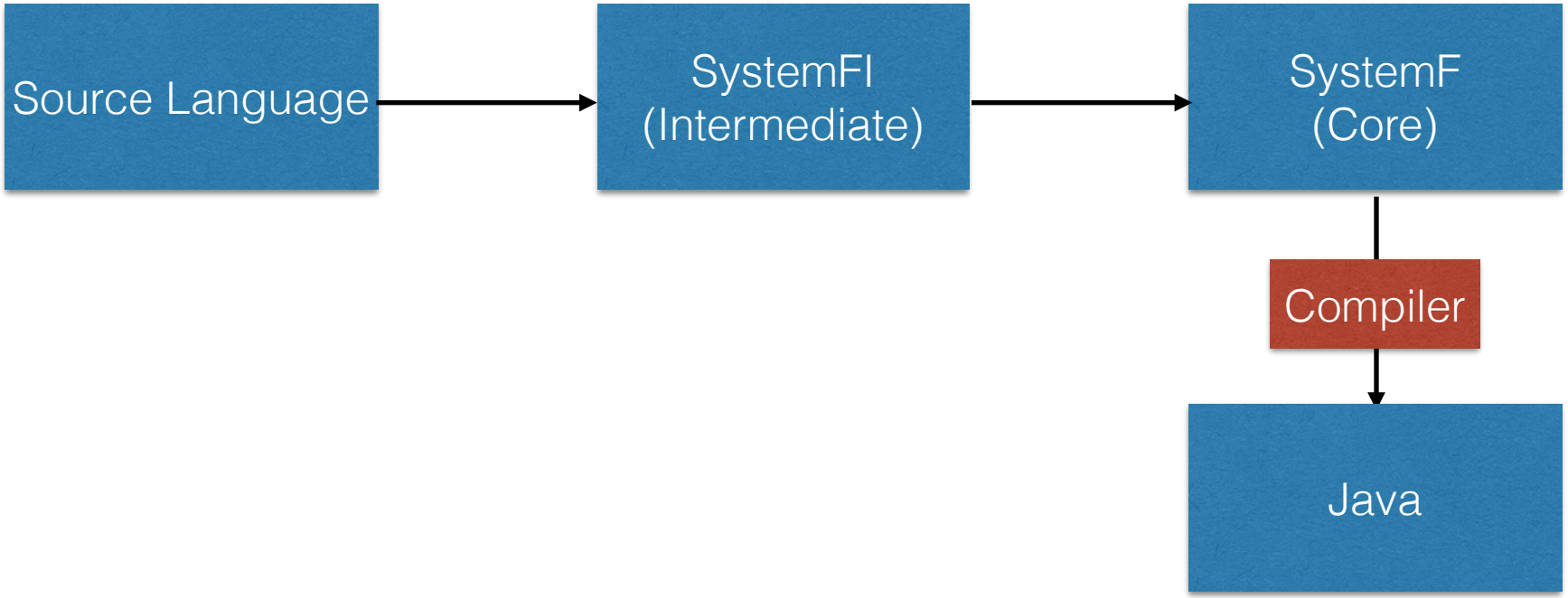
  A. Investigating new compilation strategies for FL in the JVM

  B. Investigating new language designs for modularity and extensibility of software

- F2J is based on System F[1], a well-known minimal core language for functional programming

[1]. Girard, Jean-Yves (1972), Interprétation fonctionnelle et élimination des coupures de l'arithmétique d'ordre supérieur (Ph.D. thesis) (in French), Université Paris 7. Reynolds, John (1974).Towards a Theory of Type Structure

Overview


Source Language
SystemFI
(Intermediate)
SystemF
(Core)
Compiler
Java

Project Goals

- Implement a REPL for F2J — f2ji
- Build a simple module system that supports separate compilation

f2ji

- Overview:

  f2ji is integrated with F2J and can interact with the JVM

- Motivations:

