FDD3023 ITP Homework 1

Karl Norrman, knorrman@kth.se

January 24, 2020

This is typeset in latex to make the proofs a little easier to read. The list datatype is defined as

```
datatype 'a kList = kNil | kCons of 'a  * 'a kList
```

1 kAppend

Prove that ∀l. `kAppend l kNil = l`. The definition of `kAppend` is

```
fun kAppend kNil xs = xs
  | kAppend (kCons (x, xs)) ys = kCons (x, kAppend xs ys)
```

Proof by structural induction on second argument `xs`.

Base case `xs = kNil`:

1. `kAppend kNil kNil = kNil` [by `kAppend`]
2. qed – base case

Induction step. Assume ind-hypothesis: `kAppend xs kNil = xs`. Must show that ∀ `x xs`. `kAppend (kCons (x, xs)) kNil = kCons (x, xs)`.

3. `kAppend (kCons (x, xs)) kNil = kCons (x, (kAppend xs kNil))` [by `kAppend`]
4. `kCons (x, (kAppend xs kNil)) = kCons (x, xs)` [by ind-hypothesis]
5. qed – ind-step

2 kAppend and kLength interaction

Prove that ∀ `l1 l2`. `length (append l1 l2) = length l1 + length l2`. Definitions of `kAppend` and `kLength` is as follows:

```
fun kAppend kNil xs = xs
  | kAppend (kCons (x, xs)) ys = kCons (x, kAppend xs ys)

fun kLength kNil = 0
  | kLength (kCons (x, xs)) = 1 + kLength xs
```

Proof by structural induction on first argument `l1`.

Base case `l1 = kNil`:

1. `kLength (kAppend kNil l2) = kLength l2` [by `kAppend`]
2. `kLength l2 = 0 + kLength l2` [by arithmetic]
3. `0 + kLength l2 = kLength kNil + kLength l2` [by `kLength`]
4. qed – base case

Induction step. Assume ind-hypothesis: `kLength (kAppend xs l2) = kLength xs + kLength l2`.
Must show that `kLength (kAppend (kCons (x, xs)) l2) = kLength (kCons (x, xs)) + kLength l2`.

5. `kLength (kAppend (kCons(x, xs)) l2) = kLength kCons(x, (kAppend (xs, l2)))` [by `kAppend`]
6. `kLength kCons(x, (kAppend (xs, l2))) = 1 + kLength (kAppend (xs, l2))` [by kLength]
7. `1 + kLength (kAppend (xs, l2)) = 1 + kLength xs + kLength l2` [by ind-hypothesis]
8. `1 + kLength xs + kLength l2 = kLength (kCons (x, xs)) + kLength l2` [by kLength]
9. qed – ind-step