Assignment 4

# Question 1

## Section b

We will prove using induction on the number of the operands ():

Base Case:

:

The function will get (the operand) and return it, so we will get .

The function will get and , so will be empty and will be , therefore the function will return. In conclusion, they are equivalent.

Induction Hypothesis: We assume that for operands the functions are equivilant.

Induction Step:

Assume that we give the functions operands.

will return .

will start by checking if is not empty (and indeed it is not), so it will return the value that comes back the call:

# Question 2

## Section d

We'll use reduce1-lzl when we are only interested in the final result and when we know the list is finite.

We'll use reduce2-lzl when the list may be infinite and/or when we are interested only in some prefix of it (for example when approximating some values up to a certain precision).

We'll use reduce3-lzl in the same scenario of reduce2-lzl, but this time we are also interested in the process (to draw a graph, maybe).

## Section g

Advantages are that when using generate-pi-approximations we only calculate the desired values when we want them - no redundant calculations are made. We also do not use recursion - and thus saving a lot of space (on the call stack for example). Disadvantages may be that pi-sum gives us the ability to decide when to stop calculating depending on the actual fraction and not on the number of steps - thus it is more predictable and flexible.

# Question 3

## Section 1

1.

2.

3.

4.

## Section 3

edge(a,b). %e1

edge(a,c). %e2

edge(c,b). %e3

edge(c,a). %e4

path(Node1, Node2, Path)

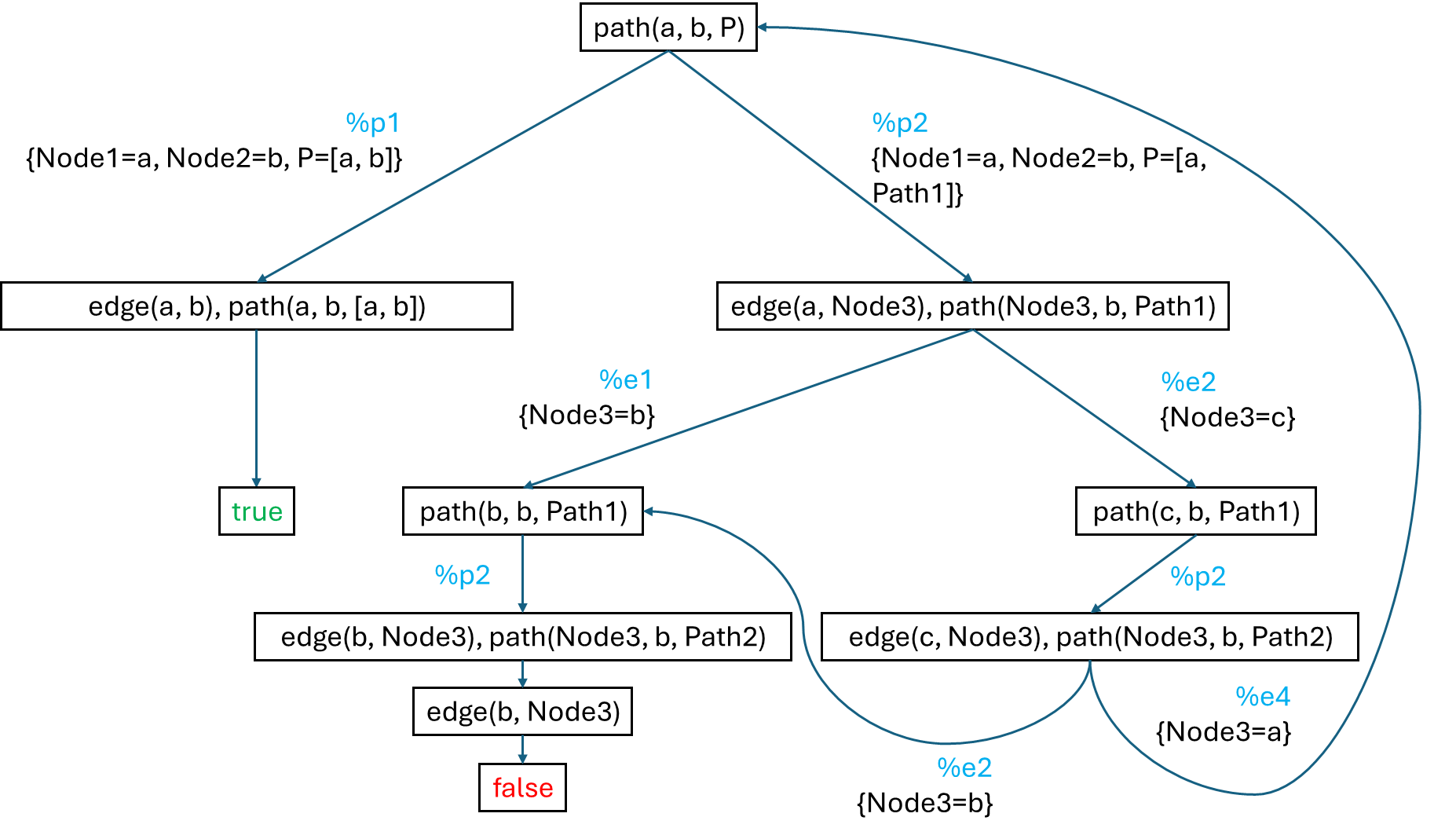
:- edge(Node1, Node2), Path = [Node1, Node2]. %p1

path(Node1, Node2, Path)

:- edge(Node1, Node3), path(Node3, Node2, Path1),

Path = [Node1 | Path1].  %p2

path(a, b, P)



It is an infinite tree and also a success tree.