Question 1:

b.

given lists lst1 and lst2 and a continuation procedure cont, we will prove using induction on length of the first list - (len(lst1)) and using applicative-eval (a - e) operation semantics that

$(cont\ (append\ lst1\ lst2)) = (append\$\ lst1\ lst2\ cont).$

Base: len(lst1) = 0

$$a - e[cont(append('()\ lst2)] => a - e[(cont\ lst2)]$$

$$a - e[append\$\ ('()\ lst2\ cont)] => a - e[(cont\ lst2)]$$

Hypothesis: for every $k\ in\ \mathbb{N}$, len(lst1) = k, we assume that

$$(cont\ (append\ lst1\ lst2)) = (append\$\ lst1\ lst2\ cont).$$

Step: we prove that for k+1, len(lst1') = k+1 that

$$a - e[(cont\ (append\ lst1' lst2))] = \\ > a - e\left[\left(cont\ \left(cons\ (car\ lst1')\ (append\ (cdr\ lst1')\ lst2)\right)\right)\right] *$$

$$(cont\ (append\ lst1'\ lst2)) = (append\$\ lst1'\ lst2\ cont) :$$

$$a - e[(append\$\ lst1'\ lst2\ cont)] => a - e[(append\$\ (cdr\ lst1')\ lst2) \\ (lambda\ (res)\ (cont\ (cons((car\ lst1')\ res))))]$$

Len(lst1') = k+1, len(cdr lst1') = k, because we take the list without the first element.

From using the induction hypothesis:

$$a - e[(append\$\ (cdr\ lst1')lst2)\ (lambda\ (res)\ (cont\ (cons((car\ lst1')\ res))))]$$

$$= a - e[(lambda\ (res)\ (cont\ (cons((car\ lst1')\ res))) \\ (append(cdr\ lst1')\ lst2))]$$

$$=> a - e\left[\left(cont\ \left(cons\ (car\ lst1')\ (append\ (cdr\ lst1')\ lst2)\right)\right)\right] **$$

From * and ** we prove that

$$(cont\ (append\ lst1' lst2)) = (append\$\ lst1' lst2\ cont).$$

Question 3:

3.1

1. $\mathbf{unify}[\mathbf{t}(\mathbf{s}(\mathbf{s}), \mathbf{G}, \mathbf{s}, \mathbf{p}, \mathbf{t}(\mathbf{K}), \mathbf{s}), \mathbf{t}(\mathbf{s}(\mathbf{G}), \mathbf{G}, \mathbf{s}, \mathbf{p}, \mathbf{t}(\mathbf{K}), \mathbf{U})]$

Initial:

1. Substitution = {}, Equation = {t(s(s), G, s, p, t(K), s) = t(s(G), G, s, p, t(K), U)}

We choose an equation: t(s(s), G, s, p, t(K), s) = t(s(G), G, s, p, t(K), U)

Both have the same number of arguments and predicate, we divide them in their small equations.

2. Substitution = {}, Equation = {s(s) = s(G), s = U}

We choose an equation: s(s) = s(G)

Both have the same number of arguments and predicate, we divide them in their small equations.

3. Substitution = {}, Equation = {s=G, s = U}

We choose an equation: s=G

We don't have compound expression so add to substitution:

$$\{\ \} \circ \{s = G\} = \{s = G\}$$

4. Substitution = {}, Equation = {s = U}

We choose an equation: s=U

We don't have compound expression so add to substitution:

$$\{s = G\} \circ \{s = U\} = \{s = G, s = U\}$$

In the end we get Substitution : $\{s = G, s = U\}$.

2. $\mathbf{unify}[\mathbf{g}(\mathbf{l}, \mathbf{M}, \mathbf{g}, \mathbf{G}, \mathbf{U}, \mathbf{g}, \mathbf{v}(\mathbf{M})), \mathbf{g}(\mathbf{l}, \mathbf{v}(\mathbf{U}), \mathbf{g}, \mathbf{v}(\mathbf{M}), \mathbf{v}(\mathbf{G}), \mathbf{g}, \mathbf{v}(\mathbf{M}))]$

Initial:

1. Substitution = {}, Equation = { $\mathrm{g(l, M, g, G, U, g, v(M))= g(l, v(U), g, v(M), v(G), g, v(M))}$ }

We choose an equation: t(s(s), G, s, p, t(K), s) = t(s(G), G, s, p, t(K), U)

Both have the same number of arguments and predicate, we divide them in their small equations.

