I217E: Functional Programming

Nao Hirokawa

JAIST

Term 2-1, 2022

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

Schedule

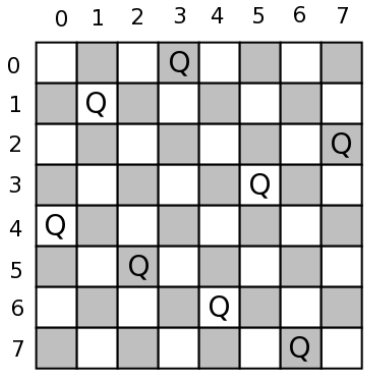
10/12	introduction	11/9	interpreters
10/14	algebraic data types I	11/11	compilers
10/19	algebraic data types II	11/16	termination
10/21	applications	11/18	confluence
10/26	program reasoning	11/25	verification
10/28	data structures I	11/30	review
11/2	data structures II	12/5	exam
11/4	computational models		

Evaluation

exam (60) + reports (40)

Generate-and-Test Method

Mission: Solve N-Queens Problem



One of Solutions: $[4, 1, 5, 0, 6, 3, 7, 2]$, depicted above

Zip (Parallel Composition)

$$\texttt{zip}\ [x_1,\ldots,x_n]\ [y_1,\ldots,y_n] = [(x_1,y_1),\ldots,(x_n,y_n)]$$
$$\texttt{zipWith}\ f\ [x_1,\ldots,x_n]\ [y_1,\ldots,y_n] = [f\ x_1\ y_1,\ldots,f\ x_n\ y_n]$$

Exercise

define `myZip` and `myZipWith`

Permutations

$$\texttt{prefixes}\ [2,3] = [[\,],[2],[2,3]]$$

$$\texttt{suffixes}\ [2,3] = [[2,3],[3],[\,]]$$

$$\texttt{interleave}\ 1\ [2,3] = [[1,2,3],[2,1,3],[2,3,1]]$$

$$\texttt{permutations}\ [\ \ 2,3] = [[2,3],[3,2]]$$
$$\texttt{permutations}\ [1,2,3] = [[1,2,3],[2,1,3],[2,3,1],$$
$$[1,3,2],[3,1,2],[3,2,1]]$$

Generate-and-Test Method

1. `safe` $i\ j\ i'\ j'$ returns True if
   queens at (i,j) and (i',j') have no conflict, or
   they are identical
2. ok $[x_0,\ldots,x_{n-1}]$ returns True if
   `safe` $i\ x_i\ j\ x_j$ holds for all $i,j \in \{0,\ldots,n-1\}$
3. complete N-queens problem solver:

$$\texttt{nqueen}\ n =$$
$$[\,xs \mid xs \texttt{ <- permutations } [0\,..\,n-1], \texttt{ok xs}\,]$$

Mechanism of List Comprehension

Transformation Rules for List Comprehension

Example

$$
\begin{aligned}
&[\,x+1 \mid x \leftarrow [1,2,3]\,] && = \text{map}\ (\backslash x \to x+1)\ [1,2,3]\\
&[\,x \mid x \leftarrow [1,2,3],\ \text{even}\ x\,] && = \text{filter}\ \text{even}\ [1,2,3]\\
&[\,(x,y) \mid x \leftarrow [1,2], y \leftarrow [\texttt{"a"},\texttt{"b"}]\,] = \\
&\quad \text{concat}\ [\,[\,(x,y) \mid y \leftarrow [\texttt{"a"},\texttt{"b"}]\,] \mid x \leftarrow [1,2]\,]
\end{aligned}
$$

Translation Rules

$$
\begin{aligned}
&[\,x \mid x \leftarrow xs\,] && = xs\\
&[\,e \mid x \leftarrow xs\,] && = \text{map}\ (\backslash x \to e)\ xs\\
&[\,e \mid x \leftarrow xs, p, \cdots] && = [\,e \mid x \leftarrow \text{filter}\ (\backslash x \to p)\ xs, \cdots]\\
&[\,e \mid x \leftarrow xs, y \leftarrow ys, \cdots] && = \text{concat}\ [\,[\,e \mid y \leftarrow ys, \cdots] \mid x \leftarrow xs\,]
\end{aligned}
$$

