Assignment 1 by Lucas Karlsson

1. Prove using induction that, for every finite alphabet $\Sigma, \forall n \in N. \mid \Sigma^n \mid = \mid \Sigma \mid^n$.

Solution:
Firstly, we need to show that $\mid S \cdot \Sigma \mid = \mid S \mid \cdot \mid \Sigma \mid$, where $\cdot$ is the concatenation operator.

By definition: $s \in S$ and $\sigma \in \Sigma$ this implies that also the string $s \cdot \sigma$ is in $S \cdot \Sigma$ and this string is different for every combination of s and σ

Now we can show $\mid \Sigma^n \mid = \mid \Sigma \mid^n$.

Base case: $n = 0$

$\Sigma^0 = \{\epsilon\}$, $\mid \Sigma^0 \mid = 1$

The length of a list with a empty string inside is always one, by definition. And anything to the power of zero will always be one. Giving us the following equality:

$$\mid \Sigma^0 \mid = \mid \Sigma \mid^0$$

We now assume that $\mid \Sigma^k \mid = \mid \Sigma \mid^k$ for every n=k

Induction when n = k+1

$\Sigma^{k+1} = \Sigma^k \cdot \Sigma$ implies that $\mid \Sigma^{k+1} \mid = \mid \Sigma^k \cdot \Sigma \mid$ and we established before that $\mid \Sigma^k \cdot \Sigma \mid = \mid \Sigma^k \mid \cdot \mid \Sigma \mid$ and $\mid \Sigma \mid^k \cdot \mid \Sigma \mid^1 = \mid \Sigma \mid^{k+1}$

By the inductive hypothesis we now have:

$$\mid \Sigma^{k+1} \mid = \mid \Sigma \mid^{k+1}$$

$\square$

2. Define a language S containing words over the alphabet $\Sigma = \{a, b\}$ inductively given a set of rules.

Solution 1:
$\#_a$ () = 0
$\#_a$ (cons(a,as)) = 1 + $\#_a$ (as)
$\#_a$ (cons(_,as)) = $\#_a$ (as)

$\#_b$ () = 0
$\#_b$ (cons(b,bs)) = 1 + $\#_b$ (bs)
$\#_b$ (cons(_,bs)) = $\#_b$ (bs)

Solution 2:
Prove that $\forall w \in S.\ \#_a(w) = 2\#_b(w)$ using induction!

To do this first we need to show that $\#_a(auavb) = 2 + \#_a(u) + \#_a(v)$ and this can be done using a lemma that you can prove by induction. The lemma:

$$\forall u, v \in \Sigma^*.\#_a(uv) = \#_a(u) + \#_a(v)$$

In other words, this is a three-part problem starting by using induction to prove the previous lemma, then using this lemma to show that $\#_a(auavb) = 2 + \#_a(u) + \#_a(v)$ holds. Then using all of this information to prove, again using induction, our first and root problem.

PART1 Basecase:
$\#_a(uv)$ where u = ϵ this is the same as writing $\#_a(\epsilon) + \#_a(v)$ because we know $\#_a(\epsilon) = 0$, by definition the empty string cannot contain any a:s

We now assume $P(L) = \#_a(uv) = \#_a(u) + \#_a(v)\ \forall u, v \in S$ where $L =\mid u \mid$ This holds because that is how we have defined our function.

Induction, now we need to show that it holds for $P(L+1)$, when $P(L+1) = \#_a(u'uv)$ where u' $\in \Sigma$.

By our previous defintion of our $\#_a$ function we know that we can write $\#_a(u'uv)$ as $\#_a(u') + \#_a(uv)$ and $\#_a(uv)$ as $\#_a(u) + \#_a(v)$ giving us: $P(L+1) = \#_a(u') + \#_a(u) + \#_a(v)$ and $P(L+1) = \#_a(u'u) + \#_a(v)$ where $\mid u'u \mid= L + 1$ by our previous assumption we are now done.

