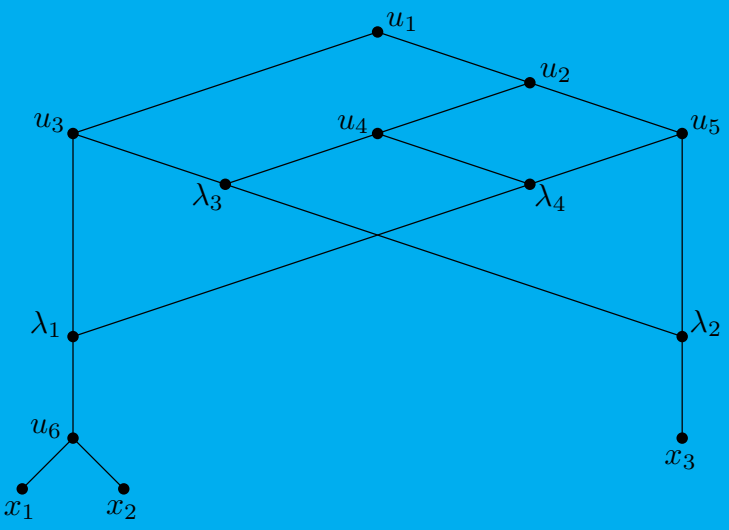
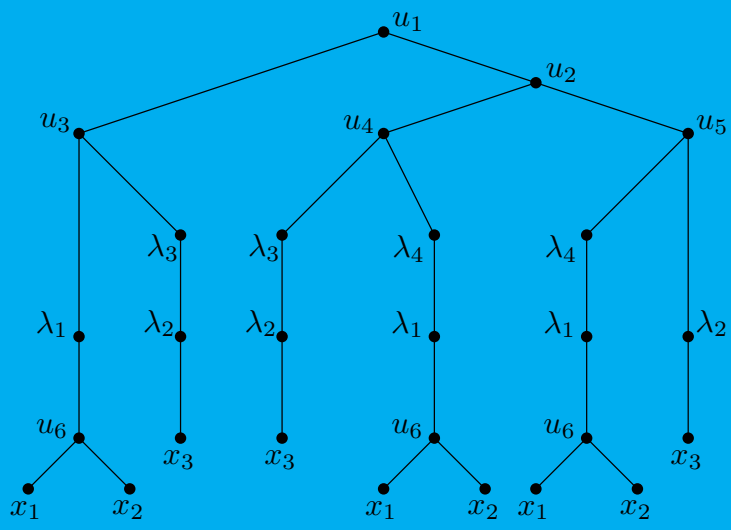
A new polynomial for describing Phylogenetic Networks

M. van de Klok

Bachelor End Project 2022



N_1



$U(N_1)$

TU Delft

A new polynomial for describing Phylogenetic Networks

by

M. van de Klok

to obtain the degree of Bachelor of Science
at the Delft University of Technology,
to be defended publicly on Wednesday July 13, 2022 at 10:00 AM.

Student number:	5034124	
Project duration:	April 19, 2022 – July 13, 2022	
Thesis committee:	Dr. ir. L. J. J. van Iersel,	TU Delft, supervisor
	Dr. K. Marynets,	TU Delft

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

Preface

I want to thank everyone who taught me more about Mathematics, finishing my Bachelor would not be possible without them.
I would like to specially thank my supervisor Leo van Iersel, for his guidance during my Bachelor End Project. All the meetings we had gave me much inspiration and motivation to write what lies in front of you, my Bachelor Thesis. I also would like to thank Kateryna Marynets for being part of my Bachelor Committee. Finally, I would like to thank my friends, family and everyone who supported me during my Bachelor.

M. van de Klok
Delft, July 2022

Abstract

Describing phylogenetic trees or networks with a polynomial is a tool to distinguish between them. In this thesis, a new polynomial for describing rooted binary internally labeled phylogenetic networks and trees is introduced based on the research of P. Liu [1] and J. Pons et al.[2]. Two different cases are considered, one where the reticulation nodes have distinct labels λ_i and one where the reticulation nodes have the same label λ. There are a few conjectures stated about the uniqueness of the polynomial and the relation of the polynomial with the primary subtrees and their monomial. Also the folding and unfolding of a network is described. Furthermore, an algorithm is provided with which a tree can be made out of different monomials. With use of the lemma that states when a tree can be folded to a network, it can be determined if the tree can be folded to a network.

Contents

1

Introduction

A phylogenetic network is a graph which shows the evolutionary relationships between for example different species. It is possible for an animal or plant to breed with an individual of another species, resulting in a new species, we call this process hybridisation. Hybridisation leads to a node in the network with in-degree two and out-degree one, which is called a reticulation node. When no hybridisation or other processes where lineages combine take place, we speak of a phylogenetic tree.
In this thesis, a polynomial for describing a phylogenetic network is introduced. With this, one can compare two networks. Also the distance between networks could be measured. The research done in this thesis is based on the research that P. Liu [1] and J. Pons et al. [2] have done before. In his research P. Liu has introduced a distinguishing polynomial that is a complete isomorphism invariant for trees, which is known as the Liu polynomial. J. Pons et al. generalized the Liu polynomial for trees to define a polynomial invariant for rooted phylogenetic networks. Also, a new class of phylogenetic networks, seperable networks, are introduced and their extension of the Liu polynomial characterizes this class completely.
This thesis is structured as follows. In Chapter 2, the mathematical notation that is throughout the thesis is introduced, among other things, a rooted binary internally labeled phylogenetic network is defined as a rooted directed acyclic graph with no parallel arcs and with two labeling functions, a bijective one for the leaves and a surjective one for the reticulation nodes. Then in Chapter 3, a new polynomial for describing rooted binary internally labeled phylogenetic networks will be defined. The process of unfolding a rooted binary internally labeled phylogenetic network to a rooted binary internally labeled phylogenetic tree, and its reverse, folding a tree to recover the network will be stated in Chapter 4 in two different algorithms. After that, primary subtrees, which are subtrees S of a tree T with the same root as T and at each node either all children or none are part of the subtree, will be introduced and discussed in Chapter 5, also the polynomial of a primary subtree which consists of only one term will be defined as a monomial. Finally, an algorithm to find a tree from a set of monomials will be described in Chapter 6. Finally, a conclusion will be drawn and recommendations for further research will be stated in the discussion.

2

Definitions

The mathematical notation that is going to be used, will be introduced in this section.
Let $X = \{x_1, x_2, \ldots, x_n\}$ denote a non-empty finite set, for which each x_j, $j \in \{1, 2, \ldots, n\}$ is an irreducable polynomial in $\mathbb{Z}[x_1, x_2, \ldots, x_n]$. The labels of the leaves, in the networks used, are these x_j, $j \in \{1, 2, \ldots, n\}$. To start, a phylogenetic network will be defined.

Definition 1. A *rooted binary phylogenetic network* $N = (V, E)$ *on* X, or simply a *phylogenetic network* on X, is a rooted directed acyclic graph with no parallel arcs satisfying the following conditions:

(i) any node with out-degree zero (a *leaf*) has in-degree one, and the set of nodes with out-degree zero, denoted by $L(N)$, is identified with X via a bijection $\varphi : L(N) \to X$;

(ii) the root is the only node with in-degree zero, and has out-degree two;

(iii) any other node has either in-degree one and out-degree two (a *tree* node), or in-degree two and out-degree one (a *reticulation* node).

Now that it is clear what a phylogenetic network is, it will be defined when a phylogenetic network is internally labeled.

Definition 2. A *rooted binary internally labeled phylogenetic network* $N = (V, E)$ *on* X, is a rooted binary phylogenetic network with a surjective labeling function l on the set of reticulation nodes $R(N)$:

$$l : R(N) \to \{\lambda_1, \lambda_2, \ldots, \lambda_r\}$$

These λ_i, $i \in \{1, 2, \ldots, r\}$ are irreducible polynomials in $\mathbb{Z}[x_1, x_2, \ldots, x_n, \lambda_1, \lambda_2, \ldots, \lambda_r]$.

Not only will networks be discussed, also a look will be taken on trees, which brings us to the next definition.

Definition 3. A *rooted binary internally labeled phylogenetic tree* $T = (V, E)$ *on* X, is a rooted directed acyclic graph with no parallel arcs satisfying the following conditions:

(i) any node with out-degree zero (a *leaf*) has in-degree one, and the set of nodes with out-degree zero, denoted by $L(N)$, is identified with X via surjection $\varphi : L(N) \to X$;

(ii) the root is the only node with in-degree zero, and it can have out degree one (an *elementary* node), or two (a *tree* node);

(iii) any other node has either in-degree one and out-degree two (a *tree* node) or in-degree one and out-degree one (also an *elementary* node);

(iv) if $E(N)$ denotes the set of elementary nodes of N, then there is a labeling function $l : E(N) \to \{\lambda_1, \lambda_2, \ldots, \lambda_r\}$.

Note that in a tree, leaves and elementary nodes with the same label can occur multiple times unlike in networks. Next, we will define when two networks are isomorphic. In order to do this, we use the notation (V, E, φ) for a phylogenetic network and (V, E, φ, l) for an internally labeled phylogenetic network.

Definition 4. Two phylogenetic networks $N_1 = (V_1, E_1, \varphi_1)$ and $N_2 = (V_2, E_2, \varphi_2)$ on X are isomorphic ($N_1 \simeq N_2$) if there exists a bijection $f : V_1 \to V_2$ such that $\varphi_1(x) = \varphi_2(f(x))$ for all $x \in L(N_1)$, and $(u, v) \in E_1$ if and only if $(f(u), f(v)) \in E_2$.

Definition 5. Two internally labeled phylogenetic networks $N_1 = (V_1, E_1, \varphi_1, l_1)$ and $N_2 = (V_2, E_2, \varphi_2, l_2)$ on X are isomorphic ($N_1 \simeq N_2$) if there exists a bijection $f : V_1 \to V_2$ such that $\varphi_1(x) = \varphi_2(f(x))$ for all $x \in L(N_1)$, $l_1(x) = l_2(f(x))$ for all $x \in R(N_1)$ and $(u, v) \in E_1$ if and only if $(f(u), f(v)) \in E_2$.

3

Polynomial

In this chapter, a polynomial for describing a phylogenetic network will be introduced.

3.1. Reticulation nodes with distinct labels

Let N be an internally labeled phylogenetic network with bijective labelling function l of the reticulations. Then, consider

$$p : V(N) \to \mathbb{Z}[x_1, x_2, \ldots, x_n, \lambda_1, \lambda_2, \ldots, \lambda_r, y]$$

to be defined recursively as follows. Let $u \in V(N)$, then:

- if u is a leaf, $p(u) = \varphi(u)$;
- if u is an internal tree node whose children are v_1, v_2, then $p(u) = y + p(v_1)p(v_2)$;
- otherwise, i.e. if u has only one child v, then $p(u) = l(u) + p(v)$

Finally, the polynomial of the network $p(N)$ is equal to $p(\rho)$, where ρ is the root of N.
We refer to the polynomial as defined above as the new polynomial. The polynomial which is devised by Pons et al. is considered as the Pons polynomial. The Pons polynomial is similar to the new polynomial, the difference with the new polynomial is found in the last part of the definition. Where in the new polynomial, the polynomial of u with only one child v is defined as $p(u) = l(u) + p(v)$, Pons has defined this as $p(u) = l(u)p(v)$. Now we determine the new polynomial for two networks from Figure 3.1 which are inspired by the

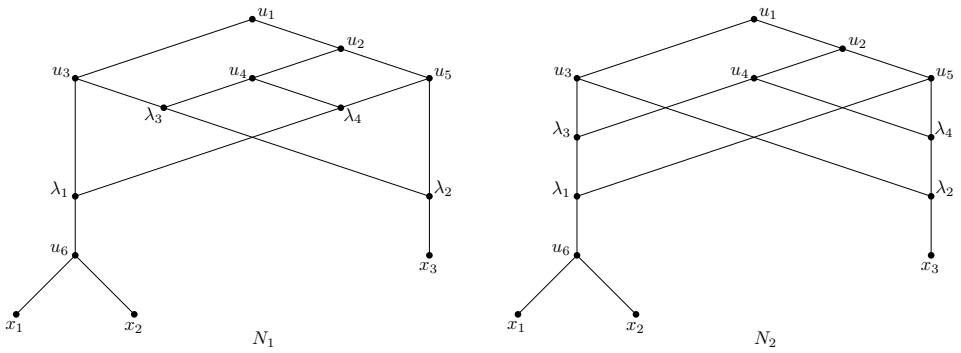


Figure 3.1: Two networks

networks from [2] for which the Pons polynomial is equal (Appendix B).
For network N_1, the polynomial at all nodes will be described. The polynomial for network N_2 is obtained the same way. To start, the polynomial at the leaves is going to be determined, they have label x_j, $j = 1,2,3,4$, so

that is their polynomial. Then,

$p(u_6) = x_1x_2 + y$,
$p(\lambda_1) = x_1x_2 + y + \lambda_1$, $p(\lambda_2) = \lambda_2 + x_3$,
$p(\lambda_3) = \lambda_2 + \lambda_3 + x_3$, $p(\lambda_4) = x_1x_2 + y + \lambda_1 + \lambda_4$,
$p(u_3) = y + (x_1x_2 + y + \lambda_1)(\lambda_2 + \lambda_3 + x_3)$, $p(u_4) = y + (x_1x_2 + y + \lambda_1 + \lambda_4)(\lambda_2 + \lambda_3 + x_3)$,
$p(u_5) = y + (x_1x_2 + y + \lambda_1 + \lambda_4)(\lambda_2 + x_3)$, $p(u_2) = (y + (x_1x_2 + y + \lambda_1 + \lambda_4)(\lambda_2 + \lambda_3 + x_3))(y + (x_1x_2 + y + \lambda_1 + \lambda_4)(\lambda_2 + x_3)) + y$

and finally,
$p(N_1) = p(u_1) = y + (y + (x_1x_2 + y + \lambda_1)(\lambda_2 + \lambda_3 + x_3))((y + (x_1x_2 + y + \lambda_1 + \lambda_4)(\lambda_2 + \lambda_3 + x_3))(y + (x_1x_2 + y + \lambda_1 + \lambda_4)(\lambda_2 + x_3)) + y)$.

For network N_2, the polynomial
$p(N_2) = p(u_1) = y + (y + (x_1x_2 + y + \lambda_1 + \lambda_3)(\lambda_2 + x_3))((y + (x_1x_2 + y + \lambda_1 + \lambda_3)(x_3 + \lambda_2 + \lambda_4))(y + (x_1x_2 + y + \lambda_1)(x_3 + \lambda_2 + \lambda_4)) + y)$ is obtained.
When the obtained polynomials are expanded (Appendix A) and subtracted from eachother, the result (Appendix A) is unequal to zero so the polynomials are different.

3.2. Reticulation nodes with the same label

For phylogenetic networks, it is also interesting to look at the case when the labels of the reticulation nodes are all the same.
When we consider an internally labeled phylogenetic network N with non-unique labels of the reticulation nodes, so all the reticulation nodes have label λ. Then, consider

$$p : V(N) \to \mathbb{Z}[x_1, x_2, \ldots, x_n, \lambda, y]$$

to be defined recursively as follows. Let $u \in V(N)$, then:

- if u is a leaf, $p(u) = \varphi(u)$;
- if u is an internal tree node whose children are v_1, v_2, $p(u) = y + p(v_1)p(v_2)$;
- otherwise, i.e. if u has only one child v, then $p(u) = \lambda + p(v)$

When looking at the networks from Figure 3.1 and replacing the λ_i with λ, the polynomials
$p(N_1) = p(u_1) = y + (y + (x_1x_2 + \lambda + y)(2\lambda + x_3))((y + (x_1x_2 + 2\lambda + y)(2\lambda + x_3))(y + (x_1x_2 + 2\lambda + y)(\lambda + x_3)) + y)$ and
$p(N_2) = p(u_1) = y + (y + (x_1x_2 + 2\lambda + y)(\lambda + x_3))((y + (x_1x_2 + 2\lambda + y)(2\lambda + x_3))(y + (x_1x_2 + \lambda + y)(2\lambda + x_3)) + y))$ are obtained for networks N_1 and N_2.
When these polynomials are expanded (Appendix A) and subtracted, the result equals $\lambda x_1x_2y - \lambda x_3y + \lambda y^2$, which is obviously not equal to zero, so the polynomials are still different. In Chapter 5, it will be discussed how this difference can be interpreted.

4

Folding and Unfolding

In this chapter, the unfolding and folding algorithms will be introduced. Furthermore, the uniqueness of folding and unfolding will be discussed.

4.1. Unfolding a phylogenetic network

To start, the set $R_{min}(N)$ will be introduced. This is the set of reticulation nodes of a rooted binary internally labeled phylogenetic network N, such that none of its descendants are reticulation nodes theirselves. In each repetition of the algorithm, the set $R_{min}(N)$ will be updated.

4.1.1. Unfolding algorithm

Let N be a rooted binary internally labeled phylogenetic network, and let $R_{min}(N)$ be as described above. Then:

1. take $u \in R_{min}(N)$, let v_1, v_2 be its parents;
2. delete the edge (v_2, u);
3. duplicate the rooted binary internally labeled phylogenetic tree that is rooted at u;
4. add an edge from v_2 to the copied u, i.e. the u which has no incoming edge;
5. update the set $R_{min}(N)$;
   (a) if $R_{min}(N) \neq \emptyset$, go to the first step.
   (b) if $R_{min}(N) = \emptyset$, the network is completely unfolded, the result is a rooted binary internally labeled phylogenetic tree.