  A. To efficiently compile and run source files (save the overheads of quitting and restarting the JVM)

  B. To take in user inputs, evaluate them and return the results.

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

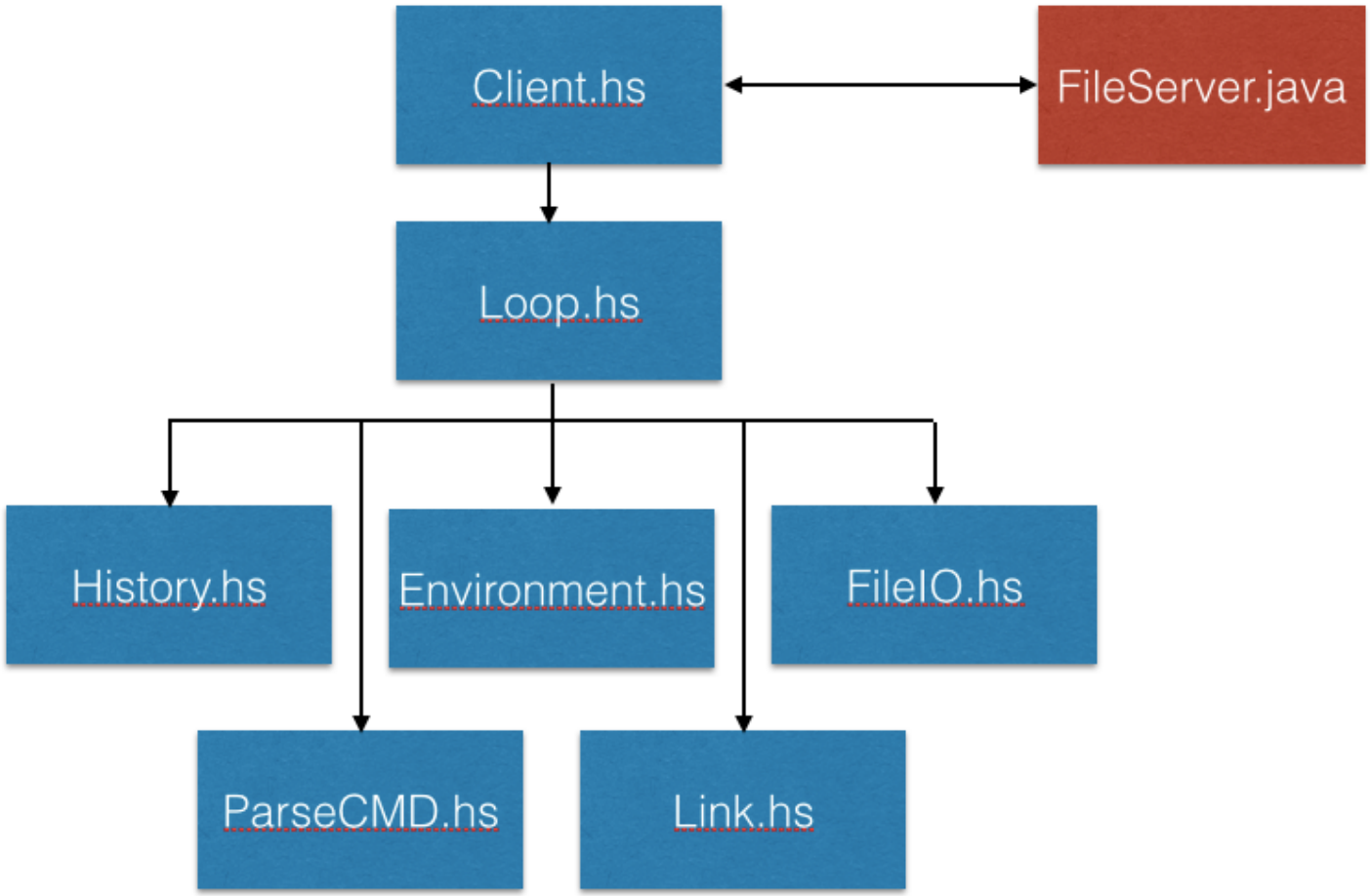
:show file on

:show time on

:show fibo.sf

:run fibo.sf

f2ji -- Implementation



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!