Lab #3. F- Interpreter

Prof. Jaeseung Choi

Dept. of Computer Science and Engineering

Sogang University

General Information

- **Check the *Assignment* tab of *Cyber Campus***
  - Skeleton code (`Lab3.tgz`) is attached together with this slide
  - Submission will be accepted in the same post, too
- **Deadline: 5/10 Friday 23:59**
  - Late submission deadline: **5/12 Sunday 23:59** **(-20% penalty)**
  - Delay penalty is applied uniformly **(not problem by problem)**
- **Please read the instructions in this slide carefully**
  - This slide is a step-by-step tutorial for the lab
  - It also contains important submission guidelines
    - If you do not follow the guidelines, **you will get penalty**

Skeleton Code Structure

- **Copy `Lab3.tgz` into CSPRO server and decompress it**
  - This course will use cspro**2**.sogang.ac.kr (**don't miss the 2**)
  - Don't decompress-and-copy; copy-and-decompress
- **`FMinus`: Only one directory for `F-` this time**
- **`check.py`, `config`: Script and config file for self-grading (same as before)**

```
jschoi@cspro2:~$ tar -xzf Lab3.tgz
jschoi@cspro2:~$ cd Lab3/
jschoi@cspro2:~/Lab3$ ls
FMinus  check.py  config
```

Directory Structure of `FMinus`

- **Skeleton code structure under `src/` is same as before**
  - `AST.fs`: Syntax definition of the `F-` language
  - `FMinus.fs`: You have to **implement the semantics** here
  - `Types.fs`: Type definitions needed for semantics
  - `Main.fs`: Main driver code of the interpreter
  - `Lexer.fsl`, `Parser.fsy`: Parser (you don't have to care)
- **Do NOT fix any source files other than `FMinus.fs`**

```
jschoi@cspro2:~/Lab3$ cd FMinus/
jschoi@cspro2:~/Lab3/FMinus$ ls
FMinus.fsproj  src  testcase
jschoi@cspro2:~/Lab3/FMinus$ ls src
AST.fs  FMinus.fs  Lexer.fsl  Main.fs  Parser.fsy  Types.fs
```

`F-` Language Syntax

- **Similar to the extended version of `F-` in our lecture slide**
  - The whole program is an expression

$$
\begin{aligned}
E \rightarrow\ & n \\
& \mid \textbf{true} \mid \textbf{false} \\
& \mid x \\
& \mid -E \\
& \mid E + E \mid E - E \\
& \mid E < E \mid E > E \mid E = E \mid E <> E \\
& \mid \textbf{if}\ E\ \textbf{then}\ E\ \textbf{else}\ E \\
& \mid \textbf{let}\ x = E\ \textbf{in}\ E \\
& \mid \textbf{let}\ f\ x = E\ \textbf{in}\ E \\
& \mid \textbf{let rec}\ f\ x = E\ \textbf{in}\ E \\
& \mid \textbf{fun}\ x \rightarrow E \\
& \mid E\ \ E
\end{aligned}
$$

Expression

F- Language Semantics

- **Relation $\boldsymbol{\rho \vdash e \Downarrow v}$ defines the evaluation of expression**
  - We use the same semantic domain to our lecture slide
  - $\boldsymbol{\rho \in Env = Var \to Val}$, $\mathbf{v} \in \boldsymbol{Val = Z + B + Func + RecFunc}$

$$\frac{}{\rho \vdash n \Downarrow n} \qquad \frac{}{\rho \vdash \textbf{true} \Downarrow true} \qquad \frac{}{\rho \vdash \textbf{false} \Downarrow false} \qquad \frac{}{\rho \vdash x \Downarrow \rho(x)}$$

$$\frac{\rho \vdash e_1 \Downarrow n_1}{\rho \vdash -e_1 \Downarrow -n_1} \qquad \frac{\rho \vdash e_1 \Downarrow n_1 \quad \rho \vdash e_2 \Downarrow n_2}{\rho \vdash e_1 + e_2 \Downarrow n_1 + n_2} \qquad \frac{\rho \vdash e_1 \Downarrow n_1 \quad \rho \vdash e_2 \Downarrow n_2}{\rho \vdash e_1 - e_2 \Downarrow n_1 - n_2}$$