We define $U(N, u)$ as the result of the unfolding at node u of network N and $U(N)$ as the completely unfolded network. When there is more than one node in $R_{min}(N)$ say u_1 and u_2, the order of unfolding does not matter, i.e. $U(U(N, u_1), u_2) = U(U(N, u_2), u_1)$. [2]

4.1.2. Examples of unfolding

In this part, the network from Figure 4.1(a) will be unfolded using the algorithm above, resulting in the tree from Figure 4.1(c).
Starting with the algorithm, it can be seen that $R_{min} = \{\lambda_2\}$, thus we take λ_2. Now, the edge (u_5, λ_2) will be deleted (marked purple in Figure 4.1(a)). Then, the tree rooted at λ_2 is copied, this copy is marked green in Figure 4.1(b). Next, an edge between the copy of λ_2 and u_5 is added (marked orange in Figure 4.1(b)). An update on R_{min} is performed, resulting $R_{min} = \{\lambda_1\}$. This is not empty so another repetion of the algorithm needs to be done. First, we delete the edge (u_3, λ_1) (marked purple in Figure 4.1(b)). Then, the tree rooted at λ_1 is copied (marked green in Figure 4.1(c)). Finally, the edge from u_3 to the copy of λ_1 is added, marked orange in Figure 4.1(c). Updating R_{min} results in the empty set thus the network is completely unfolded.
The unfolded networks from Figure 3.1 are shown in Figure 4.2, these will be used in Chapter 5.

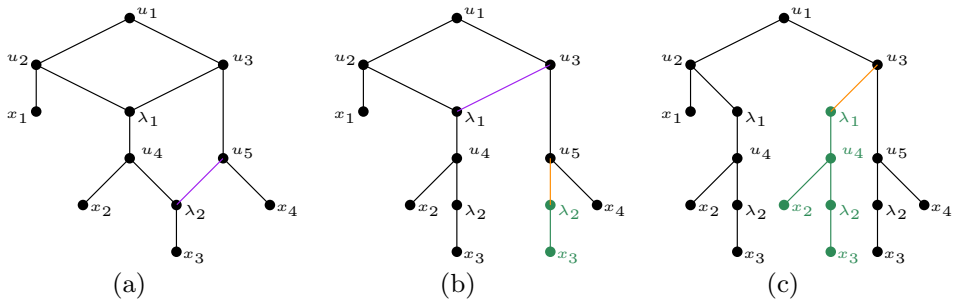


Figure 4.1: Unfolding a network to a tree

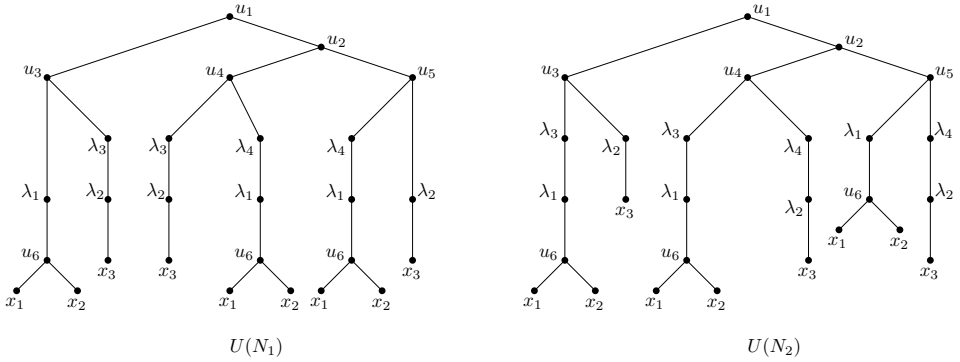


Figure 4.2: Unfoldings of N_1 and N_2

4.2. Folding a phylogenetic network

Let $E(M)$ be the set of elementary nodes of a (partially) unfolded network M. Now we define an order in this set. Let $u, v \in E(M)$, then $u \leq_E v$ if and only if there exist $u', v' \in E(M)$ with $l(u) = l(u')$ and $l(v) = l(v')$ and u is a descendant of v, i.e. there is a path from v to u. The set of elementary nodes that are maximal under this order is called $E_{max}(M)$. [2] We define the subnetwork with the node u as the root as $M(u)$

4.2.1. Folding algorithm

Let M be a (partially) unfolded rooted binary internally labeled phylogenetic network, and let $E_{max}(M)$ be as described above. Then:

1. consider the nodes $u, v \in E_{max}(M)$ such that $l(u) = l(v)$ and $M(u) = M(v)$ and let w be the parent of v;
2. delete both $M(v)$ and the edge (w, v);
3. add an edge from w to u, a reticulation node is formed;
4. update the set $E_{max}(M)$;
   (a) if $E_{max} \neq \emptyset$, go to the first step.
   (b) if $E_{max}(M) = \emptyset$, the (partially) unfolded network is completely folded, the result is a rooted binary internally labeled phylogenetic network.

Note that M can be a rooted binary internally labeled phylogenetic tree the first time you start at the first step, after that it is not a tree anymore, since there is at least one reticulation node. We define $F(M, u)$ as the result of the folding at node u of network M and $F(M)$ as the result of the algorithm with input M. Also, just as with unfolding, the order of folding does not matter.

4.2.2. Examples of folding

In this part, the tree from Figure 4.3(a) will be folded using the algorithm above, resulting in the network from Figure 4.3(c). Suppose that $l(u) = l(v) = \lambda_1$ and $l(u') = l(v') = \lambda_2$.
Starting with the algorithm, $E_{max} = \{u, v\}$ and u_3 is the parent of v. Now, the subnetwork $M(v)$ and the edge

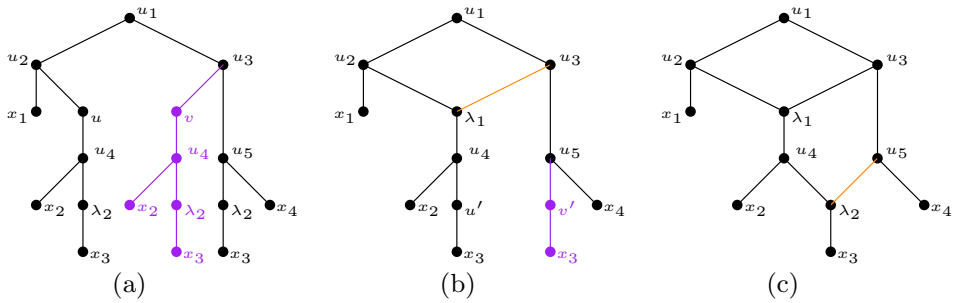


Figure 4.3: Folding a tree to a network

(u_3, v) (purple in Figure 4.3(a)) will be deleted. Then, the edge (u_3, u) (orange in Figure 4.3(b)) is added and the reticulation node will be named the same as its label. Next, the set E_{max} is updated and becomes$\{u', v'\}$. Starting at step one again, the node u_5 is the parent of v'. Then the subnetwork $M(v')$ and the edge (u_5, v') (purple in Figure 4.3(b)) are deleted. Finally, the edge (u_5, u') (orange in Figure 4.3(c)) is added and we name the new reticulation node equal to its label. Now, if the set E_{max} is updated, it is equal to the empty set, thus the tree is completely folded.

4.3. Uniqueness of folding and unfolding

As seen in the paper from Pons et al. ([2]), an internally labeled phylogenetic network can be unfolded such that the obtained internally labeled phylogenetic tree is unique for the network. So that the tree can be folded back to the network. This is stated in the following theorem which can be found in that paper.

Theorem 1. *[2] Let N be a rooted binary internally labeled phylogenetic network. Then*

$$N = F(U(N))$$

Note that the elementary nodes of the unfolded network correspond with the reticulation nodes of the original network. Since the subnetwork under each reticulation node of a network is considered twice when defining its polynomial, this results in the following.

Observation 1. The polynomial of a rooted binary internally labeled phylogenetic network N is equal to the polynomial of its unfolding $U(N)$, i.e. $p(N) = p(U(N))$.

Something similar has been found for the Pons polynomial ([2]).
Now, it is interesting to see when a rooted binary internally labeled phylogenetic tree can be folded to an rooted binary internally labeled phylogenetic network. This is stated in the next lemma.

Lemma 1. If a rooted binary internally labeled phylogenetic tree with distinct labels can be folded to a rooted binary internally labeled phylogenetic network, then for each label λ_i, $i = 1, 2, \ldots, r$, the number of elementary nodes with label λ_i is equal to 2+ the number of ancestor elementary nodes with distinct label λ_j , with $i \neq j$.

Proof. When there is only one λ, say λ_1, then a folding operation can only be done if there are 2 elementary nodes u, v with label λ_1 and $M(u) = M(v)$. Otherwise, the folding cannot be done, i.e. the number of elementary nodes with label λ_1 must equal 2 to be able to fold.
Now suppose that there are multiple different λ_i. If the nodes with these labels are not ancestors of eachother, then all these λ_i are in E_{max} and the order of the folding does not matter. It can be seen as repeating the folding of one label λ_i. In this case, again, all the labels λ_i must be present twice to be able to fold.
Now, let multiple different λ_i be ancestors of eachother. Then the label of the elementary node closest to the root, must be present twice to be able to fold. Then when the folding operation is done, a copy of everything below (including other elementary nodes) is deleted. So with every folding step, a copy of every elementary node with the same label is deleted. This means that to be able to fold all the elementary nodes, the label needs to be present as many times as a folding operation deletes a copy plus two, which is equal to the amount of different labels as ancestor plus two. □

It is useful to know when a tree can be folded back to a network. For example, when looking at Figure 4.4 from [2], these three trees have the same Pons polynomial, but they cannot be all folded to a network by Lemma 1, so they cannot be unfoldings from networks with the same Pons polynomial.

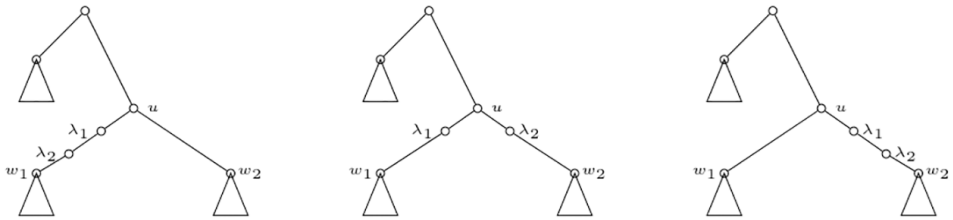


Figure 4.4: Three trees with the same Pons polynomial

5

Primary subtrees and distance between networks

In this chapter, primary subtrees, their relation with the polynomial of a network and the difference and distance between two networks will be discussed. In this chapter, only the networks and trees with one label for all the reticulation nodes and elementary nodes will be considered.

5.1. Primary subtrees

When looking at an internally labelled phylogenetic tree which is an unfolded internally labelled phylogenetic network, we can describe primary subtrees in the following way.

Definition 6. A primary subtree S of a tree T is a rooted subtree of T, such that S shares the same root with T and any leaf node of T is either a leaf node of S or a descendant of a leaf node of S. Let S_T be the set of all the primary subtrees of tree T.[1]

Now that primary subtrees are defined, we can define the monomial for a primary subtree.

Definition 7. The monomial for a primary subtree S of a tree T, is a polynomial of the form $q(S) = \lambda^{\alpha} y^{\gamma} \prod_{j=1}^{n} x_j^{\beta_j}$, where:

- $\alpha + \gamma + \sum_{j=1}^{n} \beta_j$ equals the number of leaves of the primary subtree S;
- α equals the number of leaves of the primary subtree S that are elementary nodes in the tree T;
- each β_j equals the number of leaves of the primary subtree S that are leaves labeled x_j in the tree T, similarly, $\sum_{j=1}^{n} \beta_j$ equals the number of leaves of the primary subtree S that are also leaves in the tree T;
- γ are the number of leaves of the primary subtree S that are tree nodes in the tree T.

Note that the α, β_j and γ could also be equal to zero.
Also note that the monomials can be determined for networks with distinct λ. Then λ_{α} becomes $\lambda_i^{\alpha_i}$, with $i = 1, 2, \ldots, r$ and monomial becomes $q(S) = y^{\gamma}(\prod_{i=1}^{r} \lambda_i^{\alpha_i})(\prod_{j=1}^{n} x_j^{\beta_j})$. The total number of leaves of the primary subtree S equals $\gamma + (\sum_{i=1}^{r} \alpha_i) + (\sum_{j=1}^{n} \beta_j)$ and the interpretation of the α_i is similar to the interpretation of the β_j. In Figure 5.1 a tree T and two of its primary subtrees S_1, S_2 are shown. The primary subtree in Figure 5.1(b) has monomial $q(S_1) = \lambda^2 x_1 x_4 y$, the primary subtree in Figure 5.1(c) has monomial $q(S_2) = \lambda x_1 x_2^2 x_3 y$.

5.2. Relation between primary subtrees and the polynomial of a network

In the paper of Liu [1], a polynomial for unlabeled rooted trees is defined. In this section, a few conjectures are stated for the new polynomial which are similar to what Liu proved for unlabeled rooted trees.

Conjecture 1. $p(T) = \sum_{S \in S_T} q(S)$. i.e. the sum of all the monomials of primary subtrees S of a rooted binary internally labeled phylogenetic tree T is equal to the polynomial of the tree.

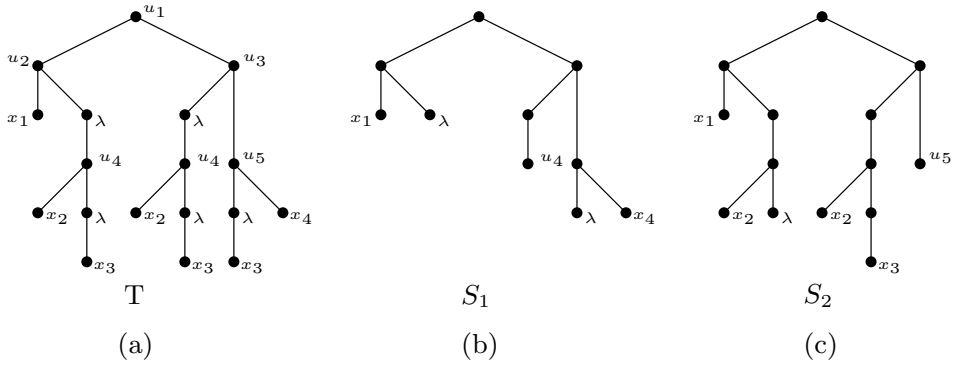


Figure 5.1: A tree and two of its primary subtrees

From Conjecture 1, the following would follow directly;

Conjecture 2. Let T be a rooted binary internally labeled phylogenetic tree. Suppose $p(T)$ has m terms and $a_1, a_2, \ldots, a_m$ are the corresponding coefficients, then T has $\sum_{i=1}^{m} a_i$ primary subtrees.

Conjecture 2 tells us that the coefficient before every term of the polynomial is equal to the number of primary subtrees with that polynomial. The following conjecture states that equal polynomials are from equal trees and the other way around.

Conjecture 3. Let T_1 and T_2 be two rooted binary internally labeled phylogenetic trees. $T_1 \simeq T_2$ if and only if $p(T_1) = p(T_2)$.

Since Observation 1 told us that the polynomial of a network is equal to the polynomial of its unfolding and since the unfolding of a network is a tree, from Conjecture 3 it follows that the polynomial of a network is unique. This is stated in the following conjecture.

Conjecture 4. Let N_1 and N_2 be two rooted binary internally labeled phylogenetic networks. $N_1 \simeq N_2$ if and only if $p(N_1) = p(N_2)$.

5.2.1. Difference between two networks

From Conjecture 4 it can be seen that the difference between two networks can be seen in the difference of their primary subtrees. Now, we take a look at the difference between N_1 and N_2 from Figure 3.1 and we use the unfoldings of these networks from Figure 4.2, but then with one label λ. The difference of these polynomials was $\lambda x_1 x_2 y - \lambda x_3 y + \lambda y^2$, in Appendix A the corresponding monomials are bold. In Figure 5.2, primary subtrees with monomial $\lambda x_1 x_2 y$ are shown. In Figure 5.3 and Figure 5.4, the primary subtrees with monomial $\lambda x_3 y$ and λy^2 are shown. Indeed, it is seen that the primary subtrees of the unfolded networks correspond with the terms of the polynomial and that the coefficient is equal to the number of primary subtrees with the same monomial.

5.2.2. Distance between two networks

Now that we have seen the difference between two networks, it is useful to assign a number to this, to create an order in the differences. We introduce two different ways to measure the distance between two networks. For both of the distances, the first step is to subtract the two different polynomials from the networks from each other resulting a polynomial of the form $p = \sum_{i=1}^{m} a_i q(S)$.
Now we define the first distance $D_1(N, M)$ as the sum of the absolute values of the coefficients of $p(N) - p(M)$, i.e. $D_1(N, M) = \sum_{i=1}^{m} |a_i|$.
We define the second distance $D_2(N, M)$ as the square root of the sum of the coefficients of $p(N) - p(M)$ squared, i.e. $D_2(N, M) = \sqrt{\sum_{i=1}^{m} a_i^2}$.
Note that it does not matter which polynomial is subtracted from the other since the coefficients will be squared or the absolute value will be taken so the sign of the coefficient does not matter.
For the networks N_1 and N_2 from Figure 3.1, the obtained differences are $D_1(N_1, N_2) = 1 + 1 + 1 = 3$ and $D_2(N_1, N_2) = \sqrt{(1)^2 + (-1)^2 + (1)^2} = \sqrt{3}$.