$\square$

We can now show $\#_a(auavb) = 2 + \#_a(u) + \#_a(v)$ with our previous proof. Seperating $\#_a(auavb)$ into multiple function calls give us $\#_a(a) + \#_a(u) + \#_a(a) + \#_a(v) + \#_a(b)$ which is the same as $1 + \#_a(u) + 1 + \#_a(v) + 0$ and this is equal to $2 + \#_a(u) + \#_a(v)$ and PART2 is now done.

PART3: The Final Proof. Now with all the information we know we can finaly use induction to prove $\forall w \in S.\ \#_a(w) = 2\#_b(w)$

Basecase: $w = uv = \epsilon$
$\#_a(\epsilon) = 0$, trivial because there cannot be any a:s in an empty string.
$2\#_b(\epsilon) = 0$, again trivial because no b:s in an empty string.
$0 = 0$ and we are done with the basecase.

We now assume that $\#_a(\mathbf{u}) = 2\#_b(\mathbf{u}) \wedge \#_a(\mathbf{v}) = 2\#_b(\mathbf{v}) \wedge \#_a(\mathbf{w}) = 2\#_b(\mathbf{w})$ where **u,v,w** $\in$ S this also implies that $\#_a(uv) = 2\#_b(uv)$ is true.

Induction
We now need to prove this for ”auavb” and ”buavaw” because they are also by definition a part of the language S.

$\#_a(auavb) = 2 + \#_a(uv)$
$2\#_b(auavb) = 2(1 + \#_b(uv)) = 2 + \#_a(uv)$
Works!

$\#_a(buavaw) = 2 + \#_a(uvw)$
$2\#_b(buavaw) = 2(1 + \#_b(uvw)) = 2 + \#_a(uvw)$
Works!

Our function $\#_a(w) = 2\#_b(w)$ holds for all the contents in our language S.

$\square$

3. Let $\Sigma = \{0\}$ and define $f, g, h \in \Sigma^* \rightarrow N$

Solution 1:
I solved this problem by recreating the functions in Haskell, a functional programming language very well shaped for using recursion. my functions looked like this and all of them where of the type String → Int

g () = 1
g ('0':w) = length w + g w - h w

h () = 0
h ('0':w = length w + g w

f () = 1
f ('0':w) = h w + 2 * g w

Computing the values gave me the answers:

f(00) = 3, g(00) = 1, h(00) = 2

f(000) = 4, g(000) = 1, h(000) =3

f(0000) = 5, g(0000) = 1, h(0000) = 4.

Solution 2:
For this problem we need to prove $\forall n \in N.f(0^n) = 1 + n$ by first proving that $\forall n \in N.g(0^n) = 1 \wedge h(0^n) = n$.

Basecase: n=0

$g(0^0) = 1 \wedge h(0^0) = 0$ the same as True ∧ True which is true.

We now assume n = k which gives us $g(0^k) = 1 \wedge h(0^k) = k$

Induction when n = k+1 which gives us $g(0^{k+1}) = 1 \wedge h(0^{k+1}) = k + 1$

If we look at the first part of the boolean expression $g(0^{k+1})$ evaluating this using the defined function g give us $g(00^{k+1}) =\mid 0^{k+1} \mid + g(0^k) - h(0^k)$ and this is the same as $k + 1 - k = 1$ and $1 = 1$ which gives us true.

Doing the same thing for the second part of the boolean expression $h(0^{k+1})$ using the function h this time gives us $h(00^k) =\mid 0^k \mid + g(0^k)$ which is the same as $k + 1$ and $k + 1 = k + 1$ and the second part is also true.

$True \wedge True = True$

□

The final part is now trivial, we can easily prove $\forall n \in N . f(0^n) = 1 + n$ because we know that $f(0^n)$ is equal to $h(0^n) + 2g(0^n)$ by the definition of our function f. We can also see that $h(0^n) = n - 1$ and $2g(0^n) = 2 * 1$ summing these up gives us, and we are now done!

$$f(0^n) = n + 1$$

$\square$