Stockholms universitet

LCA-Relations, Canonical Networks and Phylogenetic Trees

LCA-relationer, kanoniska nätverk och fylogenetiska träd.

Jon Bryder

Handledare: Marc Hellmuth
Examinator: Lars Arvestad
Inlämningsdatum: 2026-05-30

Kandidatuppsats i datalogi
Bachelor Thesis in Computer Science

Abstract

A relation R on $\mathscr{P}_2(X) \times \mathscr{P}_2(X)$ is *realizable* if there exists a directed acyclic graph (DAG) G on leaf set X such that R is realized by G in the sense of Lindeberg et al. [1]. A least common ancestor (LCA) is the vertex of two leaves in a DAG such that no proper descendant of this vertex is also the ancestor of the two leaves. R can be interpreted as a collection of LCA conditions and G realizes R if it can maintain all these conditions. In 2025, Lindeberg et al. characterised the conditions under which R is realizable and introduced Algorithm 1, which decides realizability and, in the affirmative case, constructs the canonical DAG G_R and the canonical network N_R.

In this thesis, we study the relationship between the clustering systems of phylogenetic trees and the canonical networks produced by the realizability framework of Lindeberg et al. [1]. Given one or more phylogenetic trees on a common leaf set X, we introduce the ExtractR algorithm, which extracts the lca-comparison relation $\trianglelefteq_{T_i}$ from each tree T_i and consolidates them into a single relation $R = \bigcup_i \trianglelefteq_{T_i}$. This relation is passed to Algorithm 1 of Lindeberg et al., which either certifies that R is realizable and returns G_R and N_R, or reports a falsifying constraint.

The central contribution of this paper is a proof of the equivalence

$$\mathscr{C}(T) \subseteq \mathscr{C}(T') \iff T \simeq N_R,$$

for any two phylogenetic trees T and T' on X, where N_R is the canonical network built from $R = \trianglelefteq_T \cup \trianglelefteq_{T'}$. The proof proceeds via two auxiliary lemmas: the first characterises cluster containment in terms of the lca-comparison relations, and the second characterises isomorphism with N_R in terms of the closure R^+. We extend this result to collections of n trees and derive a corollary for the star tree. Prior to the proof, the equivalence is verified computationally for leaf set sizes $|X| = 2$ through 10, with 100 randomly generated tree pairs per size; no counterexamples were found.

Keywords: DAG, networks, phylogenetic trees, clustering systems, realizability

Sammanfattning

En relation R på $\mathscr{P}_2(X) \times \mathscr{P}_2(X)$ är *realiserbar* om det finns en riktad acyklisk graf (DAG) G över lövmängden X sådan att R realiseras av G enligt definitionen i Lindeberg et al. [1]. En "least common ancestor" (LCA) är den nod till två löv i en DAG sådan att ingen äkta ättling till denna nod också är "ancestor" till de två löven. R kan tolkas som en samling LCA-villkor, och G realiserar R om den kan upprätthålla alla dessa villkor. År 2025 karakteriserade Lindeberg et al. de villkor under vilka R är realiserbar och introducerade Algoritm 1, som avgör realiserbarhet och, i bekräftande fall, konstruerar den kanoniska DAGen G_R och det kanoniska nätverket N_R.

I denna uppsats studerar vi sambandet mellan klustersystemen hos fylogenetiska träd och de kanoniska nätverk som produceras av Lindeberg et al.:s ramverk för realiserbarhet [1]. Givet ett eller flera fylogenetiska träd över en gemensam lövmängd X introducerar vi algoritmen ExtractR, som extraherar lca-jämförelserelationen $\trianglelefteq_{T_i}$

från varje träd T_i och konsoliderar dessa till en enda relation $R = \bigcup_i \trianglelefteq_{T_i}$. Denna relation skickas vidare till Algoritm 1 hos Lindeberg et al., som antingen verifierar att R är realiserbar och returnerar G_R och N_R, eller rapporterar ett falsifierande villkor.