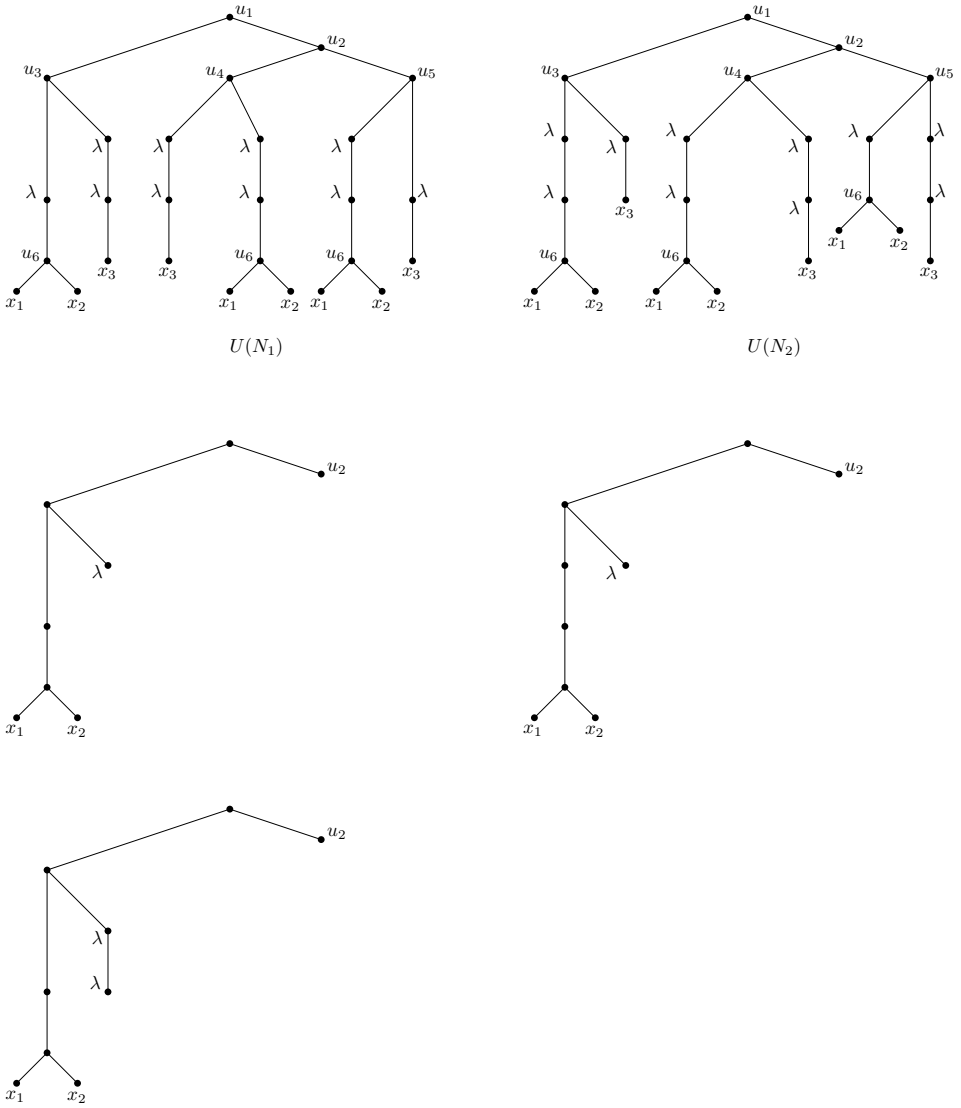


Figure 5.2: Primary subtrees of N_1 and N_2 with monomial $\lambda x_1 x_2 y$

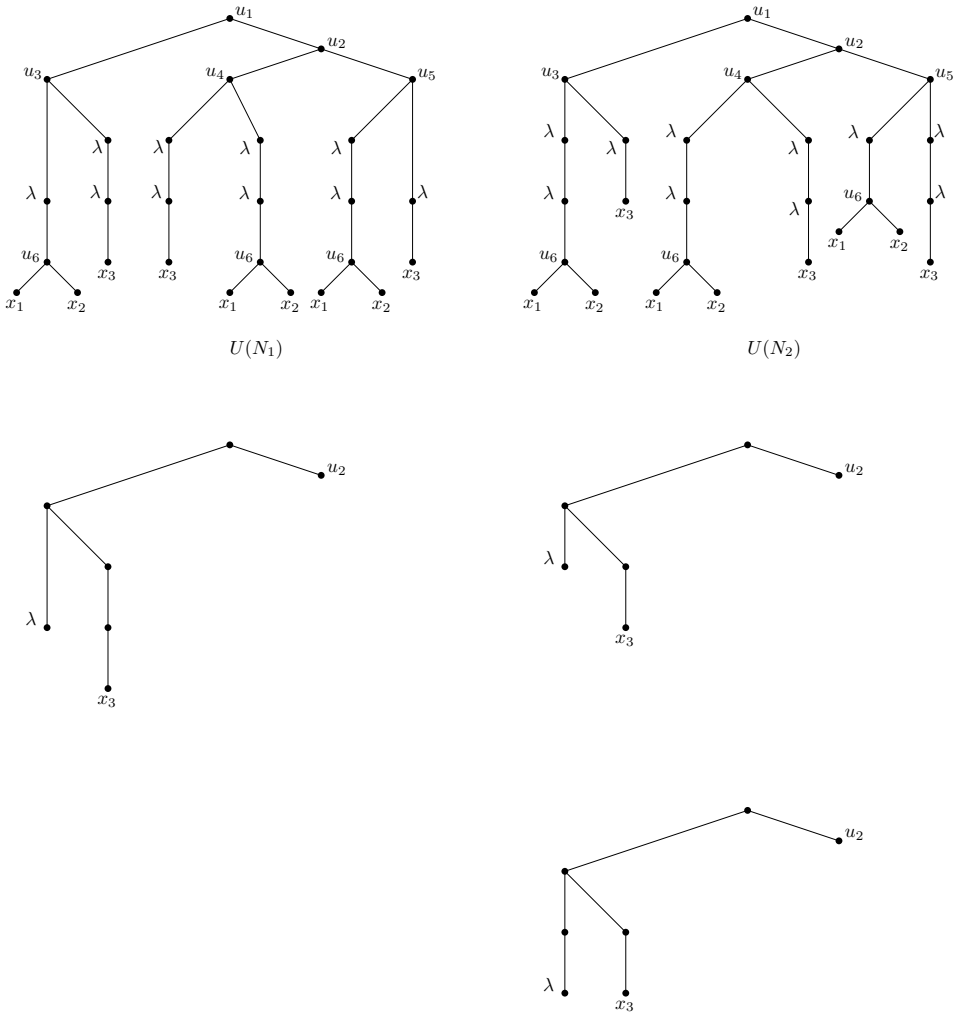


Figure 5.3: Primary subtrees of N_1 and N_2 with monomial $\lambda x_3 y$

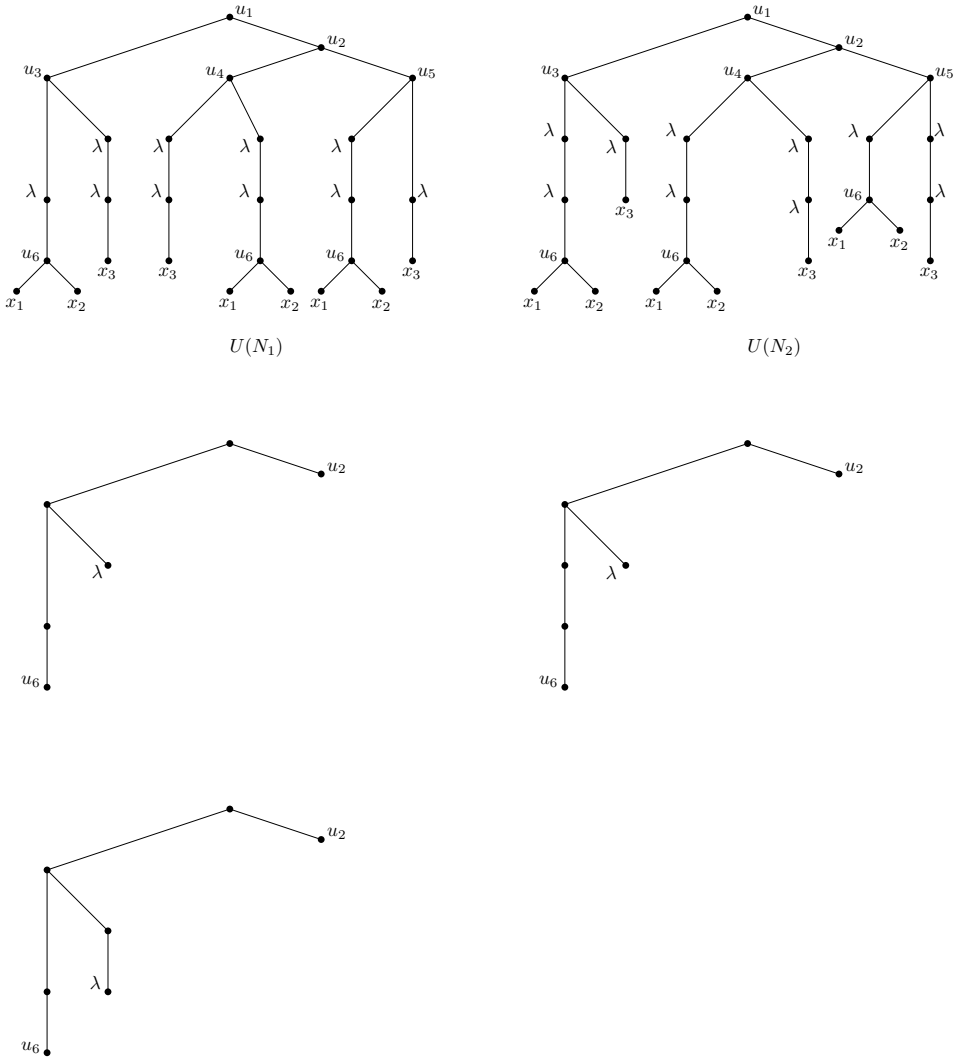


Figure 5.4: Primary subtrees of N_1 and N_2 with monomial λy^2

6

Algorithm

In this chapter, an algorithm to make an unfolded network from the sum of monomials will be introduced.

Assume that there are several different trees found and that those trees share the same root node. Then, the monomial of all these trees can be found. With the algorithm described in this chapter, it is possible to rewrite the polynomial such that the unfolded network it describes can be found. After using the algorithm, the list needs to be rewritten to a polynomial. We start with replacing all the square brackets with round brackets and deleting all the commas, when there are two opening brackets next to each other, $y+$ is written in between.

Data: list with only entry the sum of monomials, i.e. polynomial
Result: list which can be rewritten to a factorized polynomial
while *polynomial* $\neq 0$ **do**
  Factorise polynomial minus y;
  if *result is a factor ab* **then**
    delete the polynomial and insert [a],[b] at the place of the deleted polynomial;
    [a] becomes the polynomial;
  else
    if *current entry is not the last* **then**
      next entry is polynomial;
    else
      polynomial=0.
    end
  end
end

Algorithm 1: From sum of monomials to list

In Appendix B, this algorithm is used on the expanded polynomial of N_2 from Figure 3.1. The resulting lists are as follows;

Start: $[4\lambda^3x_1^3x_2^3+8\lambda^2x_1^3x_2^3x_3+5\lambda x_1^3x_2^3x_3^2+x_1^3x_2^3x_3^3+20\lambda^4x_1^2x_2^2+40\lambda^3x_1^2x_2^2x_3+12\lambda^3x_1^2x_2^2y+25\lambda^2x_1^2x_2^2x_3^2+24\lambda^2x_1^2x_2^2x_3y+5\lambda x_1^2x_2^2x_3^3+15\lambda x_1^2x_2^2x_3^2y+3x_1^2x_2^2x_3^3y+32\lambda^5x_1x_2+64\lambda^4x_1x_2x_3+40\lambda^4x_1x_2y+40\lambda^3x_1x_2x_3^2+80\lambda^3x_1x_2x_3y+12\lambda^3x_1x_2y^2+8\lambda^2x_1^2x_2^2y+8\lambda^2x_1x_2x_3^3+50\lambda^2x_1x_2x_3^2y+24\lambda^2x_1x_2x_3y^2+10\lambda x_1^2x_2^2x_3y+10\lambda x_1x_2x_3^3y+15\lambda x_1x_2x_3^2y^2+3x_1^2x_2^2x_3^2y+3x_1x_2x_3^3y^2+16\lambda^6+32\lambda^5x_3+32\lambda^5y+20\lambda^4x_3^2+64\lambda^4x_3y+20\lambda^4y^2+26\lambda^3x_1x_2y+4\lambda^3x_3^3+40\lambda^3x_3^2y+40\lambda^3x_3y^2+4\lambda^3y^3+33\lambda^2x_1x_2x_3y+16\lambda^2x_1x_2y^2+8\lambda^2x_3^3y+25\lambda^2x_3^2y^2+8\lambda^2x_3y^3+10\lambda x_1x_2x_3^2y+20\lambda x_1x_2x_3y^2+5\lambda x_3^3y^2+5\lambda x_3^2y^3+6x_1x_2x_3^2y^2+x_3^3y^3+20\lambda^4y+26\lambda^3x_3y+26\lambda^3y^2+8\lambda^2x_3^2y+33\lambda^2x_3y^2+8\lambda^2y^3+5\lambda x_1x_2y^2+10\lambda x_3^2y^2+10\lambda x_3y^3+3x_1x_2x_3y^2+3x_3^2y^3+8\lambda^2y^2+\lambda x_1x_2y+5\lambda x_3y^2+5\lambda y^3+x_1x_2x_3y+3x_3y^3+2\lambda^2y+2\lambda x_3y+\lambda y^2+x_3y^2+y^3+y^2+y]$

Iteration 1: $[[\lambda x_1x_2+x_1x_2x_3+2\lambda^2+2\lambda x_3+\lambda y+x_3y+y],[4\lambda^2x_1^2x_2^2+4\lambda x_1^2x_2^2x_3+x_1^2x_2^2x_3^2+12\lambda^3x_1x_2+12\lambda^2x_1x_2x_3+8\lambda^2x_1x_2y+3\lambda x_1x_2x_3^2+8\lambda x_1x_2x_3y+2x_1x_2x_3^2y+8\lambda^4+8\lambda^3x_3+12\lambda^3y+2\lambda^2x_3^2+12\lambda^2x_3y+4\lambda^2y^2+4\lambda x_1x_2y+3\lambda x_3^2y+4\lambda x_3y^2+2x_1x_2x_3y+x_3^2y^2+6\lambda^2y+3\lambda x_3y+4\lambda y^2+2x_3y^2+y^2+y]]$

Iteration 2: $[[[\lambda+x_3],[x_1x_2+2\lambda+y]],[4\lambda^2x_1^2x_2^2+4\lambda x_1^2x_2^2x_3+x_1^2x_2^2x_3^2+12\lambda^3x_1x_2+12\lambda^2x_1x_2x_3+8\lambda^2x_1x_2y+3\lambda x_1x_2x_3^2+8\lambda x_1x_2x_3y+2x_1x_2x_3^2y+8\lambda^4+8\lambda^3x_3+12\lambda^3y+2\lambda^2x_3^2+12\lambda^2x_3y+4\lambda^2y^2+4\lambda x_1x_2y+3\lambda x_3^2y+4\lambda x_3y^2+2x_1x_2x_3y+x_3^2y^2+6\lambda^2y+3\lambda x_3y+4\lambda y^2+2x_3y^2+y^2+y]]$

Iteration 3:$[[[\lambda+x_3],[x_1x_2+2\lambda+y]],[[2\lambda x_1x_2+x_1x_2x_3+4\lambda^2+2\lambda x_3+2\lambda y+x_3y+y],[2\lambda x_1x_2+x_1x_2x_3+2\lambda^2+\lambda x_3+2\lambda y+x_3y+y]]]$
Iteration 4: $[[[\lambda+x_3],[x_1x_2+2\lambda+y]],[[[2\lambda+x_3],[x_1x_2+2\lambda+y]],[2\lambda x_1x_2+x_1x_2x_3+2\lambda^2+\lambda x_3+2\lambda y+x_3y+y]]]$
Iteration 5:$[[[\lambda+x_3],[x_1x_2+2\lambda+y]],[[[2\lambda+x_3],[x_1x_2+2\lambda+y]],[[x_1x_2+\lambda+y],[2\lambda+x_3]]]]$
When rewriting this last step, we obtain; $(y+(y+(\lambda+x_3)(x_1x_2+2\lambda+y))(y+(y+(2\lambda+x_3)(x_1x_2+2\lambda+y))(y+(x_1x_2+\lambda+y)(2\lambda+x_3))))$.
With this polynomial it is possible to find $U(N_2)$ from Figure 4.2 with each λ_i replaced by λ. For every y, a tree node is made, if the polynomial minus y results in a factor, with each of the factors as the polynomial of a child. For every λ an elementary node is made and for each x_i a leaf is made. Note that the leaves are the last ones made and that a tree node is only made if the polynomial minus y results in a factor. In Figure 6.1 some of the steps for the polynomial above are worked out.

