I217E: Functional Programming

Nao Hirokawa

JAIST

Term 2-1, 2022

`http://www.jaist.ac.jp/~hirokawa/lectures/fp/`

Schedule

10/12	introduction	11/9	interpreters
10/14	algebraic data types I	11/11	compilers
10/19	algebraic data types II	11/16	termination
10/21	applications	11/18	confluence
10/26	program reasoning	11/25	verification
10/28	data structures I	11/30	review
11/2	data structures II	12/5	exam
11/4	computational models		

Evaluation

exam (60) + reports (40)

Interpreter and Compiler

Exercise: Implement Interpreter for Arithmetic

```
data Exp = Val Int | Add Exp Exp | Mul Exp Exp

eval :: Exp -> Int
eval (Val n) =
eval (Add e1 e2) =
eval (Mul e1 e2) =
```

for instance,

$$\texttt{eval}\ (\mathsf{Mul}\ (\mathsf{Val}\ 10)\ (\mathsf{Add}\ (\mathsf{Val}\ 20)\ (\mathsf{Val}\ 30))) = 500$$

Stack-Based Virtual Machines (VM)

```
data Exp = Val Int | Add Exp Exp | Mul Exp Exp
data Instruction = Push | IVal Int | IAdd | IMul
type Bytecode = [Instruction]
```

$$\begin{aligned} & \texttt{vm}\ (\mathsf{compile}\ (\mathsf{Mul}\ (\mathsf{Val}\ 10)\ (\mathsf{Add}\ (\mathsf{Val}\ 20)\ (\mathsf{Val}\ 30)))) \\ = {} & \texttt{vm}\ [\mathsf{Push}, \mathsf{IVal}\ 10, \mathsf{Push}, \mathsf{IVal}\ 20, \mathsf{Push}, \mathsf{IVal}\ 30, \mathsf{IAdd}, \mathsf{IMul}] \\ = {} & 500 \end{aligned}$$

Remark

use 16/32/64/128-bit integers for instruction in practice

Interpreting Bytecode

$$\mathsf{bc} = [\overset{0}{\mathsf{Push}}, \overset{1}{\mathsf{IVal}}\ 10, \overset{2}{\mathsf{Push}}, \overset{3}{\mathsf{IVal}}\ 20, \overset{4}{\mathsf{Push}}, \overset{5}{\mathsf{IVal}}\ 30, \overset{6}{\mathsf{IAdd}}, \overset{7}{\mathsf{IMul}}]$$

$$\begin{array}{rclcr} & & & \mathsf{pc} & \mathsf{stack} \\ \texttt{vm}\ \mathsf{bc} & = & \texttt{vm}'\ \mathsf{bc} & 0 & [\,] \\ & = & \texttt{vm}'\ \mathsf{bc} & 2 & [10] \\ & = & \texttt{vm}'\ \mathsf{bc} & 4 & [20, 10] \\ & = & \texttt{vm}'\ \mathsf{bc} & 6 & [30, 20, 10] \\ & = & \texttt{vm}'\ \mathsf{bc} & 7 & [50, 10] \\ & = & \texttt{vm}'\ \mathsf{bc} & 8 & [500] \\ & = & 500 & & \end{array}$$

Compiling Expressions

$$\begin{aligned} & \texttt{compile}\ (\mathsf{Mul}\ (\mathsf{Val}\ 10)\ (\mathsf{Add}\ (\mathsf{Val}\ 20)\ (\mathsf{Val}\ 30))) \\ = {} & \texttt{compile}\ (\mathsf{Val}\ 10) \mathbin{++} \\ & \texttt{compile}\ (\mathsf{Add}\ (\mathsf{Val}\ 20)\ (\mathsf{Val}\ 30)) \mathbin{++} [\mathsf{IMul}] \\ = {} & [\mathsf{Push}, \mathsf{IVal}\ 10] \mathbin{++} \\ & \texttt{compile}\ (\mathsf{Add}\ (\mathsf{Val}\ 20)\ (\mathsf{Val}\ 30)) \mathbin{++} [\mathsf{IMul}] \\ = {} & \cdots \\ = {} & [\mathsf{Push}, \mathsf{IVal}\ 10, \mathsf{Push}, \mathsf{IVal}\ 20, \mathsf{Push}, \mathsf{IVal}\ 30, \mathsf{IAdd}, \mathsf{IMul}] \end{aligned}$$

