Homework 6

Fletcher Gornick

March 8, 2021

Problem 1

```
let rec append l1 l2 =
  match l1 with
  | [] -> l2
  | h :: t -> h :: append t l2
```

Theorem 1. *For any type* `a` *and any value* `l` *of type* `a list`*, we have*

`append l []` = `l`

Domain All values of type `a list`.

Property $P($`l`$)$: `append l []` = `l`.

Inductive order $R($`l1`,`l2`$)$: `l1` is a proper suffix of `l2`.

1. `l` = `[]`.
   Left-hand side: `append [] []` = `[]`.
   Right-hand side: `[]`.
   Both sides agree.

2. `l` = `h :: t`.
   Left-hand side:

   `append (h :: t) []` = `h :: (append t [])`

   Right-hand side:

   `l` = `h :: t`

   It is thus sufficient to show that `append t []` = `t`, which is exactly $P($`t`$)$. $P($`t`$)$ is implied by the inductive hypothesis because `t` is a proper suffix of `l`. That is, $R($`t`,`l`$)$.

Problem 2 (30 points)

```
let rec sum l =
  match l with
  | [] -> 0
  | h :: t -> h + sum t

let rec append l1 l2 =
  match l1 with
  | [] -> l2
  | h :: t -> h :: append t l2
```

Theorem 2. *For any two values* `l1` *and* `l2` *of type* `int list`, *we have*

`sum (append l1 l2)` = `sum l1 + sum l2`

Domain All values of type `a list`.

Property $P($`l1`$)$: for any value `l2` of type `a list`, `sum (append l1 l2)` = `sum l1 + sum l2`.

Inductive order $R($`l1`,`l2`$)$: `l1` is a proper suffix of `l2`.

1. `l1` = `[]`.
   Left-hand side: `sum (append [] l2)` = `sum l2`.
   Right-hand side: `sum [] + sum l2` = `0 + sum l2` = `l2`.
   Both sides agree.

2. `l1` = `h :: t`.
   Left-hand side:

   `sum (append (h :: t) l2)`

   = `sum (h :: append t l2)`

   = `h + sum (append t l2)`

   Right-hand side:

   `sum (h :: t) + sum l2`

   = `(h + sum t) + sum l2`

   = `h + (sum t + sum l2)`

   It is thus sufficient to show that `sum (append t l2)` = `sum t + sum l2`, which is exactly $P($`t`$)$. $P($`t`$)$ is implied by the inductive hypothesis because `t` is a proper suffix of `l1`. That is, $R($`t`,`l1`$)$.

Bonus Problem (10 points)

```
let rec filter p l =
  match l with
  | [] -> []
  | h :: t -> if p h then h :: filter p t else filter p t
```

Theorem 3. *For any type* `a` *and any value* `p` *of type* `a -> bool`, *any value* `l` *of type* `a list`, *we have*

`filter p (filter p l)` = `filter p l`

Domain All values of type `a list`.

Property $P($`l`$)$: `filter p (filter p l)` = `filter p l`.

Inductive order $R($`l1`,`l2`$)$: `l1` is a proper suffix of `l2`.

1. `l` = `[]`.
   `filter p (filter p [])` = `filter p []`.

2. `l` = `h :: t`.

   `filter p (filter p (h :: t))`

   = `filter p (if p h then h :: filter p t else filter p t)`

The result critically depends on the value of `p h`, which is of type `bool`. We can thus proceed by case analysis.

Domain All values of type `bool`. That is, `true` and `false`.

Property *Q(b)*: If `p h` = b, then `filter p (filter p (h :: t))` = `filter p (h :: t)`

(a) `p h` = b = `true`.

```
filter p (filter p (h :: t))
= filter p (if p h then h :: filter p t else filter p t)
= filter p (if true then h :: filter p t else filter p t)
= filter p (h :: filter p t)
= if p h then h :: filter p (filter p t) else filter p (filter p t)
= if true then h :: filter p (filter p t) else filter p (filter p t)
= h :: filter p (filter p t)
```

(b) `p h` = b = `false`.

```
filter p (filter p (h :: t))
= filter p (if p h then h :: filter p t else filter p t)
= filter p (if false then h :: filter p t else filter p t)
= filter p (filter p t)
```

In both cases, it is sufficient to show that `filter p (filter p t)` = `filter p t`, which is exactly $P($`t`$)$. $P($`t`$)$ is implied by the inductive hypothesis because `t` is a proper suffix of `l`. That is, $R($`t`,`l`$)$.