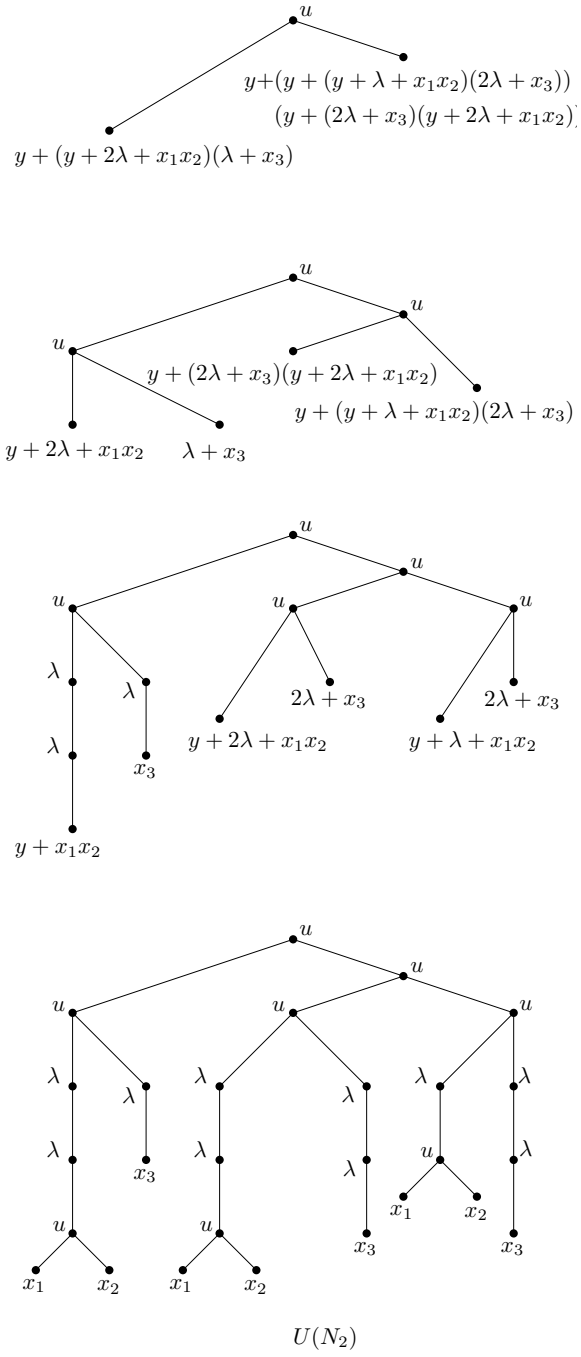


Figure 6.1: Finding $U(N_2)$ from the obtained polynomial

7

Conclusion

In this thesis, a new polynomial for describing a rooted binary internally labeled phylogenetic network was introduced for networks with reticulation nodes with distinct labels λ_i and for networks with reticulation nodes with the same label λ, which was inspired by the paper from Pons et al. [2].

Then, two different algorithms were defined for unfolding rooted binary phylogenetic networks and folding rooted binary phylogenetic trees. A look was taken at the uniqueness of folding and unfolding and it was found that the polynomial of a rooted binary internally labeled phylogenetic network is equal to the polynomial of its unfolding, which is actually a rooted binary internally labeled phylogenetic tree. Lemma 1 tells us when a rooted binary internally labeled phylogenetic tree can be folded to a rooted binary internally labeled network, which is useful to determine whether different trees could be folded to the same network.

Next, primary subtrees and monomials were defined and their relation with the polynomial of a network was observed. This leaded to four conjectures about the relation between monomials and the polynomial of a tree and the uniqueness of the polynomials of trees and networks. Then, the difference between two networks was observed by the different primary subtrees. Also, the distance between two networks was defined in two different ways.

Finally, an algorithm was made with which it is possible to find an unfolded network back from the sum of monomials. When a few primary subtrees with the same root node are known, it is possible to find out if these can form a rooted binary internally labeled tree. If they can form a tree, Lemma 1 provides a necessary condition for whether the tree can be folded to a rooted binary internally labeled phylogenetic network.

8

Discussion

In this thesis, a few conjectures were stated in Chapter 5 which are not proven yet. For further research it is useful to prove these conjectures, since for the difference and distance between networks and for the algorithm, they are assumed to be true. It is likely that they could be proven by adjusting the proofs in the paper from Liu [1].

Furthermore, it is interesting to see what happens when a look is taken at rooted binary networks (so networks without internal labels). In that case, reticulation nodes are allowed, but they are not labeled so only the leaves are labeled. In that case, it is suggested that the polynomial for a node u with only one child v is defined as $p(u) = y + p(v)$. Is the polynomial still unique or are there different networks with the same polynomial?

In this research, elementary nodes were not allowed in the rooted binary internally labeled networks. What would the effect of allowing them be on the polynomial? When they are labeled in the same way as the reticulation nodes, would the folding and unfolding operations still be possible and is it then still true that the folding of the unfolding of a network equals the network?

A

Polynomials

In this Appendix, the expanded polynomials of the networks N_1 and N_2 from Figure 3.1 are stated. The subscript of p shows if distinct λ_i or one λ is used.

$$
\begin{aligned}
p_{\lambda_i}(N_1) = {} & x_1^3x_2^3x_3^3 + 3x_1^3x_2^3x_3^2\,\lambda_2 + 2x_1^3x_2^3x_3^2\lambda_3 + 3x_1^3x_2^3x_3\lambda_2^2 + 4x_1^3x_2^3x_3\lambda_2\lambda_3 + x_1^3x_2^3x_3\lambda_3^2 + x_1^3x_2^3\lambda_2^3 + 2x_1^3x_2^3\lambda_2^2\lambda_3 \\
& + x_1^3x_2^3\lambda_2\lambda_3^2 + 3x_1^2x_2^2x_3^3y + 3x_1^2x_2^2x_3^3\lambda_1 + 2x_1^2x_2^2x_3^3\lambda_4 + 9x_1^2x_2^2x_3^2y\lambda_2 + 6x_1^2x_2^2x_3^2y\lambda_3 + 9x_1^2x_2^2x_3^2\lambda_1\lambda_2 \\
& + 6x_1^2x_2^2x_3^2\lambda_1\lambda_3 + 6x_1^2x_2^2x_3^2\lambda_2\lambda_4 + 4x_1^2x_2^2x_3^2\lambda_3\lambda_4 + 9x_1^2x_2^2x_3y\lambda_2^2 + 12x_1^2x_2^2x_3y\lambda_2\lambda_3 + 3x_1^2x_2^2x_3y\lambda_3^2 \\
& + 9x_1^2x_2^2x_3\lambda_1\lambda_2^2 + 12x_1^2x_2^2x_3\lambda_1\lambda_2\lambda_3 + 3x_1^2x_2^2x_3\lambda_1\lambda_3^2 + 6x_1^2x_2^2x_3\lambda_2^2\lambda_4 + 8x_1^2x_2^2x_3\lambda_2\lambda_3\lambda_4 + 2x_1^2x_2^2x_3\lambda_3^2\lambda_4 \\
& + 3x_1^2x_2^2y\lambda_2^3 + 6x_1^2x_2^2y\lambda_2^2\lambda_3 + 3x_1^2x_2^2y\lambda_2\lambda_3^2 + 3x_1^2x_2^2\lambda_1\lambda_2^3 + 6x_1^2x_2^2\lambda_1\lambda_2^2\lambda_3 + 3x_1^2x_2^2\lambda_1\lambda_2\lambda_3^2 + 2x_1^2x_2^2\lambda_2^3\lambda_4 \\
& + 4x_1^2x_2^2\lambda_2^2\lambda_3\lambda_4 + 2x_1^2x_2^2\lambda_2\lambda_3^2\lambda_4 + 3x_1^2x_2^2x_3^2y + 6x_1^2x_2^2x_3y\lambda_2 + 4x_1^2x_2^2x_3y\lambda_3 + 3x_1^2x_2^2y\lambda_2^2 + 4x_1^2x_2^2y\lambda_2\lambda_3 \\
& + x_1^2x_2^2y\lambda_3^2 + 3x_1x_2x_3^3y^2 + 6x_1x_2x_3^3y\lambda_1 + 4x_1x_2x_3^3y\lambda_4 + 3x_1x_2x_3^3\lambda_1^2 + 4x_1x_2x_3^3\lambda_1\lambda_4 + x_1x_2x_3^3\lambda_4^2 \\
& + 9x_1x_2x_3^2y^2\lambda_2 + 6x_1x_2x_3^2y^2\lambda_3 + 18x_1x_2x_3^2y\lambda_1\lambda_2 + 12x_1x_2x_3^2y\lambda_1\lambda_3 + 12x_1x_2x_3^2y\lambda_2\lambda_4 + 8x_1x_2x_3^2y\lambda_3\lambda_4 \\
& + 9x_1x_2x_3^2\lambda_1^2\lambda_2 + 6x_1x_2x_3^2\lambda_1^2\lambda_3 + 12x_1x_2x_3^2\lambda_1\lambda_2\lambda_4 + 8x_1x_2x_3^2\lambda_1\lambda_3\lambda_4 + 3x_1x_2x_3^2\lambda_2\lambda_4^2 + 2x_1x_2x_3^2\lambda_3\lambda_4^2 \\
& + 9x_1x_2x_3y^2\lambda_2^2 + 12x_1x_2x_3y^2\lambda_2\lambda_3 + 3x_1x_2x_3y^2\lambda_3^2 + 18x_1x_2x_3y\lambda_1\lambda_2^2 + 24x_1x_2x_3y\lambda_1\lambda_2\lambda_3 \\
& + 6x_1x_2x_3y\lambda_1\lambda_3^2 + 12x_1x_2x_3y\lambda_2^2\lambda_4 + 16x_1x_2x_3y\lambda_2\lambda_3\lambda_4 + 4x_1x_2x_3y\lambda_3^2\lambda_4 + 9x_1x_2x_3\lambda_1^2\lambda_2^2 \\
& + 12x_1x_2x_3\lambda_1^2\lambda_2\lambda_3 + 3x_1x_2x_3\lambda_1^2\lambda_3^2 + 12x_1x_2x_3\lambda_1\lambda_2^2\lambda_4 + 16x_1x_2x_3\lambda_1\lambda_2\lambda_3\lambda_4 + 4x_1x_2x_3\lambda_1\lambda_3^2\lambda_4 \\
& + 3x_1x_2x_3\lambda_2^2\lambda_4^2 + 4x_1x_2x_3\lambda_2\lambda_3\lambda_4^2 + x_1x_2x_3\lambda_3^2\lambda_4^2 + 3x_1x_2y^2\lambda_2^3 + 6x_1x_2y^2\lambda_2^2\lambda_3 + 3x_1x_2y^2\lambda_2\lambda_3^2 \\
& + 6x_1x_2y\lambda_1\lambda_2^3 + 12x_1x_2y\lambda_1\lambda_2^2\lambda_3 + 6x_1x_2y\lambda_1\lambda_2\lambda_3^2 + 4x_1x_2y\lambda_2^3\lambda_4 + 8x_1x_2y\lambda_2^2\lambda_3\lambda_4 + 4x_1x_2y\lambda_2\lambda_3^2\lambda_4 \\
& + 3x_1x_2\lambda_1^2\lambda_2^3 + 6x_1x_2\lambda_1^2\lambda_2^2\lambda_3 + 3x_1x_2\lambda_1^2\lambda_2\lambda_3^2 + 4x_1x_2\lambda_1\lambda_2^3\lambda_4 + 8x_1x_2\lambda_1\lambda_2^2\lambda_3\lambda_4 + 4x_1x_2\lambda_1\lambda_2\lambda_3^2\lambda_4 \\
& + x_1x_2\lambda_2^3\lambda_4^2 + 2x_1x_2\lambda_2^2\lambda_3\lambda_4^2 + x_1x_2\lambda_2\lambda_3^2\lambda_4^2 + 6x_1x_2x_3^2y^2 + 6x_1x_2x_3^2y\lambda_1 + 4x_1x_2x_3^2y\lambda_4 + 12x_1x_2x_3y^2\lambda_2 \\
& + 8x_1x_2x_3y^2\lambda_3 + 12x_1x_2x_3y\lambda_1\lambda_2 + 8x_1x_2x_3y\lambda_1\lambda_3 + 8x_1x_2x_3y\lambda_2\lambda_4 + 5x_1x_2x_3y\lambda_3\lambda_4 + 6x_1x_2y^2\lambda_2^2 \\
& + 8x_1x_2y^2\lambda_2\lambda_3 + 2x_1x_2y^2\lambda_3^2 + 6x_1x_2y\lambda_1\lambda_2^2 + 8x_1x_2y\lambda_1\lambda_2\lambda_3 + 2x_1x_2y\lambda_1\lambda_3^2 + 4x_1x_2y\lambda_2^2\lambda_4 \\
& + 5x_1x_2y\lambda_2\lambda_3\lambda_4 + x_1x_2y\lambda_3^2\lambda_4 + x_3^3y^3 + 3x_3^3y^2\lambda_1 + 2x_3^3y^2\lambda_4 + 3x_3^3y\lambda_1^2 + 4x_3^3y\lambda_1\lambda_4 + x_3^3y\lambda_4^2 + x_3^3\lambda_1^3 \\
& + 2x_3^3\lambda_1^2\lambda_4 + x_3^3\lambda_1\lambda_4^2 + 3x_3^2y^3\lambda_2 + 2x_3^2y^3\lambda_3 + 9x_3^2y^2\lambda_1\lambda_2 + 6x_3^2y^2\lambda_1\lambda_3 + 6x_3^2y^2\lambda_2\lambda_4 + 4x_3^2y^2\lambda_3\lambda_4 \\
& + 9x_3^2y\lambda_1^2\lambda_2 + 6x_3^2y\lambda_1^2\lambda_3 + 12x_3^2y\lambda_1\lambda_2\lambda_4 + 8x_3^2y\lambda_1\lambda_3\lambda_4 + 3x_3^2y\lambda_2\lambda_4^2 + 2x_3^2y\lambda_3\lambda_4^2 + 3x_3^2\lambda_1^3\lambda_2 \\
& + 2x_3^2\lambda_1^3\lambda_3 + 6x_3^2\lambda_1^2\lambda_2\lambda_4 + 4x_3^2\lambda_1^2\lambda_3\lambda_4 + 3x_3^2\lambda_1\lambda_2\lambda_4^2 + 2x_3^2\lambda_1\lambda_3\lambda_4^2 + 3x_3y^3\lambda_2^2 + 4x_3y^3\lambda_2\lambda_3 + x_3y^3\lambda_3^2 \\
& + 9x_3y^2\lambda_1\lambda_2^2 + 12x_3y^2\lambda_1\lambda_2\lambda_3 + 3x_3y^2\lambda_1\lambda_3^2 + 6x_3y^2\lambda_2^2\lambda_4 + 8x_3y^2\lambda_2\lambda_3\lambda_4 + 2x_3y^2\lambda_3^2\lambda_4 + 9x_3y\lambda_1^2\lambda_2^2 \\
& + 12x_3y\lambda_1^2\lambda_2\lambda_3 + 3x_3y\lambda_1^2\lambda_3^2 + 12x_3y\lambda_1\lambda_2^2\lambda_4 + 16x_3y\lambda_1\lambda_2\lambda_3\lambda_4 + 4x_3y\lambda_1\lambda_3^2\lambda_4 + 3x_3y\lambda_2^2\lambda_4^2 \\
& + 4x_3y\lambda_2\lambda_3\lambda_4^2 + x_3y\lambda_3^2\lambda_4^2 + 3x_3\lambda_1^3\lambda_2^2 + 4x_3\lambda_1^3\lambda_2\lambda_3 + x_3\lambda_1^3\lambda_3^2 + 6x_3\lambda_1^2\lambda_2^2\lambda_4 + 8x_3\lambda_1^2\lambda_2\lambda_3\lambda_4 + 2x_3\lambda_1^2\lambda_3^2\lambda_4 \\
& + 3x_3\lambda_1\lambda_2^2\lambda_4^2 + 4x_3\lambda_1\lambda_2\lambda_3\lambda_4^2 + x_3\lambda_1\lambda_3^2\lambda_4^2 + y^3\lambda_2^3 + 2y^3\lambda_2^2\lambda_3 + y^3\lambda_2\lambda_3^2 + 3y^2\lambda_1\lambda_2^3 + 6y^2\lambda_1\lambda_2^2\lambda_3 \\
& + 3y^2\lambda_1\lambda_2\lambda_3^2 + 2y^2\lambda_2^3\lambda_4 + 4y^2\lambda_2^2\lambda_3\lambda_4 + 2y^2\lambda_2\lambda_3^2\lambda_4 + 3y\lambda_1^2\lambda_2^3 + 6y\lambda_1^2\lambda_2^2\lambda_3 + 3y\lambda_1^2\lambda_2\lambda_3^2 + 4y\lambda_1\lambda_2^3\lambda_4 \\
& + 8y\lambda_1\lambda_2^2\lambda_3\lambda_4 + 4y\lambda_1\lambda_2\lambda_3^2\lambda_4 + y\lambda_2^3\lambda_4^2 + 2y\lambda_2^2\lambda_3\lambda_4^2 + y\lambda_2\lambda_3^2\lambda_4^2 + \lambda_1^3\lambda_2^3 + 2\lambda_1^3\lambda_2^2\lambda_3 + \lambda_1^3\lambda_2\lambda_3^2 + 2\lambda_1^2\lambda_2^3\lambda_4 \\
& + 4\lambda_1^2\lambda_2^2\lambda_3\lambda_4 + 2\lambda_1^2\lambda_2\lambda_3^2\lambda_4 + \lambda_1\lambda_2^3\lambda_4^2 + 2\lambda_1\lambda_2^2\lambda_3\lambda_4^2 + \lambda_1\lambda_2\lambda_3^2\lambda_4^2 + 3x_1x_2x_3y^2 + 3x_1x_2y^2\lambda_2 + 2x_1x_2y^2\lambda_3 \\
& + 3x_3^2y^3 + 6x_3^2y^2\lambda_1 + 4x_3^2y^2\lambda_4 + 3x_3^2y\lambda_1^2 + 4x_3^2y\lambda_1\lambda_4 + x_3^2y\lambda_4^2 + 6x_3y^3\lambda_2 + 4x_3y^3\lambda_3 + 12x_3y^2\lambda_1\lambda_2 \\
& + 8x_3y^2\lambda_1\lambda_3 + 8x_3y^2\lambda_2\lambda_4 + 5x_3y^2\lambda_3\lambda_4 + 6x_3y\lambda_1^2\lambda_2 + 4x_3y\lambda_1^2\lambda_3 + 8x_3y\lambda_1\lambda_2\lambda_4 + 5x_3y\lambda_1\lambda_3\lambda_4 \\
& + 2x_3y\lambda_2\lambda_4^2 + x_3y\lambda_3\lambda_4^2 + 3y^3\lambda_2^2 + 4y^3\lambda_2\lambda_3 + y^3\lambda_3^2 + 6y^2\lambda_1\lambda_2^2 + 8y^2\lambda_1\lambda_2\lambda_3 + 2y^2\lambda_1\lambda_3^2 + 4y^2\lambda_2^2\lambda_4 \\
& + 5y^2\lambda_2\lambda_3\lambda_4 + y^2\lambda_3^2\lambda_4 + 3y\lambda_1^2\lambda_2^2 + 4y\lambda_1^2\lambda_2\lambda_3 + y\lambda_1^2\lambda_3^2 + 4y\lambda_1\lambda_2^2\lambda_4 + 5y\lambda_1\lambda_2\lambda_3\lambda_4 + y\lambda_1\lambda_3^2\lambda_4 \\
& + y\lambda_2^2\lambda_4^2 + y\lambda_2\lambda_3\lambda_4^2 + x_1x_2x_3y + x_1x_2y\lambda_2 + x_1x_2y\lambda_3 + 3x_3y^3 + 3x_3y^2\lambda_1 + 2x_3y^2\lambda_4 + 3y^3\lambda_2 + 2y^3\lambda_3 \\
& + 3y^2\lambda_1\lambda_2 + 2y^2\lambda_1\lambda_3 + 2y^2\lambda_2\lambda_4 + y^2\lambda_3\lambda_4 + x_3y^2 + x_3y\lambda_1 + y^3 + y^2\lambda_2 + y^2\lambda_3 + y\lambda_1\lambda_2 + y\lambda_1\lambda_3 + y^2 + y
\end{aligned}
$$