Det centrala bidraget i denna uppsats är ett bevis av ekvivalensen

$$\mathscr{C}(T) \subseteq \mathscr{C}(T') \iff T \simeq N_R,$$

för godtyckliga fylogenetiska träd T och T' över X, där N_R är det kanoniska nätverket konstruerat från $R = \trianglelefteq_T \cup \trianglelefteq_{T'}$. Beviset bygger på två hjälplemman: det första karakteriserar klusterinklusion i termer av lca-jämförelserelationerna, och det andra karakteriserar isomorfi med N_R i termer av slutningen R^+. Vi utvidgar detta resultat till samlingar av n träd och härleder ett korollarium för stjärnträdet. Innan beviset presenteras verifieras ekvivalensen beräkningsmässigt för lövmängdsstorlekar $|X| = 2$ till 10, med 100 slumpmässigt genererade trädpar per storlek; inga motexempel påträffades.

Nyckelord: DAG, nätverk, fylogenetiska träd, klustersystem, realiserbarhet

1 Introduction

In a rooted tree, the least common ancestor (lca) of two vertices x and y, denoted $lca(x, y)$, is the vertex that is an ancestor of both x and y; such that none of its proper descendants are also ancestors of both. In the primary source for this paper, [1] Lindeberg et al. expand on previous work established on the topic of *lca* and their usage in phylogenetic trees. Notably, they establish the realizability of *lca* constraints on a directed acyclic graph (DAG), meaning that a DAG can be generated while under all *lca*-constraints given. In particular, Lindeberg et al. formalize the $\trianglelefteq_G$ which is a relationship that is always realizable in a DAG G.

In this thesis, we seek to expand on the ideas in Lindeberg et al. by examining the effects of extracting all $\trianglelefteq$ relations from multiple trees and examining the DAG that can be generated while maintaining all these conditions. To support these efforts we establish algorithmic tools for extracting $\trianglelefteq$ relations from any given phylogenetic tree using *lca* relations defined in Hellmuth et al. [2] and Lindeberg et al. [1]. After extracting such relations from one or more trees, we consolidate them into a union $R = \bigcup \trianglelefteq_i$ and apply Algorithm 1 from Lindeberg et al. [1] to determine whether R is realizable; if so, the algorithm produces a Canonical DAG and a Canonical Network. Through this algorithmic pipeline we can make several interesting observations, notably how the structure of the Canonical Network is determined by conditions imposed on the trees used as input. In addition, we can use our algorithms to test assumptions on a large set of randomly generated trees in order to establish confidence before proof. We will use the pipeline to test one such assumption, verify that the assumption holds, and subsequently provide a proper proof.

The outline of the paper is as follows. After introducing basic definitions and terminology in the next section, Section 3 presents the full algorithmic pipeline, divided into three subsections. The first of the subsections focuses on tree generation

and selection, in particular the usage of the [3] TRALDA package for random tree generation. The second subsection delves into the EXTRACTR algorithm, which is an algorithm created for this paper whose purpose is to extract the $\trianglelefteq$-relations from a given tree. Finally in the last subsection covering our pipeline, we examine the utilization of ALGORITHM 1 using the output from EXTRACTR, notably using the output of ALGORITHM 1 in comparison with the trees used as input in EXTRACTR. To put our pipeline in action, Section 3 ends with a large worked example in which we test the condition $\mathscr{C}(T) \subseteq \mathscr{C}(T') \iff T \simeq N_R$ on an extensive set of randomly generated tree pairs. In Section 4, we follow up on the results from Section 3 and rigorously prove that this condition holds in general and expand on the result with several corollaries. In Section 5, our penultimate section, we discuss potential improvements and new applications of the algorithmic pipeline from Section 3, and outline potential directions for future research, in particular regarding generalizations of the proof from Section 4.

2 Basics

The following terminology is used throughout. Definitions follow Hellmuth et al. [2] and Lindeberg et al. [1] unless otherwise noted.

Sets and relations. Throughout, X denotes a finite non-empty set called the *leaf set*. The *powerset* of X, written $\mathcal{P}(X)$, is the collection of all subsets of X. A *set system* on X is any subset of $\mathcal{P}(X)$. We make frequent use of

$$\mathcal{P}_2(X) := \{\{a, b\} \mid a, b \in X\},$$

where $a = b$ is permitted, so $\mathcal{P}_2(X)$ consists of all one- and two-element subsets of X. Elements are written ab for $\{a, b\}$ and aa for the singleton $\{a\}$; since sets are unordered, $ab = ba$ always holds.