Translating List Comprehension: `permutations`

$$
\begin{aligned}
&\text{permutations}\ (x : xs)\\
&= [zs \mid ys \leftarrow \text{permutations}\ xs, zs \leftarrow \text{interleave}\ x\ ys]\\
&= \text{concat}\ [\,[\,zs \mid zs \leftarrow \text{interleave}\ x\ ys\,] \mid ys \leftarrow \text{permutations}\ xs\,]\\
&= \text{concat}\ [\,\text{interleave}\ x\ ys \mid ys \leftarrow \text{permutations}\ xs\,]\\
&= \text{concat}\ (\text{map}\ (\backslash ys \to \text{interleave}\ x\ ys)\ (\text{permutations}\ xs))\\
&= \text{concat}\ (\text{map}\ (\text{interleave}\ x)\ (\text{permutations}\ xs))
\end{aligned}
$$

Note

actual translation in Haskell is based on monads

Binary Search Trees

Recall Binary Trees (Parametrized)

```
data Tree a = Leaf
            | Node (Tree a) a (Tree a)
  deriving Show
```

Exercise

implement following functions:

1. `member` :: Eq a => $a \to$ Tree $a \to$ Bool
2. `depth` :: Tree $a \to$ Int
3. `inorder` :: Tree $a \to [a]$

Binary Search Trees

Definition

tree t is binary search tree if $x_1 < \cdots < x_n$ for `inorder` $t = [x_1, \ldots, x_n]$

Exercise

implement membership function that is in $O(d)$ wrt depth d

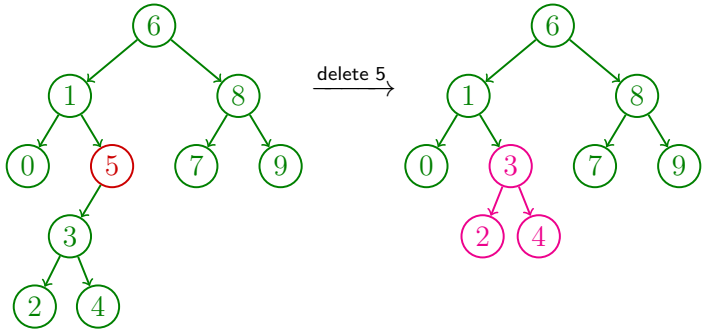
$$\texttt{member} :: \text{Ord } a \Rightarrow a \to \text{Tree } a \to \text{Bool}$$

$$\texttt{member}\ x\ \text{Leaf} = \text{False}$$

$$\texttt{member}\ x\ (\text{Node}\ \ell\ y\ r) = \begin{cases} \ldots & \text{if } x = y \\ \ldots & \text{if } x < y \\ \ldots & \text{if } x > y \end{cases}$$

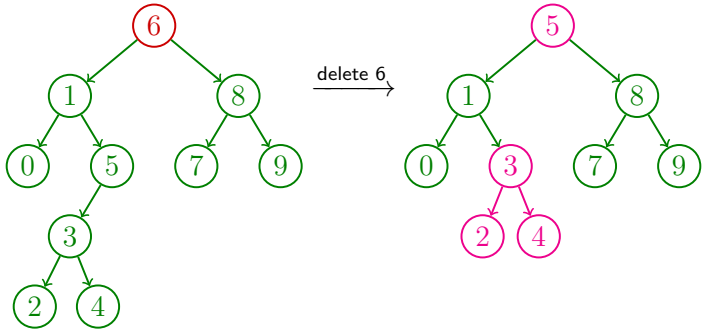
also implement `add`

How to Delete Node 5?



replace node 5 by its left branch

How to Delete Node 6?



1. replace node 5 by its left branch
2. replace 6 by 5

Homework 1/2

1. Implement the following function:

$$\texttt{myUnzip}\ [(x_1, y_1), \ldots, (x_n, y_n)] = ([x_1, \ldots, x_n], [y_1, \ldots, y_n])$$

2. Implement `power` $:: [a] \to [[a]]$, the list version of the power set function:

$$\begin{aligned} \texttt{power}\ [\,] &= \ldots \text{ (compute this by yourself)} \\ \texttt{power}\ [3] &= \ldots \\ \texttt{power}\ [2,3] &= [[\,], [3], [2], [2,3]] \\ \texttt{power}\ [1,2,3] &= \ldots \end{aligned}$$

Note that we do not care the order of sublists.

Homework 2/2

3. Use the generate-and-test method to generate all solutions to the 3×3 magic square problem:

6	1	8
7	5	3
2	9	4

8	1	6
3	5	7
4	9	2

$\ldots$

$$\texttt{magicSquare} = [\,[6,1,8,\ 7,5,3,\ 2,9,4],$$
$$[8,1,6,\ 3,5,7,\ 4,9,2],\ldots\ ]$$

See the following webpage:

`https://en.wikipedia.org/wiki/Magic_square`

4. Implement `delete` for binary search trees.