$$
\begin{aligned}
p_{\lambda_i}(N_2) = {} & x_1^3x_2^3x_3^3 + 3x_1^3x_2^3x_3^2\lambda_2 + 2x_1^3x_2^3x_3^2\lambda_4 + 3x_1^3x_2^3x_3\lambda_2^2 + 4x_1^3x_2^3x_3\lambda_2\lambda_4 + x_1^3x_2^3x_3\lambda_4^2 + x_1^3x_2^3\lambda_2^3 + 2x_1^3x_2^3\lambda_2^2\lambda_4 \\
& + x_1^3x_2^3\lambda_2\lambda_4^2 + 3x_1^2x_2^2x_3^3y + 3x_1^2x_2^2x_3^3\lambda_1 + 2x_1^2x_2^2x_3^3\lambda_3 + 9x_1^2x_2^2x_3^2y\lambda_2 + 6x_1^2x_2^2x_3^2y\lambda_4 + 9x_1^2x_2^2x_3^2\lambda_1\lambda_2 \\
& + 6x_1^2x_2^2x_3^2\lambda_1\lambda_4 + 6x_1^2x_2^2x_3^2\lambda_2\lambda_3 + 4x_1^2x_2^2x_3^2\lambda_3\lambda_4 + 9x_1^2x_2^2x_3y\lambda_2^2 + 12x_1^2x_2^2x_3y\lambda_2\lambda_4 + 3x_1^2x_2^2x_3y\lambda_4^2 \\
& + 9x_1^2x_2^2x_3\lambda_1\lambda_2^2 + 12x_1^2x_2^2x_3\lambda_1\lambda_2\lambda_4 + 3x_1^2x_2^2x_3\lambda_1\lambda_4^2 + 6x_1^2x_2^2x_3\lambda_2^2\lambda_3 + 8x_1^2x_2^2x_3\lambda_2\lambda_3\lambda_4 + 2x_1^2x_2^2x_3\lambda_3\lambda_4^2 \\
& + 3x_1^2x_2^2y\lambda_2^3 + 6x_1^2x_2^2y\lambda_2^2\lambda_4 + 3x_1^2x_2^2y\lambda_2\lambda_4^2 + 3x_1^2x_2^2\lambda_1\lambda_2^3 + 6x_1^2x_2^2\lambda_1\lambda_2^2\lambda_4 + 3x_1^2x_2^2\lambda_1\lambda_2\lambda_4^2 + 2x_1^2x_2^2\lambda_2^3\lambda_3 \\
& + 4x_1^2x_2^2\lambda_2^2\lambda_3\lambda_4 + 2x_1^2x_2^2\lambda_2\lambda_3\lambda_4^2 + 3x_1^2x_2^2x_3^2y + 6x_1^2x_2^2x_3y\lambda_2 + 4x_1^2x_2^2x_3y\lambda_4 + 3x_1^2x_2^2y\lambda_2^2 + 4x_1^2x_2^2y\lambda_2\lambda_4 \\
& + x_1^2x_2^2y\lambda_4^2 + 3x_1x_2x_3^3y^2 + 6x_1x_2x_3^3y\lambda_1 + 4x_1x_2x_3^3y\lambda_3 + 3x_1x_2x_3^3\lambda_1^2 + 4x_1x_2x_3^3\lambda_1\lambda_3 + x_1x_2x_3^3\lambda_3^2 \\
& + 9x_1x_2x_3^2y^2\lambda_2 + 6x_1x_2x_3^2y^2\lambda_4 + 18x_1x_2x_3^2y\lambda_1\lambda_2 + 12x_1x_2x_3^2y\lambda_1\lambda_4 + 12x_1x_2x_3^2y\lambda_2\lambda_3 + 8x_1x_2x_3^2y\lambda_3\lambda_4 \\
& + 9x_1x_2x_3^2\lambda_1^2\lambda_2 + 6x_1x_2x_3^2\lambda_1^2\lambda_4 + 12x_1x_2x_3^2\lambda_1\lambda_2\lambda_3 + 8x_1x_2x_3^2\lambda_1\lambda_3\lambda_4 + 3x_1x_2x_3^2\lambda_2\lambda_3^2 + 2x_1x_2x_3^2\lambda_3^2\lambda_4 \\
& + 9x_1x_2x_3y^2\lambda_2^2 + 12x_1x_2x_3y^2\lambda_2\lambda_4 + 3x_1x_2x_3y^2\lambda_4^2 + 18x_1x_2x_3y\lambda_1\lambda_2^2 + 24x_1x_2x_3y\lambda_1\lambda_2\lambda_4 \\
& + 6x_1x_2x_3y\lambda_1\lambda_4^2 + 12x_1x_2x_3y\lambda_2^2\lambda_3 + 16x_1x_2x_3y\lambda_2\lambda_3\lambda_4 + 4x_1x_2x_3y\lambda_3\lambda_4^2 + 9x_1x_2x_3\lambda_1^2\lambda_2^2 \\
& + 12x_1x_2x_3\lambda_1^2\lambda_2\lambda_4 + 3x_1x_2x_3\lambda_1^2\lambda_4^2 + 12x_1x_2x_3\lambda_1\lambda_2^2\lambda_3 + 16x_1x_2x_3\lambda_1\lambda_2\lambda_3\lambda_4 + 4x_1x_2x_3\lambda_1\lambda_3\lambda_4^2 \\
& + 3x_1x_2x_3\lambda_2^2\lambda_3^2 + 4x_1x_2x_3\lambda_2\lambda_3^2\lambda_4 + x_1x_2x_3\lambda_3^2\lambda_4^2 + 3x_1x_2y^2\lambda_2^3 + 6x_1x_2y^2\lambda_2^2\lambda_4 + 3x_1x_2y^2\lambda_2\lambda_4^2 \\
& + 6x_1x_2y\lambda_1\lambda_2^3 + 12x_1x_2y\lambda_1\lambda_2^2\lambda_4 + 6x_1x_2y\lambda_1\lambda_2\lambda_4^2 + 4x_1x_2y\lambda_2^3\lambda_3 + 8x_1x_2y\lambda_2^2\lambda_3\lambda_4 + 4x_1x_2y\lambda_2\lambda_3\lambda_4^2 \\
& + 3x_1x_2\lambda_1^2\lambda_2^3 + 6x_1x_2\lambda_1^2\lambda_2^2\lambda_4 + 3x_1x_2\lambda_1^2\lambda_2\lambda_4^2 + 4x_1x_2\lambda_1\lambda_2^3\lambda_3 + 8x_1x_2\lambda_1\lambda_2^2\lambda_3\lambda_4 + 4x_1x_2\lambda_1\lambda_2\lambda_3\lambda_4^2 \\
& + x_1x_2\lambda_2^3\lambda_3^2 + 2x_1x_2\lambda_2^2\lambda_3^2\lambda_4 + x_1x_2\lambda_2\lambda_3^2\lambda_4^2 + 6x_1x_2x_3^2y^2 + 6x_1x_2x_3^2y\lambda_1 + 4x_1x_2x_3^2y\lambda_3 + 12x_1x_2x_3y^2\lambda_2 \\
& + 8x_1x_2x_3y^2\lambda_4 + 12x_1x_2x_3y\lambda_1\lambda_2 + 8x_1x_2x_3y\lambda_1\lambda_4 + 8x_1x_2x_3y\lambda_2\lambda_3 + 5x_1x_2x_3y\lambda_3\lambda_4 + 6x_1x_2y^2\lambda_2^2 \\
& + 8x_1x_2y^2\lambda_2\lambda_4 + 2x_1x_2y^2\lambda_4^2 + 6x_1x_2y\lambda_1\lambda_2^2 + 8x_1x_2y\lambda_1\lambda_2\lambda_4 + 2x_1x_2y\lambda_1\lambda_4^2 + 4x_1x_2y\lambda_2^2\lambda_3 \\
& + 5x_1x_2y\lambda_2\lambda_3\lambda_4 + x_1x_2y\lambda_3\lambda_4^2 + x_3^3y^3 + 3x_3^3y^2\lambda_1 + 2x_3^3y^2\lambda_3 + 3x_3^3y\lambda_1^2 + 4x_3^3y\lambda_1\lambda_3 + x_3^3y\lambda_3^2 + x_3^3\lambda_1^3 \\
& + 2x_3^3\lambda_1^2\lambda_3 + x_3^3\lambda_1\lambda_3^2 + 3x_3^2y^3\lambda_2 + 2x_3^2y^3\lambda_4 + 9x_3^2y^2\lambda_1\lambda_2 + 6x_3^2y^2\lambda_1\lambda_4 + 6x_3^2y^2\lambda_2\lambda_3 + 4x_3^2y^2\lambda_3\lambda_4 \\
& + 9x_3^2y\lambda_1^2\lambda_2 + 6x_3^2y\lambda_1^2\lambda_4 + 12x_3^2y\lambda_1\lambda_2\lambda_3 + 8x_3^2y\lambda_1\lambda_3\lambda_4 + 3x_3^2y\lambda_2\lambda_3^2 + 2x_3^2y\lambda_3^2\lambda_4 + 3x_3^2\lambda_1^3\lambda_2 \\
& + 2x_3^2\lambda_1^3\lambda_4 + 6x_3^2\lambda_1^2\lambda_2\lambda_3 + 4x_3^2\lambda_1^2\lambda_3\lambda_4 + 3x_3^2\lambda_1\lambda_2\lambda_3^2 + 2x_3^2\lambda_1\lambda_3^2\lambda_4 + 3x_3y^3\lambda_2^2 + 4x_3y^3\lambda_2\lambda_4 + x_3y^3\lambda_4^2 \\
& + 9x_3y^2\lambda_1\lambda_2^2 + 12x_3y^2\lambda_1\lambda_2\lambda_4 + 3x_3y^2\lambda_1\lambda_4^2 + 6x_3y^2\lambda_2^2\lambda_3 + 8x_3y^2\lambda_2\lambda_3\lambda_4 + 2x_3y^2\lambda_3\lambda_4^2 + 9x_3y\lambda_1^2\lambda_2^2 \\
& + 12x_3y\lambda_1^2\lambda_2\lambda_4 + 3x_3y\lambda_1^2\lambda_4^2 + 12x_3y\lambda_1\lambda_2^2\lambda_3 + 16x_3y\lambda_1\lambda_2\lambda_3\lambda_4 + 4x_3y\lambda_1\lambda_3\lambda_4^2 + 3x_3y\lambda_2^2\lambda_3^2 \\
& + 4x_3y\lambda_2\lambda_3^2\lambda_4 + x_3y\lambda_3^2\lambda_4^2 + 3x_3\lambda_1^3\lambda_2^2 + 4x_3\lambda_1^3\lambda_2\lambda_4 + x_3\lambda_1^3\lambda_4^2 + 6x_3\lambda_1^2\lambda_2^2\lambda_3 + 8x_3\lambda_1^2\lambda_2\lambda_3\lambda_4 + 2x_3\lambda_1^2\lambda_3\lambda_4^2 \\
& + 3x_3\lambda_1\lambda_2^2\lambda_3^2 + 4x_3\lambda_1\lambda_2\lambda_3^2\lambda_4 + x_3\lambda_1\lambda_3^2\lambda_4^2 + y^3\lambda_2^3 + 2y^3\lambda_2^2\lambda_4 + y^3\lambda_2\lambda_4^2 + 3y^2\lambda_1\lambda_2^3 + 6y^2\lambda_1\lambda_2^2\lambda_4 \\
& + 3y^2\lambda_1\lambda_2\lambda_4^2 + 2y^2\lambda_2^3\lambda_3 + 4y^2\lambda_2^2\lambda_3\lambda_4 + 2y^2\lambda_2\lambda_3\lambda_4^2 + 3y\lambda_1^2\lambda_2^3 + 6y\lambda_1^2\lambda_2^2\lambda_4 + 3y\lambda_1^2\lambda_2\lambda_4^2 + 4y\lambda_1\lambda_2^3\lambda_3 \\
& + 8y\lambda_1\lambda_2^2\lambda_3\lambda_4 + 4y\lambda_1\lambda_2\lambda_3\lambda_4^2 + y\lambda_2^3\lambda_3^2 + 2y\lambda_2^2\lambda_3^2\lambda_4 + y\lambda_2\lambda_3^2\lambda_4^2 + \lambda_1^3\lambda_2^3 + 2\lambda_1^3\lambda_2^2\lambda_4 + \lambda_1^3\lambda_2\lambda_4^2 + 2\lambda_1^2\lambda_2^3\lambda_3 \\
& + 4\lambda_1^2\lambda_2^2\lambda_3\lambda_4 + 2\lambda_1^2\lambda_2\lambda_3\lambda_4^2 + \lambda_1\lambda_2^3\lambda_3^2 + 2\lambda_1\lambda_2^2\lambda_3^2\lambda_4 + \lambda_1\lambda_2\lambda_3^2\lambda_4^2 + 3x_1x_2x_3y^2 + 3x_1x_2y^2\lambda_2 + 2x_1x_2y^2\lambda_4 \\
& + 3x_3^2y^3 + 6x_3^2y^2\lambda_1 + 4x_3^2y^2\lambda_3 + 3x_3^2y\lambda_1^2 + 4x_3^2y\lambda_1\lambda_3 + x_3^2y\lambda_3^2 + 6x_3y^3\lambda_2 + 4x_3y^3\lambda_4 + 12x_3y^2\lambda_1\lambda_2 \\
& + 8x_3y^2\lambda_1\lambda_4 + 8x_3y^2\lambda_2\lambda_3 + 5x_3y^2\lambda_3\lambda_4 + 6x_3y\lambda_1^2\lambda_2 + 4x_3y\lambda_1^2\lambda_4 + 8x_3y\lambda_1\lambda_2\lambda_3 + 5x_3y\lambda_1\lambda_3\lambda_4 \\
& + 2x_3y\lambda_2\lambda_3^2 + x_3y\lambda_3^2\lambda_4 + 3y^3\lambda_2^2 + 4y^3\lambda_2\lambda_4 + y^3\lambda_4^2 + 6y^2\lambda_1\lambda_2^2 + 8y^2\lambda_1\lambda_2\lambda_4 + 2y^2\lambda_1\lambda_4^2 + 4y^2\lambda_2^2\lambda_3 \\
& + 5y^2\lambda_2\lambda_3\lambda_4 + y^2\lambda_3\lambda_4^2 + 3y\lambda_1^2\lambda_2^2 + 4y\lambda_1^2\lambda_2\lambda_4 + y\lambda_1^2\lambda_4^2 + 4y\lambda_1\lambda_2^2\lambda_3 + 5y\lambda_1\lambda_2\lambda_3\lambda_4 + y\lambda_1\lambda_3\lambda_4^2 \\
& + y\lambda_2^2\lambda_3^2 + y\lambda_2\lambda_3^2\lambda_4 + x_1x_2x_3y + x_1x_2y\lambda_2 + 3x_3y^3 + 3x_3y^2\lambda_1 + 2x_3y^2\lambda_3 + 3y^3\lambda_2 + 2y^3\lambda_4 + 3y^2\lambda_1\lambda_2 \\
& + 2y^2\lambda_1\lambda_4 + 2y^2\lambda_2\lambda_3 + y^2\lambda_3\lambda_4 + x_3y^2 + x_3y\lambda_1 + x_3y\lambda_3 + y^3 + y^2\lambda_2 + y\lambda_1\lambda_2 + y\lambda_2\lambda_3 + y^2 + y
\end{aligned}
$$

