1.b:

Let's redefine the equivalence of the higher-order function g and its CPS version g$ as follows: For any CPS-equivalent parameters f1…fn and f1$...fn$, (g$ f1$...fn$ cont) is CPS-equivalent to (cont (g f1…fn)). Using this definition, we demonstrate that pipe$ is equivalent to pipe by induction on the length of the list.

Base case (N=1):

1. (cont (pipe(f1$))) = (cont f1$)
2. (pipe$ f1$ cont) = (cont (lambda (x cont2) (f1$ x cont2))) = (cont f1$)

Induction step:

Assume that (pipe$ f1$ … fn$ cont) = (cont (pipe f1$ … fn$)) holds true.

Now consider the case for n+1:

1. (pipe$ (f1$ … fn$ fn+1$ cont))
2. = (pipe$ f2$ … fn+1$ (lambda (f2-n$) (cont (lambda (x cont2) (f1$ x (lambda (res) (fn2-n$ res cont2)))))))
3. = (lambda (f2-n$) (cont (lambda (x cont2) (f1$ x (lambda (res) (fn2-n$ res cont2))))))
4. = (pipe f2$ … fn+1$)
5. = (cont (lambda (x cont2) ((pipe f2$ … fn$1) x (lambda (res) (fn2-n$ res cont2)))))
6. = (cont (f2-n$ (pipe f1$ f2$ … fn+1$)))
7. = (cont (pipe f1$ … fn+1$))

By following these steps, we establish the equivalence of pipe$ and pipe.

2.d:

reduce1-lzl: for a reduce of a finite lazy list

reduce2-lzl: for a reduce of one specific prefix of a given infinite lazy list

reduce3-lzl: for a reduce of each prefix of an infinite lazy list

2.g:

Advantage: It can be used for any level of approximation, unlike the pi-sum method which is limited to a fixed 'b' value.

Disadvantage: It produces many closures.

3.1 Unification:

Expressions to Unify:

E1: x(y(y), T, y, z, k(K), y)

E2: x(y(T), T, y, z, k(K), L)

Equations:

[x(y(y), T, y, z, k(K), y) = x(y(T), T, y, z, k(K), L)]

Substitution: `{}`

Decompose the Expressions:

Both sides are compound terms with the same functor `x` and arity (6). Decompose into:

[y(y) = y(T), T = T, y = y, z = z, k(K) = k(K), y = L]

Equation: `y(y) = y(T)`

Case 3: Both sides are compound terms with the same functor `y` and arity (1). Decompose into:[y=T]

Equation: `T = T`

Case 1: Both sides are identical. No changes needed.

Equation: `y = y`

Case 1: Both sides are identical. No changes needed.

Equation: `z = z`

Case 1: Both sides are identical. No changes needed.

Equation: `k(K) = k(K)`

Case 1: Both sides are identical. No changes needed.

Equation: `y = L`

Case 2: One side is a logical variable. Add to substitution:

Substitution: {L = y}

Applying the substitution `{L = y}` to the remaining equations:
[y = T, L = y]

Equation: `y = T`

Case 2: One side is a logical variable. Add to substitution:

Substitution: {L = y, T = y}

Final Substitution: {L = y, T = y}

2:

Initialization:

Equations: f(a, M, f, F, Z, f, x(M)) = f(a, x(Z), f, x(M), x(F), f, x(M))

Substitution: {}

Step-by-Step Unification:

Step 1: Compare f(a, M, f, F, Z, f, x(M)) with f(a, x(Z), f, x(M), x(F), f, x(M)).

Since both terms have the same function symbol f with the same arity (7), decompose the terms into smaller equations:

a = a

M = x(Z)

f = f

F = x(M)

Z = x(F)

f = f

x(M) = x(M)

Step 2: Solve the simpler equations:

a = a (trivially true)

M = x(Z)

f = f (trivially true)

F = x(M)

Z = x(F)

f = f (trivially true)

x(M) = x(M) (trivially true)

Step 3: Apply substitutions iteratively:

Start with M = x(Z) and substitute it into F = x(M) and Z = x(F):

F = x(x(Z))

Z = x(F) becomes Z = x(x(x(Z)))

Simplify further by substituting back:

Z = x(x(x(Z))) leads to a circular dependency and cannot be resolved directly in the context of simple unification.

Result:

The unification fails due to the circular dependency introduced by Z = x(x(x(Z))).

Therefore, unification of f(a, M, f, F, Z, f, x(M)) and f(a, x(Z), f, x(M), x(F), f, x(M)) fails because of an unsolvable circular dependency in the equations.

3:

Initialization:

Equations: t(A, B, C, n(A, B, C), x, y) = t(a, b, c, m(A, B, C), X, Y)

Substitution: {}

Step-by-Step Unification:

Step 1: Compare t(A, B, C, n(A, B, C), x, y) with t(a, b, c, m(A, B, C), X, Y).

Since both terms have the same function symbol t with the same arity (6), decompose the terms into smaller equations:

A = a

B = b

C = c

n(A, B, C) = m(A, B, C)

x = X

y = Y

Step 2: Solve the simpler equations:

A = a

B = b

C = c

n (A, B, C) = m (A, B, C) (unifiable only if n = m and arguments are identical)

x = X

y = Y

Step 3: Evaluate the complex equation n(A, B, C) = m(A, B, C):

Since n and m are different function symbols, the equation n(A, B, C) = m(A, B, C) cannot be unified.

Result:

The unification fails because the function symbols n and m do not match, making the equation n(A, B, C) = m(A, B, C) unsolvable.