2. Substitution = {}, Equation = {M=v(U), G=v(M), U=v(G)}

We choose an equation: M=v(U)

We don't have compound expression so add to substitution

$$\{\,\} \circ \{M = v(U)\} = \{M = v(U)\}$$

3. Substitution = $\{M = v(U)\}$, Equation = { G=v(M), U=v(G)}

We choose an equation: G=v(M),

We don't have compound expression so add to substitution:

$$\{M = v(U)\} \circ \{G = v(M)\} = \{M = v(U), G = v(v(U))\}$$

4. Substitution = $\{M = v(U), G = v(v(U))\}$, Equation = {U=v(G)}

We choose an equation: U=v(G)

We don't have compound expression so add to substitution:

$$U = \big(v(G)\big) = v\Big(v\big(v(U)\big)\Big)\, != U$$

We get failure when we perform unification on wanted equation.

3. $\mathbf{unify}[\mathbf{m}(\mathbf{M},\mathbf{N}), \mathbf{n}(\mathbf{M},\mathbf{N})]$

Initial:

1. Substitution = {}, Equation = $\{m(M, N) = n(M, N)\}$

We choose an equation: $m(M, N) = n(M, N)$

We have different predicates m != n so we get failure.

4. $\mathbf{unify}[\mathbf{p}([\mathbf{v} \mid [\mathbf{V} \mid \mathbf{VV}]]), \mathbf{p}([[\mathbf{v} \mid \mathbf{V}] \mid \mathbf{VV}])]$

Initial:

1. Substitution = {}, Equation = { $\mathrm{p}([\mathrm{v} \mid [V \mid VV]]) = \mathrm{p}\big([[\mathrm{v} \mid \mathrm{V}] \mid \mathrm{VV}]\big)$}

We choose an equation: $\mathrm{p}([\mathrm{v} \mid [V \mid VV]]) = \mathrm{p}\big([[\mathrm{v} \mid \mathrm{V}] \mid \mathrm{VV}]\big)$

Both have the same number of arguments and predicate, we divide them in their small equations.

2. Substitution = {}, Equation = $\big\{[\mathrm{v} \mid [V \mid VV]] = [[\mathrm{v} \mid \mathrm{V}] \mid \mathrm{VV}]\big\}$

We choose an equation: $[v \mid [V \mid VV]] = [[\mathrm{v} \mid \mathrm{V}] \mid \mathrm{VV}]$

Both have the same number of arguments and predicate, we divide them in their small equations.

3. Substitution = {}, Equation = $v = [v \mid V], [V \mid VV] = VV$

We choose an equation: $v = [v \mid V]$,

v != [v | V], because v does exist in both sides of the equation so we get failure.

5. $\boldsymbol{unify}[\boldsymbol{g}([\boldsymbol{T}]), \boldsymbol{g}(\boldsymbol{T})]$

Initial:

1. Substitution = {}, Equation = $\{g([T]) = \ g(T)\}$

We choose an equation: g([T])= g(T)

Both have the same number of arguments and predicate, we divide them in their small equations.

2. Substitution = {}, Equation = $\{[T] = T\}$

We choose an equation: $[T] = T$

[T] != T, because T does exist in both sides of the equation so we get failure.

3.3

le(X, s(zero)), times(X, s(s(zero)), Y).

%l1 {X = zero, s(zero) = X_1}

le(zero, X_1), times(X, s(s(zero)), Y)

fail

%t1

{X = zero, Y = zero}

natural_number(s(s(zero)))

true

{X = zero, Y = zero}

%l2 {X = S(X_2), s(Z_2) = s(zero)}

le(X, s(zero)), times(X, s(s(zero)), Y)

%l2

fail

le(X_2, zero), times(s(X_2), s(s(zero)), Y)

fail

%l1

{X_2 = zero, Z_2 = zero,
X = s(zero)}

times(s(zero), s(s(zero)), Y)

Fail

%t2

{s(X_3) = s(zero), Y_2 = s(s(zero)),
Z_3 = Y}

times(zero, s(s(zero)), XY_1), plus(XY_1, s(s(zero)), Y)

Fail

%t1 {XY_1 = zero}

natural_number(s(s(zero)), plus(zero, s(s(zero)), Y)

Fail

%p2 {s(Y_1) = s(s(zero)),
s(Z_4) = Y}

plus(zero, s(zero), Z_4)

fail

%p2

{s(zero) = s(Y_4),
Z_5 = S(Z_4)}

plus(zero, zero, Z_5)

%p1

{zero = Z_5}

fail

natural_number(zero)

true

{ Y: s(s(zero)), X: s(zero) }