A *binary relation* R on a set A is a subset $R \subseteq A \times A$, written $a\ R\ b$ rather than $(a, b) \in R$. A sequence $a_0\ R\ a_1\ R \cdots R\ a_k$ is called an (a_0, a_k)-*chain* in R. The *support* of R is

$$\mathrm{supp}_R := \{p \in A \mid \text{there exists } q \in A \text{ with } p\ R\ q \text{ or } q\ R\ p\}.$$

When $A = \mathcal{P}_2(X)$, the *extended support* is $\mathrm{supp}_R^+ := \mathrm{supp}_R \cup \{xx \mid x \in X\}$, adding all singletons [1, Sec. 2].

Transitive closure. The *transitive closure* $\mathrm{tc}(R)$ of a relation R on A is defined by $(p, q) \in \mathrm{tc}(R)$ if and only if there is a (p, q)-chain in R; equivalently, it is the smallest transitive relation on A containing R.

Graphs and DAGs. A *directed graph* $G = (V, E)$ consists of a finite non-empty *vertex set* V and an *arc set* $E \subseteq V \times V$. We write $u \to v$ for an arc $(u, v) \in E$. For $v \in V$, its *outdegree* and *indegree* are

$$\mathrm{outdeg}_G(v) := |\{u : (v, u) \in E\}|, \qquad \mathrm{indeg}_G(v) := |\{u : (u, v) \in E\}|.$$

A vertex with $\mathrm{indeg}_G(v) = 0$ is a *root*, with $\mathrm{outdeg}_G(v) = 0$ a *leaf*, and with $\mathrm{indeg}_G(v) \geq 2$ a *hybrid* [2, Def. 3.1]. If $(u, v) \in E$, then u is a *parent* of v and v is a *child* of u.

A directed graph G is a *directed acyclic graph* (DAG) if it contains no directed cycle. In a DAG G, we write $v \preceq_G w$ if there is a directed path from w to v (i.e. v is a *descendant* of w and w an *ancestor* of v), and $v \prec_G w$ if additionally $v \neq w$. Two vertices are $\preceq_G$*-comparable* if one is an ancestor of the other, and $\preceq_G$*-incomparable* otherwise. A DAG whose leaf set is exactly X is called a *DAG on* X.

An arc (u, w) in a DAG is a *shortcut* if there exists $v \in \mathrm{child}_G(u) \setminus \{w\}$ with $w \prec_G v$; equivalently, there is a directed uw-path that does not consist solely of the arc (u, w). A DAG without shortcuts is *shortcut-free* [2, Sec. 2].

Isomorphisms. An *isomorphism* between two directed graphs G and H on X is a bijection $\varphi\colon V(G) \to V(H)$ such that $(u, v) \in E(G)$ if and only if $(\varphi(u), \varphi(v)) \in E(H)$, and such that $\varphi(x) = x$ for all $x \in X$. If such a bijection exists, G and H are *isomorphic*, written $G \simeq H$.

Networks. A *network* is a DAG on X with a unique root from which every other vertex is reachable by a directed path [2, Def. 3.1]. A network is *phylogenetic* if it contains no vertex v with $\mathrm{outdeg}_G(v) = 1$ and $\mathrm{indeg}_G(v) \leq 1$ simultaneously [2, Def. 3.1]. A *phylogenetic tree* is a phylogenetic network with no hybrid vertices; in a phylogenetic tree, $\mathrm{lca}_T(A)$ is well-defined for every non-empty $A \subseteq X$. A *star tree* on X is the phylogenetic tree consisting of a single root ρ with arcs $\rho \to x$ for every $x \in X$.

Clusters. For a network T on X and $v \in V(T)$, the *cluster* of v in T is $\mathscr{C}_T(v) := \{x \in X \mid x \preceq_T v\}$ [2, Def. 3.4]. The *clustering system* of T is

$$\mathscr{C}(T) := \{\mathscr{C}_T(v) \mid v \in V(T)\};$$

by definition every clustering system on X contains all singletons $\{x\}$ for $x \in X$ and X itself [2, Def. 3.5].

