ETH zürich

P. Müller and D. Basin

Formal Methods and Functional Programming

Exercise Sheet 9: Induction

Submission deadline: May 2/3, 2023

Solutions should be submitted during the exercise class.

Assignment 1

Let U be a sequence of integers, defined by $U_0 = U_1 := -1$, and, for all $n \geq 0$, $U_{n+2} = 5 \times U_{n+1} - 6 \times U_n$.

Task 1.1: Prove that, for all natural numbers n, $U_n = 3^n - 2^{n+1}$ using *strong* induction.

Task 1.2: Prove that, for all natural numbers n, $U_n = 3^n - 2^{n+1}$ using *weak* induction.

Assignment 2 (Run-Length Encoding)

The background of this assignment is a simple run-length encoding scheme[1]. In our case, the input data is encoded as a list of natural numbers[2] of even length. The encoded representation has the form $n_1 : v_1 : n_2 : v_2 : \ldots :$ `[]`, where each pair $n_i : v_i$ denotes, that the input data contained n_i consecutive occurrences of v_i. For example, the input `1:1:1:5:5:5:5:[]` will be encoded as `3:1:4:5:[]`.

The function `enc` computes the run-length encoding of a given list of natural numbers. It is defined in terms of the auxiliary function `aux` that performs the actual encoding.

[1] `http://en.wikipedia.org/wiki/Run-length_encoding`
[2] We include zero in the natural numbers.

```
enc []     = []                                -- (E1)
enc (x:xs) = aux xs 1 x []                     -- (E2)

aux [] n v ys = ys ++ [n,v]                    -- (A1)
aux (x:xs) n v ys
  | x == v    = aux xs (n+1) x ys              -- (A2)
  | otherwise = aux xs 1 x (ys ++ [n,v])       -- (A3)
```

The function dec decodes run-length encoded data represented as a list of natural numbers. The function rep n v creates a list of length n where each element is v.

```
dec []       = []                              -- (D1)
dec [n]      = []                              -- (D2)
dec (n:v:xs) = rep n v ++ dec xs               -- (D3)

rep 0 v = []                                   -- (R1)
rep n v = v:(rep (n-1) v)                      -- (R2)
```

The function srclen computes the length of the source data from the encoded representation.

```
srclen []       = 0                            -- (S1)
srclen [n]      = 0                            -- (S2)
srclen (n:v:xs) = n + srclen xs                -- (S3)
```

Note: The pathological cases (D2) and (S2) are only there to make the corresponding functions total.

Lemmas: For the tasks below, you may use the following lemmas without proving them.

(L1) $\forall x :: \texttt{Nat} \cdot \forall xs, ys :: \texttt{[Nat]} \cdot (x\texttt{:}xs)\ \texttt{++}\ ys = x\texttt{:}(xs\ \texttt{++}\ ys)$

(L2) $\forall n, m, v :: \texttt{Nat} \cdot \forall xs, ys :: \texttt{[Nat]} \cdot$
$\texttt{aux (rep}\ m\ v\ \texttt{++ xs)}\ n\ v\ ys = \texttt{aux}\ xs\ (n\texttt{+}m)\ v\ ys$

(L3) $\forall n, v :: \texttt{Nat} \cdot \forall xs :: \texttt{[Nat]} \cdot$
$\texttt{length}\ xs\ \texttt{\%}\ 2 = 0 \implies \texttt{srclen}\ (xs\ \texttt{++}\ [n,v]) = \texttt{srclen}\ xs + n$

(L4) $\forall n, v :: \texttt{Nat} \cdot \forall xs :: \texttt{[Nat]} \cdot$
$\texttt{length}\ xs\ \texttt{\%}\ 2 = 0 \implies \texttt{length}\ (xs\ \texttt{++}\ [n,v])\ \texttt{\%}\ 2 = 0$

Task 2.1: Prove the following statement:

$$\forall n, v :: \texttt{Nat} \cdot \forall xs, ys :: \texttt{[Nat]} \cdot \texttt{length}\ ys\ \texttt{\%}\ 2 = 0 \implies$$
$$\texttt{srclen (aux (dec}\ xs)\ n\ v\ ys) = \texttt{srclen}\ xs + \texttt{n} + \texttt{srclen}\ ys$$

Hint: We recommend using strong structural induction (as explained in the exercise session) on one of the list-typed variables. Alternatively, you could use a mathematical induction on the length of one of the lists.

Task 2.2: Use the result of the previous task to prove the following statement:

$$\forall xs :: \texttt{[Nat]} \cdot \texttt{srclen (enc (dec } xs\texttt{)) = srclen } xs$$