Therefore, unification of t(A, B, C, n(A, B, C), x, y) and t(a, b, c, m(A, B, C), X, Y) fails because the function symbols n and m are different and cannot be unified.

4.

Initialization:

Equations: z(a(A, x, Y), D, g) = z(a(d, x, g), g, Y)

Substitution: {}

Step-by-Step Unification:

—

Step 1: Compare z(a(A, x, Y), D, g) with z(a(d, x, g), g, Y).

Since both terms have the same function symbol z with the same arity (3), decompose the terms into smaller equations:

a(A, x, Y) = a(d, x, g)

D = g

g = Y

Step 2: Solve the simpler equations:

Equation 1: a(A, x, Y) = a(d, x, g)

Since both terms have the same function symbol a with the same arity (3), decompose the terms further:

A = d

x = x (trivially true)

Y = g

This results in the substitutions: {A = d, Y = g}

Equation 2: D = g

This adds the substitution: {D = g}

Equation 3: g = Y

Since we already have Y = g, this is consistent with our current substitution.

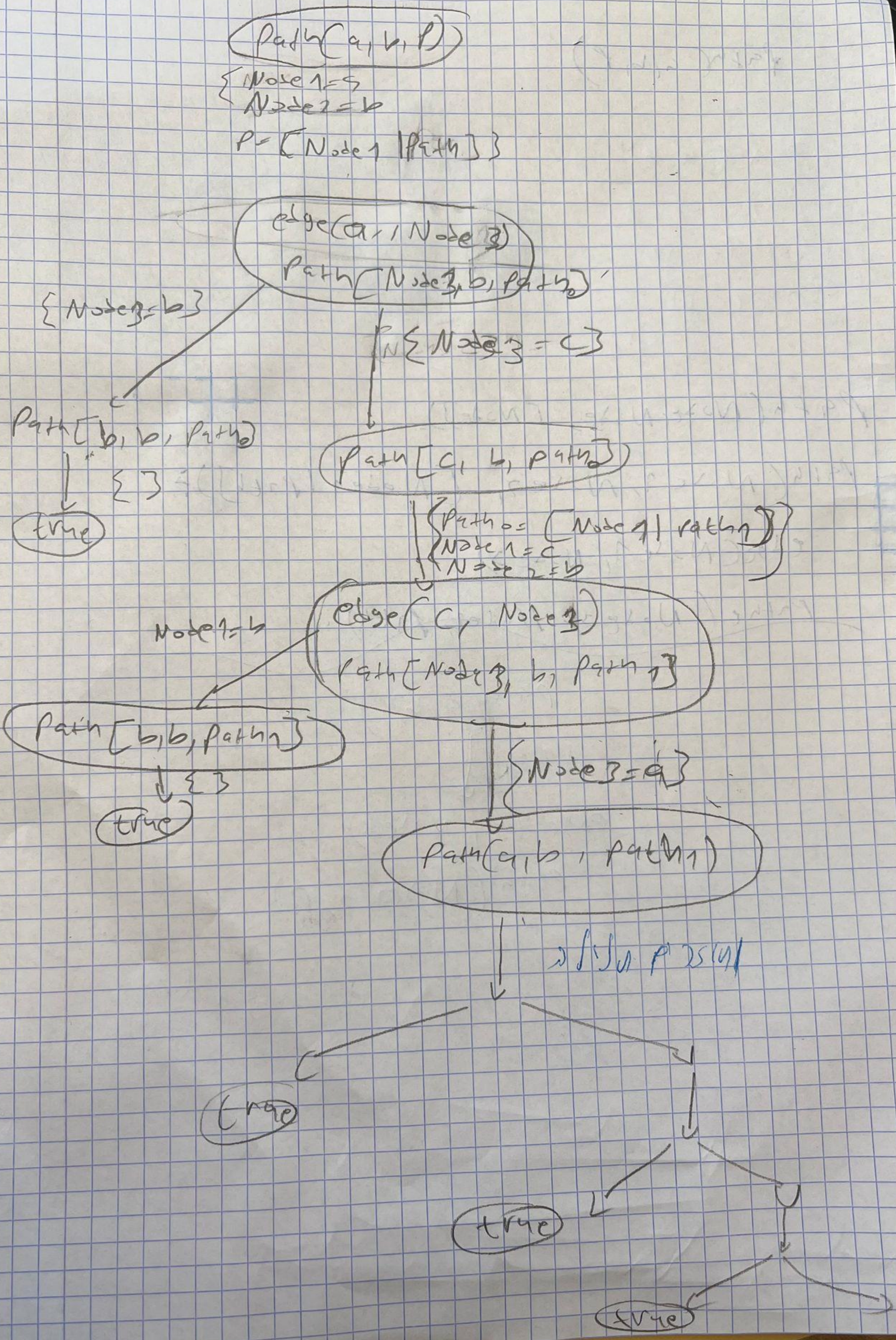
Result:

Combining all substitutions, we get: {A = d, D = g, Y = g}

Therefore, the most general unifier (MGU) for z(a(A, x, Y), D, g) and z(a(d, x, g), g, Y) is {A = d, D = g, Y = g}

3.3 : its not a finite tree but this tree is a success tree

And here is the tree:


path(a, b, P)
{Node1=a
Node2=b
P=[Node1|Path]}
edge(a, Node3)
path[Node3, b, Path0]
{Node3=b}
{Node3=c}
path[b, b, Path0]
{}
true
path[c, b, path0]
{Path0=[Node1|Path1]
Node1=c
Node2=b}
edge(c, Node3)
path[Node3, b, Path1]
Node1=b
path[b,b,Path1]
{}
true
{Node3=a}
path(a,b, path1)
אינסוף
true
true
true