Hasse Diagrams and Regularity. The Hasse diagram $\mathcal{H}(\mathscr{C})$ of a clustering system $\mathscr{C}$ is the DAG with vertex set $\mathcal{C}$ and an arc from A to B whenever $B \subsetneq A$ and no $C \in \mathscr{C}$ satisfies $B \subsetneq C \subsetneq A$ [2, Def. 3.41]. A *regular* network per definition [2, Def.3.41] is a network that is isomorphic to a Hasse diagram of some cluster system.

Regularity & Isomorphism. A network N is *regular* if it is shortcut-free and

$$\mathcal{C}_N(u) \subseteq \mathcal{C}_N(v) \iff u \preceq_N v \quad \text{for all } u, v \in V(N)$$

[2, Prop. 3.43]. In particular, every phylogenetic tree is regular, and two regular networks on the same leaf set are isomorphic if and only if they have identical clustering systems [2, Prop. 3.42].

Phylogenetic tree isomorphic to Hasse. Every phylogenetic tree T satisfies $T \simeq \mathcal{H}(\mathscr{C}(T))$ [2, Cor. 3.47], and every regular network N satisfies $N \simeq \mathcal{H}(\mathscr{C}(N))$ [2, Prop. 4.10]. In both cases the isomorphism is the identity on X.

Least common ancestors. For a DAG G on X and $A \subseteq X$, a vertex v is a *common ancestor* of A if every element of A is a descendant of v. A vertex v is a *least common*

ancestor (LCA) of A if it is $\preceq_G$-minimal among all common ancestors; the set of all LCAs of A is $\mathrm{LCA}_G(A)$. If $|\mathrm{LCA}_G(A)| = 1$ we write $\mathrm{lca}_G(A) = v$ and say the LCA is *well-defined* [2, Def. 6.1]. A DAG G is 2-*lca-relevant* if for every $v \in V(G)$ there exist $x, y \in X$ (with $x = y$ allowed) such that $v = \mathrm{lca}_G(xy)$; phylogenetic trees always have this property [1, Obs. 7.6].

The lca-comparison relations. For a DAG G on X, we define two relations on $\mathcal{P}_2(X)$ [1, Def. 2.5]:

$$\blacktriangleleft_G := \{(ab, cd) \mid \mathrm{lca}_G(ab),\ \mathrm{lca}_G(cd) \text{ are well-defined and } \mathrm{lca}_G(ab) \prec_G \mathrm{lca}_G(cd)\},$$

$$\trianglelefteq_G := \{(ab, cd) \mid \mathrm{lca}_G(ab),\ \mathrm{lca}_G(cd) \text{ are well-defined and } \mathrm{lca}_G(ab) \preceq_G \mathrm{lca}_G(cd)\}.$$

By definition, $\blacktriangleleft_G \subseteq \trianglelefteq_G$. The relation $\trianglelefteq_T$ on a phylogenetic tree T encodes the partial order on pairwise LCAs in T induced by ancestry, and is the primary object from which the canonical network is built.

Realizability. A relation R on $\mathcal{P}_2(X)$ is *realizable* if there exists a DAG G on X such that for all $ab, cd \in \mathrm{supp}_R^+$ the vertices $\mathrm{lca}_G(ab)$ and $\mathrm{lca}_G(cd)$ are well-defined and the following two implications hold:

(I1) $(ab, cd) \in \mathrm{tc}(R)$ and $(cd, ab) \notin \mathrm{tc}(R)$ implies that $\mathrm{lca}_G(ab) \prec_G \mathrm{lca}_G(cd)$

(I2) $(ab, cd) \in \mathrm{tc}(R)$ and $(cd, ab) \in \mathrm{tc}(R)$ implies that $\mathrm{lca}_G(ab) = \mathrm{lca}_G(cd)$

When **(I1)** and **(I2)** hold, we say that R is *realized* by G.

Phylogenetic Tree Realization. A phylogenetic tree T always realizes its own lca-comparison relation $\trianglelefteq_T$ [1, Lem. 3.5].

+-Closure R^+ The *+-closure* R^+ of a relation R is the reflexive, transitive, cross-consistent closure of R defined in [1, Def. 4.3]; it captures all lca-ordering constraints implied by R.