$$\frac{\rho \vdash e_1 \Downarrow n_1 \quad \rho \vdash e_2 \Downarrow n_2}{\rho \vdash e_1 < e_2 \Downarrow true}\ n_1 < n_2 \qquad \frac{\rho \vdash e_1 \Downarrow n_1 \quad \rho \vdash e_2 \Downarrow n_2}{\rho \vdash e_1 < e_2 \Downarrow false}\ n_1 \geq n_2$$

$$\frac{\rho \vdash e_1 \Downarrow n_1 \quad \rho \vdash e_2 \Downarrow n_2}{\rho \vdash e_1 > e_2 \Downarrow true}\ n_1 > n_2 \qquad \frac{\rho \vdash e_1 \Downarrow n_1 \quad \rho \vdash e_2 \Downarrow n_2}{\rho \vdash e_1 > e_2 \Downarrow false}\ n_1 \leq n_2$$

F- Language Semantics

- **Relation $\rho \vdash e \Downarrow v$ defines the evaluation of expression**
  - We use the same semantic domain to our lecture slide
  - $\boldsymbol{\rho \in Env = Var \rightarrow Val}$, $\mathbf{v} \in \boldsymbol{Val = Z + B + Func + RecFunc}$

$$\frac{\rho \vdash e_1 \Downarrow v_1 \quad \rho \vdash e_2 \Downarrow v_2}{\rho \vdash e_1 = e_2 \Downarrow true} \; (v_1 = v_2 = n) \vee (v_1 = v_2 = b)$$

$$\frac{\rho \vdash e_1 \Downarrow v_1 \quad \rho \vdash e_2 \Downarrow v_2}{\rho \vdash e_1 = e_2 \Downarrow false} \; (v_1 = n_1 \neq n_2 = v_2) \vee (v_1 = b_1 \neq b_2 = v_2)$$

$$\frac{\rho \vdash e_1 \Downarrow v_1 \quad \rho \vdash e_2 \Downarrow v_2}{\rho \vdash e_1 \;<>\; e_2 \Downarrow false} \; (v_1 = v_2 = n) \vee (v_1 = v_2 = b)$$

$$\frac{\rho \vdash e_1 \Downarrow v_1 \quad \rho \vdash e_2 \Downarrow v_2}{\rho \vdash e_1 \;<>\; e_2 \Downarrow true} \; (v_1 = n_1 \neq n_2 = v_2) \vee (v_1 = b_1 \neq b_2 = v_2)$$

F- Language Semantics

- **Relation $\boldsymbol{\rho \vdash e \Downarrow v}$ defines the evaluation of expression**
  - We use the same semantic domain to our lecture slide
  - $\boldsymbol{\rho \in Env = Var \rightarrow Val}$, $\mathbf{v} \in \boldsymbol{Val = Z + B + Func + RecFunc}$

$$\frac{\rho \vdash e_1 \Downarrow true \quad \rho \vdash e_2 \Downarrow v}{\rho \vdash \textbf{if}\ e_1\ \textbf{then}\ e_2\ \textbf{else}\ e_3 \Downarrow v}$$

$$\frac{\rho \vdash e_1 \Downarrow false \quad \rho \vdash e_3 \Downarrow v}{\rho \vdash \textbf{if}\ e_1\ \textbf{then}\ e_2\ \textbf{else}\ e_3 \Downarrow v}$$

$$\frac{\rho \vdash e_1 \Downarrow v_1 \quad \rho[x \mapsto v_1] \vdash e_2 \Downarrow v_2}{\rho \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 \Downarrow v_2}$$

$$\frac{\rho[f \mapsto \langle x, e_1, \rho \rangle] \vdash e_2 \Downarrow v}{\rho \vdash \textbf{let}\ f\ x = e_1\ \textbf{in}\ e_2 \Downarrow v}$$

$$\frac{\rho[f \mapsto \langle f, x, e_1, \rho \rangle] \vdash e_2 \Downarrow v}{\rho \vdash \textbf{let rec}\ f\ x = e_1\ \textbf{in}\ e_2 \Downarrow v}$$

$$\frac{}{\rho \vdash \textbf{fun}\ x \rightarrow e \Downarrow \langle x, e, \rho \rangle}$$

F- Language Semantics

- **Relation $\boldsymbol{\rho \vdash e \Downarrow v}$ defines the evaluation of expression**
  - We use the same semantic domain to our lecture slide
  - $\boldsymbol{\rho \in Env = Var \to Val}$, $\mathbf{v} \in \boldsymbol{Val = Z + B + Func + RecFunc}$

