Class 16: Structural Induction

Schedule

Problem Set 6 is due **tomorrow at 6:29pm**. Make sure to read the corrected version of Problem 7.

Lists

Definition. A *list* is an ordered sequence of objects. A list is either the empty list (λ), or the result of prepend(e, l) for some object e and list l.

$$\begin{aligned}
\textit{first}(\text{prepend}(e, l)) &= e \\
\textit{rest}(\text{prepend}(e, l)) &= l \\
\textit{empty}(\text{prepend}(e, l)) &= \textbf{False} \\
\textit{empty}(\textbf{null}) &= \textbf{True}
\end{aligned}$$

Definition. The *length* of a list, p, is:

$$\begin{cases} 0 & \text{if } p \text{ is } \textbf{null} \\ \textit{length}(q) + 1 & \text{otherwise } p = \textit{prepend}(e, q) \text{ for some object } e \text{ and some list } q \end{cases}$$

```python
def list_length(l):
    if list_empty(l):
        return 0
    else:
        return 1 + list_length(list_rest(l))
```

Prove: for all lists, p, `list_length(p)` returns the length of the list p.

Concatenation

Definition. The *concatenation* of two lists, $p = (p_1, p_2, \cdots, p_n)$ and $q = (q_1, q_2, \cdots, q_m)$ is

$$(p_1, p_2, \cdots, p_n, q_1, q_2, \cdots, q_m).$$

Provide a *constructuve* definition of *concatenation.*

Note that prepend(p, q) is not a good idea for two reasons. If we use this definition, then the first element of the constructed list will be the object (list) p (as a whole) rather than the first element p_1 of the list p. Also, if we want to *only* define lists of *specific* objects, for example integers, we can still use the same recursive/constructive definition of lists by substituting "object" with "integer", but in that case prepend(p, q) will not even well defined, as it can only accept integers as first input.

Structural Induction

To prove proposition $P(x)$ for element $x \in D$ where D is a recursively-constructed data type, we do two things:

1. Show $P(x)$ is true for all $x \in D$ that are defined using base cases.
2. Show that if $P(y)$ is true for element y and x is constructed from y using any "construct case" rules, then $P(x)$ is true as well.

Comparing Various forms of Induction

	Regular Induction	**Invariant Principle**	**Structural Induction**
Works on:	natural numbers	state machines	data types
To prove $P(\cdot)$	*for all natural numbers*	*for all reachable states*	*for all data type objects*
Prove **base case(s)**	$P(0)$	$P(q_0)$	P(base object(s))
and **inductive step**	$\forall m \in \mathbb{N}.$	$\forall (q, r) \in G.$	$\forall s \in$ *Type.*
	$P(m) \implies P(m+1)$	$P(q) \implies P(r)$	$P(s) \implies P(t)$
			$\forall t$ constructable from s

Prove. For any two lists, p and q, length$(p + q) =$ length$(p) +$ length(q).