R **Conditions**. Per [1, Thrm.5.10] Lindeberg et al, R is realizable if and only if it satisfies:

- **(X1):** For all $a, b, x \in X : ab \neq xx$ *implies* $(ab, xx) \notin R^+$.
- **(X2):** For all $a, b, x, y \in X$: $(ab, xy) \in tc(R)$ and $(xy, ab) \notin tc(R)$ *implies* $(xy, ab) \notin R^+$.

The Class $[ab]$**.** Let R be a realizable relation on $\mathcal{P}_2(X)$. The relation $\sim_{R^+}$ on supp_R^+ defined by

$$ab \sim_{R^+} cd \quad :\Longleftrightarrow \quad ab\ R^+\ cd \ \text{ and } \ cd\ R^+\ ab$$

is an equivalence relation; it identifies precisely those pairs whose LCAs must coincide in any DAG realizing R. For $ab \in \mathrm{supp}_R^+$ we write $[ab]$ for its $\sim_{R^+}$-class. These classes are the vertices of the canonical network, and the vertex $[ab]$ is exactly the least common ancestor $\mathrm{lca}_{N_R}(ab)$ for all $ab \in \mathrm{supp}_R^+$ with $a \neq b$.

Canonical Network. If R is realizable then the Canonical Network N_R realizes R. The Canonical Network is the phylogenetic regular network N on X with $\trianglelefteq_N = R^+$ [1, prop.6.4].

Ancestor relation in N_R. The ancestor relation in N_R is completely determined by the $+$-closure R^+: for all $ab, cd \in \mathrm{supp}_R^+$,

$$ab\ R^+\ cd \iff \mathrm{lca}_{N_R}(ab) \preceq_{N_R} \mathrm{lca}_{N_R}(cd)$$

[1, Lem. 7.2(2a)].

Canonical Networks isomorphic to Hasse. Per the definition of regularity and since N_R is regular, $N_R \simeq \mathcal{H}(\mathscr{C}(N_R))$.

3 ExtractR and Tree-DAG comparison

In this section, we present an algorithmic pipeline, beginning with the selection of one or more phylogenetic trees, either randomly generated or manually chosen. We then examine the ExtractR algorithm created for this paper. Next, we describe how the output of ExtractR is fed into Algorithm 1, established by Lindeberg et al. [1], which either rejects the relation as non-realizable or returns the canonical DAG and canonical network realizing it. Finally, at the end of this section, we apply the full pipeline to test the equivalence

$$\mathscr{C}(T) \subseteq \mathscr{C}(T') \iff T \simeq N_R.$$

For readability, the pseudocode for the pipeline is collected in an appendix.

3.1 Tree Generation/Selection

The first stage of the pipeline concerns the selection of one or more phylogenetic trees to serve as input. Trees can either be constructed by hand and supplied directly, or generated at random. For the experimental work in this thesis, we rely on random generation via the `tralda` library [3], a Python package for working with phylogenetic trees and related data structures. Specifically, we use `Tree.random_tree(L)`, which produces a random rooted phylogenetic tree on L leaves.

Our implementation supports two generation strategies. The first, used for the experiment at the end of this section and matching the assumptions of most existing literature, is *equal-leaf-set generation*: all N trees in a batch are drawn independently with exactly the same leaf set of size L. This ensures that the relation $\trianglelefteq_{T_i}$ extracted from each tree is defined over a common ground set X, which is a prerequisite for taking their union and applying Algorithm 1. The second strategy, *variable-leaf-set generation*, produces a *principal* tree on L leaves and fills the remaining $N-1$ slots with trees whose leaf sets are randomly chosen subsets of X of size between 2 and L; this mode is intended for future experiments in which the input trees need not share a common leaf set.

In either case, certain degenerate tree shapes may optionally be excluded. In particular, the *star tree* on X may be excluded. Since every pair of distinct leaves has the root as its lca in a star tree, the extracted relation $\trianglelefteq_T$ contains no strict inequalities and so imposes no nontrivial ordering constraints. Likewise, duplicate trees may also be excluded, so that each tree in a batch contributes genuinely new lca-constraints to the union R.