(Application of ***non-recursive*** function)

$$\frac{\rho \vdash e_1 \Downarrow \langle x, e_b, \rho' \rangle \qquad \rho \vdash e_2 \Downarrow v_{arg} \qquad \rho'[x \mapsto v_{arg}] \vdash e_b \Downarrow v}{\rho \vdash e_1 \;\; e_2 \Downarrow v}$$

(Application of ***recursive*** function)

$$\frac{\rho \vdash e_1 \Downarrow \langle f, x, e_b, \rho' \rangle \qquad \rho \vdash e_2 \Downarrow v_{arg} \qquad \rho'[x \mapsto v_{arg}][f \mapsto \langle f, x, e_b, \rho' \rangle] \vdash e_b \Downarrow v}{\rho \vdash e_1 \;\; e_2 \Downarrow v}$$

Implementing Semantics

- **To complete the interpreter of `F-`, you must implement the semantics of `F-` language in `FMinus.fs` file**
  - You have to implement only one function: **`evalExp()`**
    - **Execution of program = Evaluation of the expression**
  - Type definition of **`Env`** and **`Val`** are provided in **`Types.fs`**
  - If the semantics of program is not defined, your interpreter must raise **`UndefinedSemantics`** exception defined in **`Types.fs`**

```
let rec evalExp (exp: Exp) (env: Env) : Val =
  ...

// This part is given for you.
let run (prog: Program) : Val =
  evalExp prog Map.empty
```

Building and Testing

- **In `testcase` directory, `tc-*` and `ans-*` files are provided**
  - After compiling the interpreter with the **`dotnet build -o out`** command, you can run program written in **`F-`** language
  - The interpreter will **print the evaluation result of the program**

```
jschoi@cspro2:~/Lab3/FMinus$ cat testcase/tc-1
let x = 10 in
...
jschoi@cspro2:~/Lab3/FMinus$ dotnet build -o out
...
jschoi@cspro2:~/Lab3/FMinus$ ./out/FMinus testcase/tc-1
13
```

This result must match with the content of **`ans-1`** (expected output)

Tip: Printing AST

- **In F- language, you may feel confused about how the input program is parsed into AST**
  - For example, is "`f x + 1`" parsed into `(f x) + 1` or `f (x + 1)`?
  - You can *temporarily* add the following `printfn()` to print out the program AST (don't forget to erase it before the submission)

```
let run (prog: Program) : Val =
  printfn "%A" prog
  evalExp prog Map.empty
```

```
jschoi@cspro2:~/Lab3/FMinus$ cat parsing-test
f x + 1
jschoi@cspro2:~/Lab3/FMinus$ ./out/FMinus parsing-test
Add (App (Var "f", Var "x"), Num 1)
...
```

Self-Grading Script

- **If you think you have solved all the problems, you can run `check.py` as a final check**
  - '**`O`**': Correct, '**`X`**': Incorrect, '**`E`**': Unhandled exception in your code
  - '**`C`**': Compile error, '**`T`**': Timeout (maybe infinite recursion)
- **If you correctly raise `UndefinedSemantics` exception for an invalid program, it will be graded as '`O`' (not '`E`')**
  - If you raise **`UndefinedSemantics`** for valid program, it is **'`X`'**

```
jschoi@cspro2:~/Lab3$ ls
FMinus  check.py  config
jschoi@cspro2:~/Lab3$ $ ./check.py
[*] FMinus : OOOO
```

Actual Grading

- **I will use different test case set during the real grading**
  - So you are encouraged to run you code with your own test cases (try to think of various inputs)
  - Some students ask me to provide more test cases, but **it is important to practice this on your own**
- **You will get the point based on the number of test cases that you pass**
  - 100 point in total (but recall that each lab has different weight)

Submission Guideline

- **You should submit only one F# source code file**
  - **`FMinus.fs` (from `Lab3/FMinus/src/FMinus.fs`)**
- **If the submitted file fails to compile with skeleton code when I type "`dotnet build`", cannot give you any point**
- **Submission format**
  - Upload this file directly to *Cyber Campus* (**do not zip them**)
  - **Do not change the file name** (e.g., adding any prefix or suffix)
  - If your submission format is wrong, you will get **-20% penalty**