Prof. Dr. Peter Thiemann

Janek Spaderna
janek.spaderna@pluto.uni-freiburg.de

Winter Term 2022/23

Issued 2022-11-04
Discussion on 2022-11-11

Functional Programming

https://proglang.informatik.uni-freiburg.de/teaching/functional-programming/2022/

Exercise Sheet 3

Exercise 1 (List functions III—folding)

1. Use `foldr` to define the following functions on lists:
   - `filter`
   - `remdups`, which removes consecutive duplicates
   - `avg`, which computes the average of a list of `Double` in a *single* pass
2. The *r* in `foldr` indicates an *right-associative* fold. It is complemented by `foldl`, which associates to the *left:*

   ```
   foldr f z [x₁, x₂, ..., xₙ] = x₁ `f` (x₂ `f` ··· (xₙ `f` z)··)
   foldl f z [x₁, x₂, ..., xₙ] = (··(z `f` x₁) ··· `f` xₙ₋₁) `f` xₙ
   ```

   Give two implementations of `foldl :: (b -> a -> b) -> b -> [a] -> b`

   a) recursive, using pattern matching
   b) non-recursive, in terms of `foldr`

Exercise 2 (Term induction)

Let s, t, r be terms and p, q be strings over the natural numbers. Prove the following propositions:

1. If $pq \in \mathcal{P}os(s)$, then $s|_{pq} = (s|_p)|_q$.
2. If $p \in \mathcal{P}os(s)$ and $q \in \mathcal{P}os(t)$, then $(s[t]_p)|_{pq} = t|_q$.
3. If $p \in \mathcal{P}os(s)$ and $q \in \mathcal{P}os(t)$, then $(s[t]_p)[r]_{pq} = s[t[r]_q]_p$.

Exercise 3 (Terms and subterms in Haskell)

The file `BoolTerm.hs` linked on the lecture home page contains the definition of the `BoolTerm` ADT shown below. It represents terms $T(\Sigma, X)$ for $\Sigma = \{\mathbf{T}^{(0)}, \mathbf{F}^{(0)}, \neg^{(1)}, \wedge^{(2)}, \vee^{(2)}\}$ and $X =$ `Char` in Haskell.

```
data BoolTerm
  = T
  | F
  | Not BoolTerm
  | Conj BoolTerm BoolTerm
  | Disj BoolTerm BoolTerm
  | Var Char
  deriving (Eq, Show)
```

The file contains stubs for the following functions you should write:

- The function `pos` should return the set $\mathcal{P}os$(`t`) for a term `t`.

  Note: A position of a term is a string over the alphabet of the natural numbers. It is represent as `[Integer]`. Represent a *set of positions* as a list of lists. You can verify element uniqueness through QuickCheck properties.

- `(|.)`: the term `t |. p` should correspond to $t|_p$.
- `replace`: the term `replace t r p` should correspond to $t[r]_p$.

Below the stubs you will find a set of QuickCheck properties which correspond to the properties from Exercise 2. If you want, you can of course define further properties. At the very bottom of the file you can find code which makes it possible in the first place to write properties quantifying over values of `BoolTerm`.

Exercise 4 (Substitution)

For this exercise, we use infix notation for terms. For 1. and 2. let $\Sigma = \{\mathbf{T}^{(0)}, \mathbf{F}^{(0)}, \neg^{(1)}, \wedge^{(2)}, \vee^{(2)}\}$.

1. Suppose $t = \neg(x \wedge (\mathbf{T} \vee y)) \in T(\Sigma, X)$.
   Compute $\sigma(t)$ and $\tau\sigma(t)$ where $\sigma = \{x \mapsto \mathbf{F}, y \mapsto \mathbf{T} \wedge x\}$ and $\tau = \{z \mapsto \mathbf{T}, x \mapsto \mathbf{T}\}$.
2. Suppose $t = \neg(\mathbf{F} \wedge (\mathbf{T} \vee y)) \in T(\Sigma, X)$ and $s = \neg(x \wedge (\mathbf{T} \vee (x \wedge \mathbf{F})) \in T(\Sigma, X)$.
   Find a substitution σ such that $\sigma(t) = \sigma(s)$.
3. Let σ, τ be substitutions. Prove $\widehat{\sigma\tau} = \widehat{\sigma}\widehat{\tau}$.
4. Is substitution composition commutative? If yes, give a proof. If not, give a counterexample.