$$
\begin{aligned}
p_{\lambda_i}(N_1) - p_{\lambda_i}(N_2) = {} & 2x_1^3x_2^3x_3^2\lambda_3 - 2x_1^3x_2^3x_3^2\lambda_4 + 4x_1^3x_2^3x_3\lambda_2\lambda_3 - 4x_1^3x_2^3x_3\lambda_2\lambda_4 + x_1^3x_2^3x_3\lambda_3^2 - x_1^3x_2^3x_3\lambda_4^2 + 2x_1^3x_2^3\lambda_2^2\lambda_3 \\
& - 2x_1^3x_2^3\lambda_2^2\lambda_4 + x_1^3x_2^3\lambda_2\lambda_3^2 - x_1^3x_2^3\lambda_2\lambda_4^2 - 2x_1^2x_2^2x_3^3\lambda_3 + 2x_1^2x_2^2x_3^3\lambda_4 + 6x_1^2x_2^2x_3^2y\lambda_3 - 6x_1^2x_2^2x_3^2y\lambda_4 + 6x_1^2x_2^2x_3^2\lambda_1\lambda_3 \\
& - 6x_1^2x_2^2x_3^2\lambda_1\lambda_4 - 6x_1^2x_2^2x_3^2\lambda_2\lambda_3 + 6x_1^2x_2^2x_3^2\lambda_2\lambda_4 + 12x_1^2x_2^2x_3y\lambda_2\lambda_3 - 12x_1^2x_2^2x_3y\lambda_2\lambda_4 + 3x_1^2x_2^2x_3y\lambda_3^2 \\
& - 3x_1^2x_2^2x_3y\lambda_4^2 + 12x_1^2x_2^2x_3\lambda_1\lambda_2\lambda_3 - 12x_1^2x_2^2x_3\lambda_1\lambda_2\lambda_4 + 3x_1^2x_2^2x_3\lambda_1\lambda_3^2 - 3x_1^2x_2^2x_3\lambda_1\lambda_4^2 - 6x_1^2x_2^2x_3\lambda_2^2\lambda_3 \\
& + 6x_1^2x_2^2x_3\lambda_2^2\lambda_4 + 2x_1^2x_2^2x_3\lambda_3^2\lambda_4 - 2x_1^2x_2^2x_3\lambda_3\lambda_4^2 + 6x_1^2x_2^2y\lambda_2^2\lambda_3 - 6x_1^2x_2^2y\lambda_2^2\lambda_4 + 3x_1^2x_2^2y\lambda_2\lambda_3^2 - 3x_1^2x_2^2y\lambda_2\lambda_4^2 \\
& + 6x_1^2x_2^2\lambda_1\lambda_2^2\lambda_3 - 6x_1^2x_2^2\lambda_1\lambda_2^2\lambda_4 + 3x_1^2x_2^2\lambda_1\lambda_2\lambda_3^2 - 3x_1^2x_2^2\lambda_1\lambda_2\lambda_4^2 - 2x_1^2x_2^2\lambda_2^3\lambda_3 + 2x_1^2x_2^2\lambda_2^3\lambda_4 + 2x_1^2x_2^2\lambda_2\lambda_3^2\lambda_4 \\
& - 2x_1^2x_2^2\lambda_2\lambda_3\lambda_4^2 + 4x_1^2x_2^2x_3y\lambda_3 - 4x_1^2x_2^2x_3y\lambda_4 + 4x_1^2x_2^2y\lambda_2\lambda_3 - 4x_1^2x_2^2y\lambda_2\lambda_4 + x_1^2x_2^2y\lambda_3^2 - x_1^2x_2^2y\lambda_4^2 \\
& - 4x_1x_2x_3^3y\lambda_3 + 4x_1x_2x_3^3y\lambda_4 - 4x_1x_2x_3^3\lambda_1\lambda_3 + 4x_1x_2x_3^3\lambda_1\lambda_4 - x_1x_2x_3^3\lambda_3^2 + x_1x_2x_3^3\lambda_4^2 + 6x_1x_2x_3^2y^2\lambda_3 \\
& - 6x_1x_2x_3^2y^2\lambda_4 + 12x_1x_2x_3^2y\lambda_1\lambda_3 - 12x_1x_2x_3^2y\lambda_1\lambda_4 - 12x_1x_2x_3^2y\lambda_2\lambda_3 + 12x_1x_2x_3^2y\lambda_2\lambda_4 + 6x_1x_2x_3^2\lambda_1^2\lambda_3 \\
& - 6x_1x_2x_3^2\lambda_1^2\lambda_4 - 12x_1x_2x_3^2\lambda_1\lambda_2\lambda_3 + 12x_1x_2x_3^2\lambda_1\lambda_2\lambda_4 - 3x_1x_2x_3^2\lambda_2\lambda_3^2 + 3x_1x_2x_3^2\lambda_2\lambda_4^2 - 2x_1x_2x_3^2\lambda_3^2\lambda_4 \\
& + 2x_1x_2x_3^2\lambda_3\lambda_4^2 + 12x_1x_2x_3y^2\lambda_2\lambda_3 - 12x_1x_2x_3y^2\lambda_2\lambda_4 + 3x_1x_2x_3y^2\lambda_3^2 - 3x_1x_2x_3y^2\lambda_4^2 + 24x_1x_2x_3y\lambda_1\lambda_2\lambda_3 \\
& - 24x_1x_2x_3y\lambda_1\lambda_2\lambda_4 + 6x_1x_2x_3y\lambda_1\lambda_3^2 - 6x_1x_2x_3y\lambda_1\lambda_4^2 - 12x_1x_2x_3y\lambda_2^2\lambda_3 + 12x_1x_2x_3y\lambda_2^2\lambda_4 + 4x_1x_2x_3y\lambda_3^2\lambda_4 \\
& - 4x_1x_2x_3y\lambda_3\lambda_4^2 + 12x_1x_2x_3\lambda_1^2\lambda_2\lambda_3 - 12x_1x_2x_3\lambda_1^2\lambda_2\lambda_4 + 3x_1x_2x_3\lambda_1^2\lambda_3^2 - 3x_1x_2x_3\lambda_1^2\lambda_4^2 - 12x_1x_2x_3\lambda_1\lambda_2^2\lambda_3 \\
& + 12x_1x_2x_3\lambda_1\lambda_2^2\lambda_4 + 4x_1x_2x_3\lambda_1\lambda_3^2\lambda_4 - 4x_1x_2x_3\lambda_1\lambda_3\lambda_4^2 - 3x_1x_2x_3\lambda_2^2\lambda_3^2 + 3x_1x_2x_3\lambda_2^2\lambda_4^2 - 4x_1x_2x_3\lambda_2\lambda_3^2\lambda_4 \\
& + 4x_1x_2x_3\lambda_2\lambda_3\lambda_4^2 + 6x_1x_2y^2\lambda_2^2\lambda_3 - 6x_1x_2y^2\lambda_2^2\lambda_4 + 3x_1x_2y^2\lambda_2\lambda_3^2 - 3x_1x_2y^2\lambda_2\lambda_4^2 + 12x_1x_2y\lambda_1\lambda_2^2\lambda_3 \\
& - 12x_1x_2y\lambda_1\lambda_2^2\lambda_4 + 6x_1x_2y\lambda_1\lambda_2\lambda_3^2 - 6x_1x_2y\lambda_1\lambda_2\lambda_4^2 - 4x_1x_2y\lambda_2^3\lambda_3 + 4x_1x_2y\lambda_2^3\lambda_4 + 4x_1x_2y\lambda_2\lambda_3^2\lambda_4 \\
& - 4x_1x_2y\lambda_2\lambda_3\lambda_4^2 + 6x_1x_2\lambda_1^2\lambda_2^2\lambda_3 - 6x_1x_2\lambda_1^2\lambda_2^2\lambda_4 + 3x_1x_2\lambda_1^2\lambda_2\lambda_3^2 - 3x_1x_2\lambda_1^2\lambda_2\lambda_4^2 - 4x_1x_2\lambda_1\lambda_2^3\lambda_3 \\
& + 4x_1x_2\lambda_1\lambda_2^3\lambda_4 + 4x_1x_2\lambda_1\lambda_2\lambda_3^2\lambda_4 - 4x_1x_2\lambda_1\lambda_2\lambda_3\lambda_4^2 - x_1x_2\lambda_2^3\lambda_3^2 + x_1x_2\lambda_2^3\lambda_4^2 - 2x_1x_2\lambda_2^2\lambda_3^2\lambda_4 + 2x_1x_2\lambda_2^2\lambda_3\lambda_4^2 \\
& - 4x_1x_2x_3^2y\lambda_3 + 4x_1x_2x_3^2y\lambda_4 + 8x_1x_2x_3y^2\lambda_3 - 8x_1x_2x_3y^2\lambda_4 + 8x_1x_2x_3y\lambda_1\lambda_3 - 8x_1x_2x_3y\lambda_1\lambda_4 \\
& - 8x_1x_2x_3y\lambda_2\lambda_3 + 8x_1x_2x_3y\lambda_2\lambda_4 + 8x_1x_2y^2\lambda_2\lambda_3 - 8x_1x_2y^2\lambda_2\lambda_4 + 2x_1x_2y^2\lambda_3^2 - 2x_1x_2y^2\lambda_4^2 + 8x_1x_2y\lambda_1\lambda_2\lambda_3 \\
& - 8x_1x_2y\lambda_1\lambda_2\lambda_4 + 2x_1x_2y\lambda_1\lambda_3^2 - 2x_1x_2y\lambda_1\lambda_4^2 - 4x_1x_2y\lambda_2^2\lambda_3 + 4x_1x_2y\lambda_2^2\lambda_4 + x_1x_2y\lambda_3^2\lambda_4 - x_1x_2y\lambda_3\lambda_4^2 \\
& - 2x_3^3y^2\lambda_3 + 2x_3^3y^2\lambda_4 - 4x_3^3y\lambda_1\lambda_3 + 4x_3^3y\lambda_1\lambda_4 - x_3^3y\lambda_3^2 + x_3^3y\lambda_4^2 - 2x_3^3\lambda_1^2\lambda_3 + 2x_3^3\lambda_1^2\lambda_4 - x_3^3\lambda_1\lambda_3^2 + x_3^3\lambda_1\lambda_4^2 \\
& + 2x_3^2y^3\lambda_3 - 2x_3^2y^3\lambda_4 + 6x_3^2y^2\lambda_1\lambda_3 - 6x_3^2y^2\lambda_1\lambda_4 - 6x_3^2y^2\lambda_2\lambda_3 + 6x_3^2y^2\lambda_2\lambda_4 + 6x_3^2y\lambda_1^2\lambda_3 - 6x_3^2y\lambda_1^2\lambda_4 \\
& - 12x_3^2y\lambda_1\lambda_2\lambda_3 + 12x_3^2y\lambda_1\lambda_2\lambda_4 - 3x_3^2y\lambda_2\lambda_3^2 + 3x_3^2y\lambda_2\lambda_4^2 - 2x_3^2y\lambda_3^2\lambda_4 + 2x_3^2y\lambda_3\lambda_4^2 + 2x_3^2\lambda_1^3\lambda_3 - 2x_3^2\lambda_1^3\lambda_4 \\
& - 6x_3^2\lambda_1^2\lambda_2\lambda_3 + 6x_3^2\lambda_1^2\lambda_2\lambda_4 - 3x_3^2\lambda_1\lambda_2\lambda_3^2 + 3x_3^2\lambda_1\lambda_2\lambda_4^2 - 2x_3^2\lambda_1\lambda_3^2\lambda_4 + 2x_3^2\lambda_1\lambda_3\lambda_4^2 + 4x_3y^3\lambda_2\lambda_3 - 4x_3y^3\lambda_2\lambda_4 \\
& + x_3y^3\lambda_3^2 - x_3y^3\lambda_4^2 + 12x_3y^2\lambda_1\lambda_2\lambda_3 - 12x_3y^2\lambda_1\lambda_2\lambda_4 + 3x_3y^2\lambda_1\lambda_3^2 - 3x_3y^2\lambda_1\lambda_4^2 - 6x_3y^2\lambda_2^2\lambda_3 + 6x_3y^2\lambda_2^2\lambda_4 \\
& + 2x_3y^2\lambda_2^2\lambda_4 - 2x_3y^2\lambda_3\lambda_4^2 + 12x_3y\lambda_1^2\lambda_2\lambda_3 - 12x_3y\lambda_1^2\lambda_2\lambda_4 + 3x_3y\lambda_1^2\lambda_3^2 - 3x_3y\lambda_1^2\lambda_4^2 - 12x_3y\lambda_1\lambda_2^2\lambda_3 \\
& + 12x_3y\lambda_1\lambda_2^2\lambda_4 + 4x_3y\lambda_1\lambda_3^2\lambda_4 - 4x_3y\lambda_1\lambda_3\lambda_4^2 - 3x_3y\lambda_2^2\lambda_3^2 + 3x_3y\lambda_2^2\lambda_4^2 - 4x_3y\lambda_2\lambda_3^2\lambda_4 + 4x_3y\lambda_2\lambda_3\lambda_4^2 \\
& + 4x_3\lambda_1^3\lambda_2\lambda_3 - 4x_3\lambda_1^3\lambda_2\lambda_4 + x_3\lambda_1^3\lambda_3^2 - x_3\lambda_1^3\lambda_4^2 - 6x_3\lambda_1^2\lambda_2^2\lambda_3 + 6x_3\lambda_1^2\lambda_2^2\lambda_4 + 2x_3\lambda_1^2\lambda_3^2\lambda_4 - 2x_3\lambda_1^2\lambda_3\lambda_4^2 \\
& - 3x_3\lambda_1\lambda_2^2\lambda_3^2 + 3x_3\lambda_1\lambda_2^2\lambda_4^2 - 4x_3\lambda_1\lambda_2\lambda_3^2\lambda_4 + 4x_3\lambda_1\lambda_2\lambda_3\lambda_4^2 + 2y^3\lambda_2^2\lambda_3 - 2y^3\lambda_2^2\lambda_4 + y^3\lambda_2\lambda_3^2 - y^3\lambda_2\lambda_4^2 \\
& + 6y^2\lambda_1\lambda_2^2\lambda_3 - 6y^2\lambda_1\lambda_2^2\lambda_4 + 3y^2\lambda_1\lambda_2\lambda_3^2 - 3y^2\lambda_1\lambda_2\lambda_4^2 - 2y^2\lambda_2^3\lambda_3 + 2y^2\lambda_2^3\lambda_4 + 2y^2\lambda_2\lambda_3^2\lambda_4 - 2y^2\lambda_2\lambda_3\lambda_4^2 \\
& + 6y\lambda_1^2\lambda_2^2\lambda_3 - 6y\lambda_1^2\lambda_2^2\lambda_4 + 3y\lambda_1^2\lambda_2\lambda_3^2 - 3y\lambda_1^2\lambda_2\lambda_4^2 - 4y\lambda_1\lambda_2^3\lambda_3 + 4y\lambda_1\lambda_2^3\lambda_4 + 4y\lambda_1\lambda_2\lambda_3^2\lambda_4 - 4y\lambda_1\lambda_2\lambda_3\lambda_4^2 \\
& - y\lambda_2^3\lambda_3^2 + y\lambda_2^3\lambda_4^2 - 2y\lambda_2^2\lambda_3^2\lambda_4 + 2y\lambda_2^2\lambda_3\lambda_4^2 + 2\lambda_1^3\lambda_2^2\lambda_3 - 2\lambda_1^3\lambda_2^2\lambda_4 + \lambda_1^3\lambda_2\lambda_3^2 - \lambda_1^3\lambda_2\lambda_4^2 - 2\lambda_1^2\lambda_2^3\lambda_3 + 2\lambda_1^2\lambda_2^3\lambda_4 \\
& + 2\lambda_1^2\lambda_2\lambda_3^2\lambda_4 - 2\lambda_1^2\lambda_2\lambda_3\lambda_4^2 - \lambda_1\lambda_2^3\lambda_3^2 + \lambda_1\lambda_2^3\lambda_4^2 - 2\lambda_1\lambda_2^2\lambda_3^2\lambda_4 + 2\lambda_1\lambda_2^2\lambda_3\lambda_4^2 + 2x_1x_2y^2\lambda_3 - 2x_1x_2y^2\lambda_4 \\
& - 4x_3^2y^2\lambda_3 + 4x_3^2y^2\lambda_4 - 4x_3^2y\lambda_1\lambda_3 + 4x_3^2y\lambda_1\lambda_4 - x_3^2y\lambda_3^2 + x_3^2y\lambda_4^2 + 4x_3y^3\lambda_3 - 4x_3y^3\lambda_4 + 8x_3y^2\lambda_1\lambda_3 \\
& - 8x_3y^2\lambda_1\lambda_4 - 8x_3y^2\lambda_2\lambda_3 + 8x_3y^2\lambda_2\lambda_4 + 4x_3y\lambda_1^2\lambda_3 - 4x_3y\lambda_1^2\lambda_4 - 8x_3y\lambda_1\lambda_2\lambda_3 + 8x_3y\lambda_1\lambda_2\lambda_4 - 2x_3y\lambda_2\lambda_3^2 \\
& + 2x_3y\lambda_2\lambda_4^2 - x_3y\lambda_3^2\lambda_4 + x_3y\lambda_3\lambda_4^2 + 4y^3\lambda_2\lambda_3 - 4y^3\lambda_2\lambda_4 + y^3\lambda_3^2 - y^3\lambda_4^2 + 8y^2\lambda_1\lambda_2\lambda_3 - 8y^2\lambda_1\lambda_2\lambda_4 \\
& + 2y^2\lambda_1\lambda_3^2 - 2y^2\lambda_1\lambda_4^2 - 4y^2\lambda_2^2\lambda_3 + 4y^2\lambda_2^2\lambda_4 + y^2\lambda_3^2\lambda_4 - y^2\lambda_3\lambda_4^2 + 4y\lambda_1^2\lambda_2\lambda_3 - 4y\lambda_1^2\lambda_2\lambda_4 + y\lambda_1^2\lambda_3^2 - y\lambda_1^2\lambda_4^2 \\
& - 4y\lambda_1\lambda_2^2\lambda_3 + 4y\lambda_1\lambda_2^2\lambda_4 + y\lambda_1\lambda_3^2\lambda_4 - y\lambda_1\lambda_3\lambda_4^2 - y\lambda_2^2\lambda_3^2 + y\lambda_2^2\lambda_4^2 - y\lambda_2^2\lambda_3\lambda_4 + y\lambda_2\lambda_3\lambda_4^2 + x_1x_2y\lambda_3 - 2x_3y^2\lambda_3 \\
& + 2x_3y^2\lambda_4 + 2y^3\lambda_3 - 2y^3\lambda_4 + 2y^2\lambda_1\lambda_3 - 2y^2\lambda_1\lambda_4 - 2y^2\lambda_2\lambda_3 + 2y^2\lambda_2\lambda_4 - x_3y\lambda_3 + y^2\lambda_3 + y\lambda_1\lambda_3 - y\lambda_2\lambda_3
\end{aligned}
$$

