Part 1 – assignment 5

Nitzan Hochman 316264845 Dor Lebel 205872807

Q1

1.a

Two lazy lists lzl1 lzl2 are equivalent if:

- Lzl1 and lzl2 are empty

Or

- **Lzl1**(T) = Pair (T1 , (f : [empty -> lzl (T)]))
  Lzl2(D) = Pair (D1 , (g : [empty -> lzl (D)]))
  Such that D = T and T1 = D1 and in the i'th call to f the result is equivalent to the result of the i'th call to g for every i.

1.b

Even-square-1 applies lzl-filter (with the predicate that returns only the even numbers on the list) on the lazy list that is returned from lzl-map which applies the square function on all the numbers of the list that is returned from 'integers-from' which returns a list of all natural numbers starting from 0.

Even-square-2 applies lzl-map which applies the square function on all the numbers of the list that is returned from lzl-filter (with the predicate that returns only the even numbers on the list) on the lazy list that is returned from 'integers-from' which returns a list of all natural numbers starting from 0.

Therefore, the lists returned are a list of the even squared natural numbers and the squared even natural numbers.

By our definition:

- T=D - Both lists always return a number
- T1 = D1 - both start with 0.
- in the i'th call to even-square-1 we will get the i'th even squared number and in the i'th call to even-square-2 we will get the i'th even number squared which are the same.

Q2

a

$$f\ is\ equivalent\ to\ f\$\ (its\ success - fail - continuations\ version)$$

$$\Leftrightarrow$$

1) $if\ f : [\,D1 * D2 * \ldots * Dn \to T]\ then\ f\$: [\,D1 * D2 * \ldots * Dn * g * h \to S]\ when\ g\ and\ f\ are\ functions$

$$and$$

2) $if\ f\$(\,x1 \ldots\ xn, succ\ , fail\,)\ is\ successful\ then\ f\$\ (x1 \ldots\ xn, succ\ , fail\,) = succ(f(x1 \ldots\ xk))$

$$else\ \ f\$\ (\,x1 \ldots\ xn, succ\ , fail\,) = fail(\,f(x1 \ldots\ xk)\,)$$

d

Get-value is of type [List<Pair<Symbol,T>> * Symbol -> T | 'fail] and get-value$ is of type

[List<Pair<Symbol,T>> * Symbol * [T->T1] *[Empty->T2]] -> T1 | T2] so (1) is satisfied.

Now we'll show that (2) is satisfied too.

When get-value$ is successful, it means that a value v is found, and the result of the function succ on that value is returned. This means, that the function get-value is successful too and it will return v, which in turn will mean that get-value$ (list, key, succ , fail) = succ (v) = succ (get-value(list, key)).

When get-value$ has failed, it means that no value has been found, and so the result of fail is returned. This means, that the function get-value is unsuccessful as well and returns a failure too.

Q3

1.a

Equations = [t (s (s) , G , H , p, t (E) , s) = t (s (H) , G , p , p, t (E) , K)] , sub = {}

```
if the predicate symbols and the number of arguments are the same: eq'_1 =
(p(t_1,...,t_n) = p(s_1, ..., s_n)):
split eq'_1 into equations: equations = equation U (t_i = s_i) for i=1..n.
```

Equations = [s(s) = s(H) , G = G, H = p , p = p , t(E) = t(E) , s=K] , sub = {}

```
Equations = Equations ∘ {s = H}
```

Equations = [s = H , G = G, H = p , p = p , t(E) = t(E) , s = K] , sub = {}

```
sub = sub ∘ {s = H}
```

Equations = [G = G, H = p , p = p , t(E) = t(E) , s = K] , sub = { s = H }

```
if the equation is the same variable on both sides, continue.
```

Equations = [H = p , p = p , t(E) = t(E) , s = K] , sub = { s = H }

$\{H\ =\ p\,\} \circ \{\,s\ =\ H\} \to \{s\ =\ p\}$

```
FAIL - both sides are different constant symbols
```

1.b

Equations = [g (c , v (U) , g , G , U , E , v (M)) = g (c , M , g ,v (M) , v (G) , g , v (M))]

Sub = {}

```
if the predicate symbols and the number of arguments are the same: eq'_1 =
(p(t_1,...,t_n) = p(s_1, ..., s_n)):
split eq'_1 into equations: equations = equation U (t_i = s_i) for i=1..n
```

Equations = [c = c , v(U) = M , g = g , G = v(M) , U =v(G) , E = g , v(M) = v(M)], sub={}

```
if both sides are the same constant symbol then continue
```

Equations = [v(U) = M , g = g , G = v(M) , U =v(G) , E = g , v(M) = v(M)], sub={}

```
sub = sub o { v( U ) = M }
```