Graphs

Directed Graphs

Definition

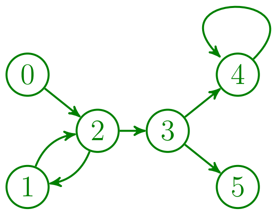
directed graph is pair (V, E) with $E \subseteq V \times V$

Example

graph $G = (V, E)$ with

$$V = \{0, 1, 2, 3, 4, 5\}$$

$$E = \left\{ \begin{array}{l} (0,2), \\ (1,2), \\ (2,1), (2,3), \\ (3,4), (3,5), \\ (4,4) \end{array} \right\}$$



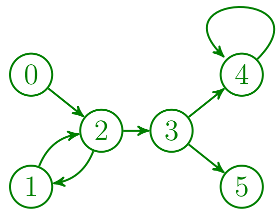
Adjacency Lists

Definition

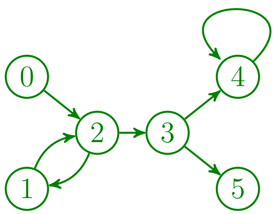
adjacency list for (V, E) is $\left\{ (x, [\, y \mid (x,y) \in E \,]) \;\middle|\; x \in V \right\}$

Example

$\left\{(0, [2]), (1, [2]), (2, [1, 3]), (3, [4, 5]), (4, [4]), (5, [\,])\right\}$ depicts



Exercises



for $g = \left[(0,2), (1,2), (2,1), (2,3), (3,4), (3,5), (4,4)\right]$

$\texttt{succ}\ g\ 2 = [1, 3]$ $\qquad$ $\texttt{pred}\ g\ 2 = [0, 1]$

Exercise

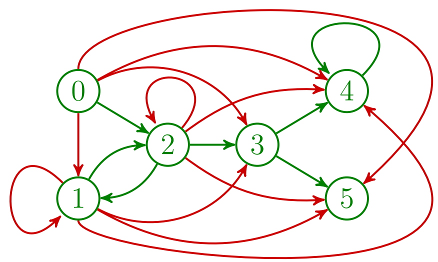
implement `succ` and `pred` for graphs

Transitive Closure

Definition

transitive closure E^+ of relation E is **smallest** relation such that

$E \subseteq E^+$, and $(x, z) \in E^+$ whenever $(x, y) \in E^+$ and $(y, z) \in E^+$



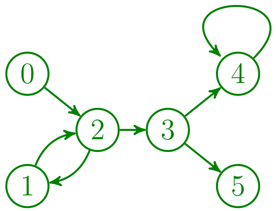
Fixed Point Characterization of E^+

E^+ is fixed point of $\mathtt{F}$ starting from E:

$$\mathtt{F}(X) = X \cup \{\ (x, z) \mid (x, y) \in X \text{ and } (y, z) \in X\}$$

Example

E:



$$\mathtt{F}(E) = E \cup \{(0,1),(0,3),(1,1),(1,3),(2,2),(2,4),(2,5)\}$$
$$\mathtt{F}(\mathtt{F}(E)) = \mathtt{F}(E) \cup \{(0,4),(0,5),(1,4),(0,5)\}$$
$$\mathtt{F}(\mathtt{F}(\mathtt{F}(E))) = \mathtt{F}(\mathtt{F}(E)) \quad \text{fixed point}$$

Homework 1/2

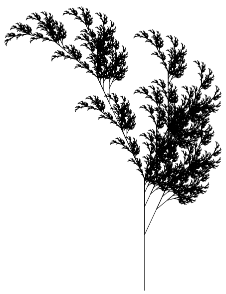
1. Implement `compile` :: Exp → Bytecode.
2. Implement `vm` :: Bytecode → Int.
3. Implement `tc` :: Eq a => $[(a,a)] \to [(a,a)]$ that computes the transitive closure of a given relation. For instance,

$$\mathtt{tc}\ \big[(0,2),(1,2),(2,1),(2,3),(3,4),(3,5),(4,4)\big]$$
$$= \left[\begin{array}{l} (0,1),(0,2),(0,3),(0,4),(0,5), \\ (1,1),(1,2),(1,3),(1,4),(1,5), \\ (2,1),(2,2),(2,3),(2,4),(2,5), \\ (3,4),(3,5), \\ (4,4) \end{array}\right]$$

Homework 2/2

4. Complete `eval` in `LSystem.hs` to draw the fractal plant:

S → ++++++++++++++++++X
F → FF
X → F-[[X]+X]+F[+FX]-X
$\delta = 25^\circ$



See `https://en.wikipedia.org/wiki/L-system` for the interpretations of the stack operations `[` and `]`.