$$
\begin{aligned}
p_\lambda(N_1) = {} & 4\lambda^3x_1^3x_2^3 + 8\lambda^2x_1^3x_2^3x_3 + 5\lambda x_1^3x_2^3x_3^2 + x_1^3x_2^3x_3^3 + 20\lambda^4x_1^2x_2^2 + 40\lambda^3x_1^2x_2^2x_3 + 12\lambda^3x_1^2x_2^2y + 25\lambda^2x_1^2x_2^2x_3^2 \\
& + 24\lambda^2x_1^2x_2^2x_3y + 5\lambda x_1^2x_2^2x_3^3 + 15\lambda x_1^2x_2^2x_3^2y + 3x_1^2x_2^2x_3^3y + 32\lambda^5x_1x_2 + 64\lambda^4x_1x_2x_3 + 40\lambda^4x_1x_2y \\
& + 40\lambda^3x_1x_2x_3^2 + 80\lambda^3x_1x_2x_3y + 12\lambda^3x_1x_2y^2 + 8\lambda^2x_1^2x_2^2y + 8\lambda^2x_1x_2x_3^3 + 50\lambda^2x_1x_2x_3^2y + 24\lambda^2x_1x_2x_3y^2 \\
& + 10\lambda x_1^2x_2^2x_3y + 10\lambda x_1x_2x_3^3y + 15\lambda x_1x_2x_3^2y^2 + 3x_1^2x_2^2x_3^2y + 3x_1x_2x_3^3y^2 + 16\lambda^6 + 32\lambda^5x_3 + 32\lambda^5y + 20\lambda^4x_3^2 \\
& + 64\lambda^4x_3y + 20\lambda^4y^2 + 26\lambda^3x_1x_2y + 4\lambda^3x_3^3 + 40\lambda^3x_3^2y + 40\lambda^3x_3y^2 + 4\lambda^3y^3 + 33\lambda^2x_1x_2x_3y + 16\lambda^2x_1x_2y^2 \\
& + 8\lambda^2x_3^3y + 25\lambda^2x_3^2y^2 + 8\lambda^2x_3y^3 + 10\lambda x_1x_2x_3^2y + 20\lambda x_1x_2x_3y^2 + 5\lambda x_3^3y^2 + 5\lambda x_3^2y^3 + 6x_1x_2x_3^2y^2 + x_3^3y^3 \\
& + 20\lambda^4y + 26\lambda^3x_3y + 26\lambda^3y^2 + 8\lambda^2x_3^2y + 33\lambda^2x_3y^2 + 8\lambda^2y^3 + 5\lambda x_1x_2y^2 + 10\lambda x_3^2y^2 + 10\lambda x_3y^3 + 3x_1x_2x_3y^2 \\
& + 3x_3^2y^3 + 8\lambda^2y^2 + \boldsymbol{2\lambda x_1x_2y} + 5\lambda x_3y^2 + 5\lambda y^3 + x_1x_2x_3y + 3x_3y^3 + 2\lambda^2y + \boldsymbol{\lambda x_3y} + \boldsymbol{2\lambda y^2} + x_3y^2 + y^3 + y^2 + y
\end{aligned}
$$

$$
\begin{aligned}
p_\lambda(N_2) = {} & 4\lambda^3 x_1^3 x_2^3 + 8\lambda^2 x_1^3 x_2^3 x_3 + 5\lambda x_1^3 x_2^3 x_3^2 + x_1^3 x_2^3 x_3^3 + 20\lambda^4 x_1^2 x_2^2 + 40\lambda^3 x_1^2 x_2^2 x_3 + 12\lambda^3 x_1^2 x_2^2 y + 25\lambda^2 x_1^2 x_2^2 x_3^2 \\
& + 24\lambda^2 x_1^2 x_2^2 x_3 y + 5\lambda x_1^2 x_2^2 x_3^3 + 15\lambda x_1^2 x_2^2 x_3^2 y + 3 x_1^2 x_2^2 x_3^3 y + 32\lambda^5 x_1 x_2 + 64\lambda^4 x_1 x_2 x_3 + 40\lambda^4 x_1 x_2 y \\
& + 40\lambda^3 x_1 x_2 x_3^2 + 80\lambda^3 x_1 x_2 x_3 y + 12\lambda^3 x_1 x_2 y^2 + 8\lambda^2 x_1^2 x_2^2 y + 8\lambda^2 x_1 x_2 x_3^3 + 50\lambda^2 x_1 x_2 x_3^2 y + 24\lambda^2 x_1 x_2 x_3 y^2 \\
& + 10\lambda x_1^2 x_2^2 x_3 y + 10\lambda x_1 x_2 x_3^3 y + 15\lambda x_1 x_2 x_3^2 y^2 + 3 x_1^2 x_2^2 x_3^2 y + 3 x_1 x_2 x_3^3 y^2 + 16\lambda^6 + 32\lambda^5 x_3 + 32\lambda^5 y + 20\lambda^4 x_3^2 \\
& + 64\lambda^4 x_3 y + 20\lambda^4 y^2 + 26\lambda^3 x_1 x_2 y + 4\lambda^3 x_3^3 + 40\lambda^3 x_3^2 y + 40\lambda^3 x_3 y^2 + 4\lambda^3 y^3 + 33\lambda^2 x_1 x_2 x_3 y + 16\lambda^2 x_1 x_2 y^2 \\
& + 8\lambda^2 x_3^3 y + 25\lambda^2 x_3^2 y^2 + 8\lambda^2 x_3 y^3 + 10\lambda x_1 x_2 x_3^2 y + 20\lambda x_1 x_2 x_3 y^2 + 5\lambda x_3^3 y^2 + 5\lambda x_3^2 y^3 + 6 x_1 x_2 x_3^2 y^2 + x_3^3 y^3 \\
& + 20\lambda^4 y + 26\lambda^3 x_3 y + 26\lambda^3 y^2 + 8\lambda^2 x_3^2 y + 33\lambda^2 x_3 y^2 + 8\lambda^2 y^3 + 5\lambda x_1 x_2 y^2 + 10\lambda x_3^2 y^2 + 10\lambda x_3 y^3 + 3 x_1 x_2 x_3 y^2 \\
& + 3 x_3^2 y^3 + 8\lambda^2 y^2 + \boldsymbol{\lambda x_1 x_2 y} + 5\lambda x_3 y^2 + 5\lambda y^3 + x_1 x_2 x_3 y + 3 x_3 y^3 + 2\lambda^2 y + \boldsymbol{2\lambda x_3 y} + \boldsymbol{\lambda y^2} + x_3 y^2 + y^3 + y^2 + y
\end{aligned}
$$

B

Maple code

Distinct labels

$p\_u6 := x\_1 \cdot x\_2 + y$

$$p\_u6 := x\_1\, x\_2 + y \tag{1.1}$$

Pons polynomial

$p\_ol1 := \lambda\_1 \cdot p\_u6$

$$p\_ol1 := \lambda\_1\, (x\_1\, x\_2 + y) \tag{1.1.1}$$

$p\_ol2 := \lambda\_2 \cdot x\_3$

$$p\_ol2 := \lambda\_2\, x\_3 \tag{1.1.2}$$

$p\_ou3 := y + p\_ol1 \cdot \lambda\_3 \cdot p\_ol2$

$$p\_ou3 := \lambda\_1\, (x\_1\, x\_2 + y)\, \lambda\_2\, x\_3\, \lambda\_3 + y \tag{1.1.3}$$

$p\_ou4 := y + \lambda\_3 \cdot p\_ol2 \cdot \lambda\_4 \cdot p\_ol1$

$$p\_ou4 := \lambda\_1\, (x\_1\, x\_2 + y)\, \lambda\_2\, x\_3\, \lambda\_3\, \lambda\_4 + y \tag{1.1.4}$$

$p\_ou5 := y + \lambda\_4 \cdot p\_ol1 \cdot p\_ol2$

$$p\_ou5 := \lambda\_1\, (x\_1\, x\_2 + y)\, \lambda\_2\, x\_3\, \lambda\_4 + y \tag{1.1.5}$$

$p\_ou2 := y + p\_ou4 \cdot p\_ou5$

$$p\_ou2 := (\lambda\_1\, (x\_1\, x\_2 + y)\, \lambda\_2\, x\_3\, \lambda\_3\, \lambda\_4 + y)\, (\lambda\_1\, (x\_1\, x\_2 + y)\, \lambda\_2\, x\_3\, \lambda\_4 + y) + y \tag{1.1.6}$$

$p\_ou1 := y + p\_ou2 \cdot p\_ou3$

$$p\_ou1 := ((\lambda\_1\, (x\_1\, x\_2 + y)\, \lambda\_2\, x\_3\, \lambda\_3\, \lambda\_4 + y)\, (\lambda\_1\, (x\_1\, x\_2 + y)\, \lambda\_2\, x\_3\, \lambda\_4 + y) + y)\, (\lambda\_1\, (x\_1\, x\_2 + y)\, \lambda\_2\, x\_3\, \lambda\_3 + y) + y \tag{1.1.7}$$

$expand(p\_ou1)$

$$\begin{aligned} &x\_1^3\, x\_2^3\, x\_3^3\, \lambda\_1^3\, \lambda\_2^3\, \lambda\_3^2\, \lambda\_4^2 + 3\, x\_1^2\, x\_2^2\, x\_3^3\, y\, \lambda\_1^3\, \lambda\_2^3\, \lambda\_3^2\, \lambda\_4^2 \\ &+ 3\, x\_1\, x\_2\, x\_3^3\, y^2\, \lambda\_1^3\, \lambda\_2^3\, \lambda\_3^2\, \lambda\_4^2 + x\_3^3\, y^3\, \lambda\_1^3\, \lambda\_2^3\, \lambda\_3^2\, \lambda\_4^2 \\ &+ x\_1^2\, x\_2^2\, x\_3^2\, y\, \lambda\_1^2\, \lambda\_2^2\, \lambda\_3^2\, \lambda\_4 + x\_1^2\, x\_2^2\, x\_3^2\, y\, \lambda\_1^2\, \lambda\_2^2\, \lambda\_3\, \lambda\_4^2 \\ &+ x\_1^2\, x\_2^2\, x\_3^2\, y\, \lambda\_1^2\, \lambda\_2^2\, \lambda\_3\, \lambda\_4 + 2\, x\_1\, x\_2\, x\_3^2\, y^2\, \lambda\_1^2\, \lambda\_2^2\, \lambda\_3^2\, \lambda\_4 \\ &+ 2\, x\_1\, x\_2\, x\_3^2\, y^2\, \lambda\_1^2\, \lambda\_2^2\, \lambda\_3\, \lambda\_4^2 + 2\, x\_1\, x\_2\, x\_3^2\, y^2\, \lambda\_1^2\, \lambda\_2^2\, \lambda\_3\, \lambda\_4 \\ &+ x\_3^2\, y^3\, \lambda\_1^2\, \lambda\_2^2\, \lambda\_3^2\, \lambda\_4 + x\_3^2\, y^3\, \lambda\_1^2\, \lambda\_2^2\, \lambda\_3\, \lambda\_4^2 \\ &+ x\_3^2\, y^3\, \lambda\_1^2\, \lambda\_2^2\, \lambda\_3\, \lambda\_4 + x\_1\, x\_2\, x\_3\, y^2\, \lambda\_1\, \lambda\_2\, \lambda\_3\, \lambda\_4 \\ &+ x\_1\, x\_2\, x\_3\, y^2\, \lambda\_1\, \lambda\_2\, \lambda\_3 + x\_1\, x\_2\, x\_3\, y^2\, \lambda\_1\, \lambda\_2\, \lambda\_4 \\ &+ x\_3\, y^3\, \lambda\_1\, \lambda\_2\, \lambda\_3\, \lambda\_4 + x\_1\, x\_2\, x\_3\, y\, \lambda\_1\, \lambda\_2\, \lambda\_3 + x\_3\, y^3\, \lambda\_1\, \lambda\_2\, \lambda\_3 \\ &+ x\_3\, y^3\, \lambda\_1\, \lambda\_2\, \lambda\_4 + x\_3\, y^2\, \lambda\_1\, \lambda\_2\, \lambda\_3 + y^3 + y^2 + y \end{aligned} \tag{1.1.8}$$

My polynomial

$p\_nl2 := \lambda\_2 + x\_3$

$$p\_nl2 := \lambda\_2 + x\_3 \quad \textbf{(1.2.1)}$$

$p\_nl1 := \lambda\_1 + p\_u6$

$$p\_nl1 := x\_1\, x\_2 + y + \lambda\_1 \quad \textbf{(1.2.2)}$$

Polynomial for the left network

$p\_nll3 := \lambda\_3 + p\_nl2$

$$p\_nll3 := \lambda\_3 + \lambda\_2 + x\_3 \quad \textbf{(1.2.1.1)}$$

$p\_nll4 := \lambda\_4 + p\_nl1$

$$p\_nll4 := x\_1\, x\_2 + y + \lambda\_1 + \lambda\_4 \quad \textbf{(1.2.1.2)}$$

$p\_nlu3 := y + p\_nl1 \cdot p\_nll3$

$$p\_nlu3 := (x\_1\, x\_2 + y + \lambda\_1)\,(\lambda\_3 + \lambda\_2 + x\_3) + y \quad \textbf{(1.2.1.3)}$$

$p\_nlu4 := y + p\_nll3 \cdot p\_nll4$

$$p\_nlu4 := (\lambda\_3 + \lambda\_2 + x\_3)\,(x\_1\, x\_2 + y + \lambda\_1 + \lambda\_4) + y \quad \textbf{(1.2.1.4)}$$

$p\_nlu5 := y + p\_nll4 \cdot p\_nl2$

$$p\_nlu5 := (\lambda\_2 + x\_3)\,(x\_1\, x\_2 + y + \lambda\_1 + \lambda\_4) + y \quad \textbf{(1.2.1.5)}$$