Equations = [g = g , G = v(M) , U =v(G) , E = g , v(M) = v(M)], sub={ v(U) = M }

```
if both sides are the same constant symbol then continue
```

Equations = [G = v(M) , U =v(G) , E = g , v(M) = v(M)], sub={ v(U) = M }

```
{G = v ( M )} o sub = {G = v ( v( U ) )}

sub = sub o {G = v ( v( U ) )}
```

Equations = [U = v(G) , E = g , v(M) = v(M)], sub={ v(U) = M , G = v (v (U))}

```
{U = v ( G )} o sub = { U = v(v(v(U)))}

FAIL - occurs check
```

1.c

Equations = [s ([v | [[v | V] | A]]) = s ([v | [v | A]])] , Sub = {}

```
if the predicate symbols and the number of arguments are the same: eq'_1 =
(p(t_1,...,t_n) = p(s_1, ..., s_n)):
split eq'_1 into equations: equations = equation U (t_i = s_i) for i=1..n
```

Equations = [[v | [[v | V] | A]] = [v | [v | A]]], sub={}

```
the predicate symbols and the number of arguments are the same: eq'_1 =
(p(t_1,...,t_n) = p(s_1, ..., s_n)):
split eq'_1 into equations: equations = equation U (t_i = s_i) for i=1..n,
continue.
```

Equations = [v = v , [[v | V] | A]= [v | A]], sub={}

```
if both sides are the same constant symbol then continue
```

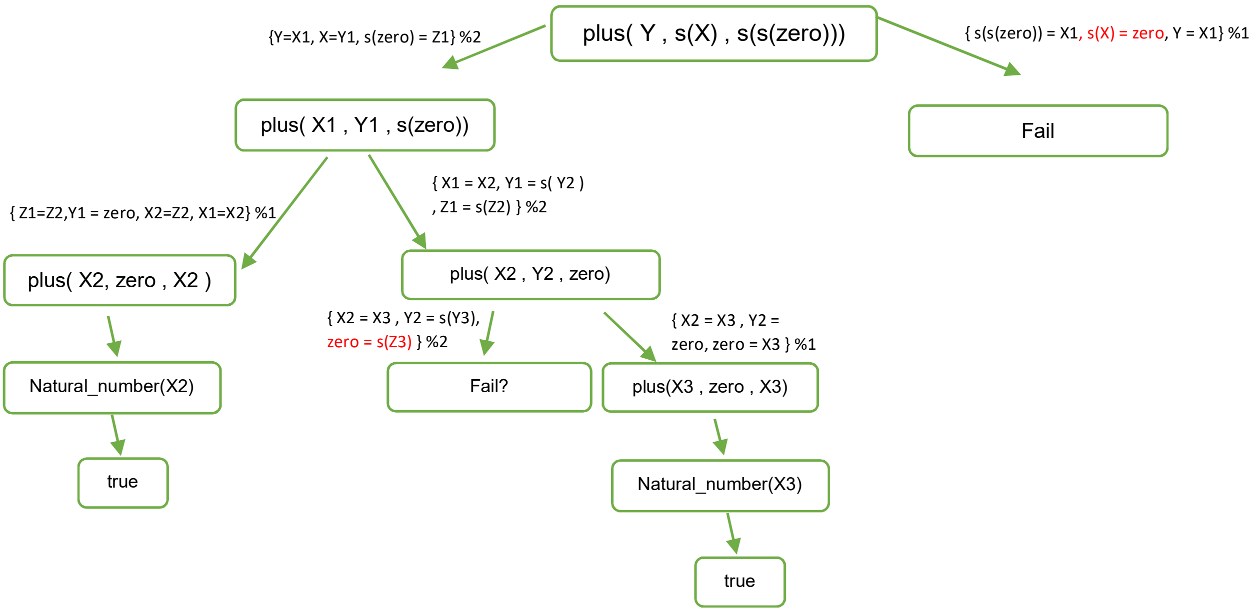
Equations = [[[v | V] | A]= [v | A]], sub={}

```
the predicate symbols and the number of arguments are the same: eq'_1 =
(p(t_1,...,t_n) = p(s_1, ..., s_n)):
split eq'_1 into equations: equations = equation U (t_i = s_i) for i=1..n,
continue.
```

Equations = [[v | V] = v , A = A], sub={}

```
FAIL - Constant symbol value v can't be equal to list predicate
```

2.a



2.b

Y = zero, X = s(zero)

Y = s(zero), X = zero

2.c

This is a success proof tree because it has at least one successful path.

2.d

This tree is finite because it has no infinite path .