CSC208 Lab: List Induction Practice

Havin, Patrick

February 7th 2023

1 Problem 1: Warm-Up

For claim:

```
for all lists l, list_length(list_tails(l)) = 1 + list_length(l).5
```

1.1 Write a proof sketch of this claim that outlines the proof's overall trajectory, as described in the reading.

First, we would derive both sides until we get 1 + list_length(tail(l)) on the left side and 1 + list_length(list_tails(tail(l))). From then, we would use Inductive Reasoning to assume that both sides have a similarity.

1.2 Write a formal proof of this claim following your sketch.

For the claim we will use Inductive hypothesis that, with a non-empty list l,

```
list_length(list_tails(tail(l))) = 1 + list_length(tail(l)).
```

Since l is non-empty, we can simplify the original equations.
The right-hand-side is evaluated as follows

```
1 + list_length(l)
--> 1 + 1 + list_length(tail(l))
```

The left-hand-side is evaluated as follows

```
list_length(list_tails(l))
--> list_length(cons(l, (list_tails)tail(l)))
--> 1 + list_length(list_tails(tail(l)))
--> 1 + 1 + list_length(tail(l))
```

This makes both sides equal, and from the induction hypothesis, the simplified equation can complete the proof.

2 Problem 2: a World Apart

For claim:

```
Claim: for any list of numbers l, list_sum(list_inc(l)) = list_length(l) + list_sum(l).
```

2.1 For the following claim, (i) develop a proof sketch and (ii) a formal proof of the claim based on that sketch.

First, we would make an inductive assumption on the claim, and evaluate the claim until we find somewhere to use the assumption.

For the formal evaluation.

Assumption: list_sum(list_inc(tail(l))) ≡ list_length(tail(l)) + list_sum(tail(l))

```
Left Hand Side
-->* list_sum(cons(head(l)+1, list_inc(tail(l))))
--> head(l) + 1 + list_sum(list_inc(tail(l)))
Using our assumption:
--> head(l) + 1 + list_length(tail(l)) + list_sum(tail(l))

Right-Hand-Side
-->* head(l) + 1 + length(tail(l)) + list_sum(tail(l))

Both sides are equivalent.
```

3 Problem 3: Building Up Facts

1. Write an implementation of a function list_snoc(l, v) that takes a list l and an element v as input and returns l but with v appended onto the end of l. In other words, snoc is "backwards-cons," adding the element to the end of the list instead of the front.

```
def list_snoc(l, v):
    return l + [v]
```

2. Using snoc, write a function list_rev(l) that reverses the list l.(Hint: what happens when you traverse l and snoc every element?)

```
def list_rev(l) :
    if is_null(l) :
        return []
    else :
        return list_snoc(list_rev(tail(l)), head(l))
```

3. Prove the following property of snoc: Claim (Snoc Length): for all lists l and values v, list_length(list_snoc(l, v)) ≡ 1 + list_length(l).

```
    list_length(list_snoc(l, v)) = 1 + list_length(l).

    For empty list
    Left-hand side
    --> list_length(list_snoc(l, v))
    -->* list_length(return [] + [v])
    --> 1
    1
    Right-hand side
    --> 1 + list_length(l).
    --> 1 + list_length([]).
```

```
--> 1
1

For non-empty lists
Left-hand side
    --> list_length(list_snoc(l, v))
    -->* list_length(return l + [v])
    --> list_length(l) + 1
    list_length(l) + 1
Right-hand side
    --> list_length(l) + 1
    list_length(l) + 1
```

The claim is true since - for empty lists and non-empty lists - the Left-hand-side and the Right-hand-side are equivalent.

4. Use the property of snoc to prove the following partial correctness property of rev:

```
Claim (Rev Length): for all lists l, list_length(list_rev(l)) = list_length(l).
```

(Hint: where do you get stuck in the rev proof? Can you get the proof goal into a form where the Snoc Length property can help you?)

```
list_length(list_rev(l)) = list_length(l)
Assumption: list_length(list_rev(tail(l))) = list_length(tail(l))

For empty lists
Left-hand side
    --> list_length(list_rev([]))
    -->* list_length([])
    -->* 0
    0
Right-hand side
    --> list_length([])
    -->* 0
    0

For non-empty lists
Left-hand side
    --> list_length(list_rev(l))
    -->* list_length(list_snoc(list_rev(tail(l)), head(l)))
    --> 1 + list_length(list_rev(tail(l)))
    Using our assumption
    --> 1 + list_length(tail(l))
    1 + list_length(tail(l))

Right-hand side
    --> list_length(l)
    -->* 1 + list_length(tail(l))
    1 + list_length(tail(l))
```

For all cases, each side is equivalent to the other. Thus, the claim is correct.