$p\_nlu2 := y + p\_nlu5 \cdot p\_nlu4$

$$p\_nlu2 := ((\lambda\_3 + \lambda\_2 + x\_3)\,(x\_1\, x\_2 + y + \lambda\_1 + \lambda\_4) + y)\,((\lambda\_2 + x\_3)\,(x\_1\, x\_2 + y + \lambda\_1 + \lambda\_4) + y) + y \quad \textbf{(1.2.1.6)}$$

$p\_nlu1 := y + p\_nlu2 \cdot p\_nlu3$

$$p\_nlu1 := (((\lambda\_3 + \lambda\_2 + x\_3)\,(x\_1\, x\_2 + y + \lambda\_1 + \lambda\_4) + y)\,((\lambda\_2 + x\_3)\,(x\_1\, x\_2 + y + \lambda\_1 + \lambda\_4) + y) + y)\,((x\_1\, x\_2 + y + \lambda\_1)\,(\lambda\_3 + \lambda\_2 + x\_3) + y) + y \quad \textbf{(1.2.1.7)}$$

Polynomial for the right network

$p\_nrl3 := \lambda\_3 + p\_nl1$

$$p\_nrl3 := x\_1\, x\_2 + y + \lambda\_1 + \lambda\_3 \quad \textbf{(1.2.2.1)}$$

$p\_nrl4 := \lambda\_4 + p\_nl2$

$$p\_nrl4 := \lambda\_4 + \lambda\_2 + x\_3 \quad \textbf{(1.2.2.2)}$$

$p\_nru3 := y + p\_nrl3 \cdot p\_nl2$

$$p\_nru3 := (\lambda\_2 + x\_3)\,(x\_1\, x\_2 + y + \lambda\_1 + \lambda\_3) + y \quad \textbf{(1.2.2.3)}$$

$p\_nru4 := y + p\_nrl3 \cdot p\_nrl4$

$$p\_nru4 := (x\_1\, x\_2 + y + \lambda\_1 + \lambda\_3)\,(\lambda\_4 + \lambda\_2 + x\_3) + y \quad \textbf{(1.2.2.4)}$$

$p\_nru5 := y + p\_nrl4 \cdot p\_nl1$

$$p\_nru5 := (x\_1\, x\_2 + y + \lambda\_1)\,(\lambda\_4 + \lambda\_2 + x\_3) + y \quad \textbf{(1.2.2.5)}$$

$p\_nru2 := y + p\_nru4 \cdot p\_nru5$

$$p\_nru2 := ((x\_1\,x\_2 + y + \lambda\_1 + \lambda\_3)\,(\lambda\_4 + \lambda\_2 + x\_3) + y)\,((x\_1\,x\_2 + y + \lambda\_1)\,(\lambda\_4 + \lambda\_2 + x\_3) + y) + y \quad (1.2.2.6)$$

$p\_nru1 := y + p\_nru2 \cdot p\_nru3$

$$p\_nru1 := (((x\_1\,x\_2 + y + \lambda\_1 + \lambda\_3)\,(\lambda\_4 + \lambda\_2 + x\_3) + y)\,((x\_1\,x\_2 + y + \lambda\_1)\,(\lambda\_4 + \lambda\_2 + x\_3) + y) + y)\,((\lambda\_2 + x\_3)\,(x\_1\,x\_2 + y + \lambda\_1 + \lambda\_3) + y) + y \quad (1.2.2.7)$$

$evalb(p\_nlu1 = p\_nru1)$

$$false \quad (1.2.3)$$

$pr := expand(p\_nru1):$
$pl := expand(p\_nlu1):$
$pl - pr:$

One label

$p\_u6 := x\_1 \cdot x\_2 + y$

$$p\_u6 := x\_1\,x\_2 + y \quad (2.1)$$

Pons Polynomial

$p\_ol1 := \lambda \cdot p\_u6$

$$p\_ol1 := \lambda\,(x\_1\,x\_2 + y) \quad (2.1.1)$$

$p\_ol2 := \lambda \cdot x\_3$

$$p\_ol2 := \lambda\,x\_3 \quad (2.1.2)$$

$p\_ou3 := y + p\_ol1 \cdot \lambda \cdot p\_ol2$

$$p\_ou3 := \lambda^3\,(x\_1\,x\_2 + y)\,x\_3 + y \quad (2.1.3)$$

$p\_ou4 := y + \lambda \cdot p\_ol2 \cdot \lambda \cdot p\_ol1$

$$p\_ou4 := \lambda^4\,(x\_1\,x\_2 + y)\,x\_3 + y \quad (2.1.4)$$

$p\_ou5 := y + \lambda \cdot p\_ol1 \cdot p\_ol2$

$$p\_ou5 := \lambda^3\,(x\_1\,x\_2 + y)\,x\_3 + y \quad (2.1.5)$$

$p\_ou2 := y + p\_ou4 \cdot p\_ou5$

$$p\_ou2 := (\lambda^4\,(x\_1\,x\_2 + y)\,x\_3 + y)\,(\lambda^3\,(x\_1\,x\_2 + y)\,x\_3 + y) + y \quad (2.1.6)$$

$p\_ou1 := y + p\_ou2 \cdot p\_ou3$

$$p\_ou1 := ((\lambda^4\,(x\_1\,x\_2 + y)\,x\_3 + y)\,(\lambda^3\,(x\_1\,x\_2 + y)\,x\_3 + y) + y)\,(\lambda^3\,(x\_1\,x\_2 + y)\,x\_3 + y) + y \quad (2.1.7)$$

$expand(p\_ou1)$

$$\lambda^{10}\,x\_1^3\,x\_2^3\,x\_3^3 + 3\,\lambda^{10}\,x\_1^2\,x\_2^2\,x\_3^3\,y + 3\,\lambda^{10}\,x\_1\,x\_2\,x\_3^3\,y^2 + \lambda^{10}\,x\_3^3\,y^3 \quad (2.1.8)$$

$$+ 2\,\lambda^7\,x\_1^2\,x\_2^2\,x\_3^2\,y + 4\,\lambda^7\,x\_1\,x\_2\,x\_3^2\,y^2 + \lambda^6\,x\_1^2\,x\_2^2\,x\_3^2\,y + 2\,\lambda^7\,x\_3^2\,y^3$$
$$+ 2\,\lambda^6\,x\_1\,x\_2\,x\_3^2\,y^2 + \lambda^6\,x\_3^2\,y^3 + \lambda^4\,x\_1\,x\_2\,x\_3\,y^2 + \lambda^4\,x\_3\,y^3$$
$$+ 2\,\lambda^3\,x\_1\,x\_2\,x\_3\,y^2 + \lambda^3\,x\_1\,x\_2\,x\_3\,y + 2\,\lambda^3\,x\_3\,y^3 + \lambda^3\,x\_3\,y^2 + y^3 + y^2 + y$$

My polynomial

$p\_nl2 := \lambda + x\_3$

$$p\_nl2 := \lambda + x\_3 \qquad \textbf{(2.2.1)}$$

$p\_nl1 := \lambda + p\_u6$

$$p\_nl1 := x\_1\,x\_2 + \lambda + y \qquad \textbf{(2.2.2)}$$

left

$p\_nll3 := \lambda + p\_nl2$

$$p\_nll3 := 2\,\lambda + x\_3 \qquad \textbf{(2.2.1.1)}$$

$p\_nll4 := \lambda + p\_nl1$

$$p\_nll4 := x\_1\,x\_2 + 2\,\lambda + y \qquad \textbf{(2.2.1.2)}$$

$p\_nlu3 := y + p\_nl1 \cdot p\_nll3$

$$p\_nlu3 := (x\_1\,x\_2 + \lambda + y)\,(2\,\lambda + x\_3) + y \qquad \textbf{(2.2.1.3)}$$

$p\_nlu4 := y + p\_nll3 \cdot p\_nll4$

$$p\_nlu4 := (2\,\lambda + x\_3)\,(x\_1\,x\_2 + 2\,\lambda + y) + y \qquad \textbf{(2.2.1.4)}$$

$p\_nlu5 := y + p\_nll4 \cdot p\_nl2$

$$p\_nlu5 := (\lambda + x\_3)\,(x\_1\,x\_2 + 2\,\lambda + y) + y \qquad \textbf{(2.2.1.5)}$$

$p\_nlu2 := y + p\_nlu5 \cdot p\_nlu4$

$$p\_nlu2 := ((2\,\lambda + x\_3)\,(x\_1\,x\_2 + 2\,\lambda + y) + y)\,((\lambda + x\_3)\,(x\_1\,x\_2 + 2\,\lambda + y) + y) + y \qquad \textbf{(2.2.1.6)}$$

$p\_nlu1 := y + p\_nlu2 \cdot p\_nlu3$

$$p\_nlu1 := (((2\,\lambda + x\_3)\,(x\_1\,x\_2 + 2\,\lambda + y) + y)\,((\lambda + x\_3)\,(x\_1\,x\_2 + 2\,\lambda + y) + y) + y)\,((x\_1\,x\_2 + \lambda + y)\,(2\,\lambda + x\_3) + y) + y \qquad \textbf{(2.2.1.7)}$$

right

$p\_nrl3 := \lambda + p\_nl1$

$$p\_nrl3 := x\_1\,x\_2 + 2\,\lambda + y \qquad \textbf{(2.2.2.1)}$$

$p\_nrl4 := \lambda + p\_nl2$

$$p\_nrl4 := 2\,\lambda + x\_3 \qquad \textbf{(2.2.2.2)}$$

$p\_nru3 := y + p\_nrl3 \cdot p\_nl2$

$$p\_nru3 := (\lambda + x\_3)\,(x\_1\,x\_2 + 2\,\lambda + y) + y \qquad \textbf{(2.2.2.3)}$$

$p\_nru4 := y + p\_nrl3 \cdot p\_nrl4$

$$p\_nru4 := (2\,\lambda + x\_3)\,(x\_1\,x\_2 + 2\,\lambda + y) + y \qquad \textbf{(2.2.2.4)}$$

$p\_nru5 := y + p\_nrl4 \cdot p\_nl1$

$$p\_nru5 := (x\_1\,x\_2 + \lambda + y)\,(2\,\lambda + x\_3) + y \qquad \textbf{(2.2.2.5)}$$

$p\_nru2 := y + p\_nru4 \cdot p\_nru5$

$$p\_nru2 := ((2\,\lambda + x\_3)\,(x\_1\,x\_2 + 2\,\lambda + y) + y)\,((x\_1\,x\_2 + \lambda + y)\,(2\,\lambda + x\_3) + y) + y \qquad \textbf{(2.2.2.6)}$$

$p\_nru1 := y + p\_nru2 \cdot p\_nru3$

$$p\_nru1 := (((2\,\lambda + x\_3)\,(x\_1\,x\_2 + 2\,\lambda + y) + y)\,((x\_1\,x\_2 + \lambda + y)\,(2\,\lambda + x\_3) + y) + y)\,((\lambda + x\_3)\,(x\_1\,x\_2 + 2\,\lambda + y) + y) + y \qquad \textbf{(2.2.2.7)}$$

$evalb(p\_nlu1 = p\_nru1)$

$$false \qquad \textbf{(2.2.3)}$$

$pr := expand(p\_nru1):$
$pl := expand(p\_nlu1):$
$pl - pr$

$$\lambda\,x\_1\,x\_2\,y - \lambda\,x\_3\,y + \lambda\,y^2 \qquad \textbf{(2.2.4)}$$

Algorithm

$factor(pr - y)$

$$(x\_2\,x\_1\,\lambda + x\_1\,x\_2\,x\_3 + 2\,\lambda^2 + 2\,\lambda\,x\_3 + \lambda\,y + x\_3\,y + y)\,(4\,\lambda^2\,x\_1^2\,x\_2^2 + 4\,\lambda\,x\_1^2\,x\_2^2\,x\_3 + x\_1^2\,x\_2^2\,x\_3^2 + 12\,\lambda^3\,x\_1\,x\_2 + 12\,\lambda^2\,x\_1\,x\_2\,x\_3 + 8\,\lambda^2\,x\_1\,x\_2\,y + 3\,\lambda\,x\_1\,x\_2\,x\_3^2 + 8\,\lambda\,x\_1\,x\_2\,x\_3\,y + 2\,x\_1\,x\_2\,x\_3^2\,y + 8\,\lambda^4 + 8\,\lambda^3\,x\_3 + 12\,\lambda^3\,y + 2\,\lambda^2\,x\_3^2 + 12\,\lambda^2\,x\_3\,y + 4\,\lambda^2\,y^2 + 4\,\lambda\,x\_1\,x\_2\,y + 3\,\lambda\,x\_3^2\,y + 4\,\lambda\,x\_3\,y^2 + 2\,x\_1\,x\_2\,x\_3\,y + x\_3^2\,y^2 + 6\,\lambda^2\,y + 3\,\lambda\,x\_3\,y + 4\,\lambda\,y^2 + 2\,x\_3\,y^2 + y^2 + y) \qquad \textbf{(2.3.1)}$$

$pr1 := (x\_2\,x\_1\,\lambda + x\_1\,x\_2\,x\_3 + 2\,\lambda^2 + 2\,\lambda\,x\_3 + \lambda\,y + x\_3\,y + y):$
$pr2 := (4\,\lambda^2\,x\_1^2\,x\_2^2 + 4\,\lambda\,x\_1^2\,x\_2^2\,x\_3 + x\_1^2\,x\_2^2\,x\_3^2 + 12\,\lambda^3\,x\_1\,x\_2 + 12\,\lambda^2\,x\_1\,x\_2\,x\_3 + 8\,\lambda^2\,x\_1\,x\_2\,y + 3\,\lambda\,x\_1\,x\_2\,x\_3^2 + 8\,\lambda\,x\_1\,x\_2\,x\_3\,y + 2\,x\_1\,x\_2\,x\_3^2\,y + 8\,\lambda^4 + 8\,\lambda^3\,x\_3 + 12\,\lambda^3\,y + 2\,\lambda^2\,x\_3^2 + 12\,\lambda^2\,x\_3\,y + 4\,\lambda^2\,y^2 + 4\,\lambda\,x\_1\,x\_2\,y + 3\,\lambda\,x\_3^2\,y + 4\,\lambda\,x\_3\,y^2 + 2\,x\_1\,x\_2\,x\_3\,y + x\_3^2\,y^2 + 6\,\lambda^2\,y + 3\,\lambda\,x\_3\,y + 4\,\lambda\,y^2 + 2\,x\_3\,y^2 + y^2 + y):$
$a := factor(pr1 - y)$

$$a := (\lambda + x\_3)\,(x\_1\,x\_2 + 2\,\lambda + y) \qquad \textbf{(2.3.2)}$$

$$factor(pr2 - y)$$

$$\left(2\,x\_2\,x\_1\,\lambda + x\_1\,x\_2\,x\_3 + 4\,\lambda^2 + 2\,\lambda\,x\_3 + 2\,\lambda\,y + x\_3\,y + y\right)\left(2\,x\_2\,x\_1\,\lambda + x\_1\,x\_2\,x\_3 + 2\,\lambda^2 + \lambda\,x\_3 + 2\,\lambda\,y + x\_3\,y + y\right) \tag{2.3.3}$$

$$pr21 := \left(2\,x\_2\,x\_1\,\lambda + x\_1\,x\_2\,x\_3 + 4\,\lambda^2 + 2\,\lambda\,x\_3 + 2\,\lambda\,y + x\_3\,y + y\right):$$

$$pr22 := \left(2\,x\_2\,x\_1\,\lambda + x\_1\,x\_2\,x\_3 + 2\,\lambda^2 + \lambda\,x\_3 + 2\,\lambda\,y + x\_3\,y + y\right):$$

$$b := factor(pr21 - y)$$

$$b := \left(2\,\lambda + x\_3\right)\left(x\_1\,x\_2 + 2\,\lambda + y\right) \tag{2.3.4}$$

$$c := factor(pr22 - y)$$

$$c := \left(x\_1\,x\_2 + \lambda + y\right)\left(2\,\lambda + x\_3\right) \tag{2.3.5}$$

$$p := ((c + y)\cdot(b + y) + y)\cdot(a + y) + y$$

$$p := \left(\left(\left(2\,\lambda + x\_3\right)\left(x\_1\,x\_2 + 2\,\lambda + y\right) + y\right)\left(\left(x\_1\,x\_2 + \lambda + y\right)\left(2\,\lambda + x\_3\right) + y\right) + y\right)\left(\left(\lambda + x\_3\right)\left(x\_1\,x\_2 + 2\,\lambda + y\right) + y\right) + y \tag{2.3.6}$$

$$evalb(expand(p) = pr)$$

$$true \tag{2.3.7}$$

References

[1] P. Liu, "A tree distinguishing polynomial," *Discrete Applied Mathematics*, vol. 288, pp. 1–8, 2021, ISSN: 0166-218X. DOI: https://doi.org/10.1016/j.dam.2020.08.019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0166218X20303851.

[2] J. C. Pons, T. M. Coronado, M. Hendriksen, and A. Francis, "A polynomial invariant for a new class of phylogenetic networks," *PLOS ONE*, vol. 17, no. 5, pp. 1–22, May 2022. DOI: 10.1371/journal.pone.0268181. [Online]. Available: https://doi.org/10.1371/journal.pone.0268181.