Once a batch of trees has been generated, we apply a *leaf standardization* step before any further processing. Because trees generated by `tralda` use independently assigned leaf identifiers, trees generated separately will in general label their leaves inconsistently: the leaf labelled `3` in one tree need not correspond to the leaf labelled `3` in another. The standardization procedure reassigns the integer labels $0, 1, \ldots, L-1$ to the leaves of each tree in a consistent order, and assigns string labels $v_0, v_1, \ldots$ to internal nodes, with the root receiving the label `root`. After this step, leaf i in every tree refers to the same element of the ground set X, making it meaningful to compare lca-values across trees and to consolidate the extracted relations into a single relation $R = \bigcup_i \trianglelefteq_{T_i}$. The extraction of these relations from the standardized trees is the subject of the following subsection.

3.2 ExtractR

With a standardized collection of trees in hand, the next stage of the pipeline is to extract the lca-comparison relation $\trianglelefteq_T$ from each tree T and consolidate the results into a single relation $R = \bigcup_i \trianglelefteq_{T_i}$ over the shared ground set X. Recall from Section 2 that for a phylogenetic tree T on X, the relation $\trianglelefteq_T$ is defined by

$$\trianglelefteq_T := \big\{(ab,\, cd) \mid \mathrm{lca}_T(ab),\, \mathrm{lca}_T(cd) \text{ are well-defined and } \mathrm{lca}_T(ab) \preceq_T \mathrm{lca}_T(cd)\big\},$$

and similarly the strict relation $\triangleleft_T$ replaces $\preceq_T$ with $\prec_T$. Since T is a phylogenetic tree, $\mathrm{lca}_T(ab)$ is well-defined for every $ab \in \mathcal{P}_2(X)$, so $\mathrm{supp}^+_{\trianglelefteq_T} = \mathcal{P}_2(X)$ holds automatically [1].

Algorithm Description

The ExtractR algorithm proceeds in two steps. In the first step, it enumerates all pairwise lca-values: for every pair $i \leq j$ with $i, j \in X$, it computes $\mathrm{lca}_T(i, j)$ using the *get* method from the *LCA* class [3] from `tralda` and stores the result as an *lca-pair* $(i, j, \mathrm{lca}_T(i, j))$. This yields a list of $\binom{|X|}{2} + |X|$ lca-pairs covering all elements of $\mathcal{P}_2(X)$.

In the second step, the algorithm compares every pair of lca-pairs to determine the relative ancestor ordering of their lca-values. For two lca-pairs $\alpha = (a_1, b_1, u)$ and $\beta = (a_2, b_2, v)$, four outcomes are possible depending on whether $u \preceq_T v$, $v \preceq_T u$, both (i.e. $u = v$), or neither:

- If $u \prec_T v$, then $(a_1 b_1) \trianglelefteq_T (a_2 b_2)$ is added to R.
- If $v \prec_T u$, then $(a_2 b_2) \trianglelefteq_T (a_1 b_1)$ is added to R.

- If $u = v$, both constraints are added, reflecting that $u \preceq_T v$ and $v \preceq_T u$ hold simultaneously.
- If u and v are $\preceq_T$-incomparable, no constraint is added.

Deduplication and Output

After ExtractR has been applied to each tree T_i individually, we are given a list of constraints for each such tree, which are concatenated into a single list. This list is then passed through a *deduplication* step, which discards any row (a, b, c, d) that has already appeared, preserving the order of first occurrence. This is not a necessary step for the algorithmic pipeline to work, but it makes reading the list of constraints far easier. The deduplicated list represents the relation $R = \bigcup_i \trianglelefteq_{T_i}$ without redundancy.

The consolidated relation is then written to a CSV file whose first row lists the elements of X and each subsequent row encodes a single constraint $ab \trianglelefteq cd$ as four comma-separated values a, b, c, d. This file serves as the input format expected by Algorithm 1 [1], to which we turn in the following subsection.

3.3 From ExtractR to Algorithm 1

With the consolidated relation $R = \bigcup_i \trianglelefteq_{T_i}$ written to a CSV file, the final stage of the pipeline passes R to Algorithm 1 of Lindeberg et al. [1], which determines whether R is realisable and, if so, constructs the canonical DAG G_R and canonical network N_R from R. We briefly recall the steps of Algorithm 1 before describing how its output is used to compare N_R against the input trees.

Algorithm 1 Overview

Algorithm 1 takes as input the ground set X and the relation R on $\mathcal{P}_2(X)$, and proceeds as follows. First, it enforces a consistent representation of each set $\{a, b\} = \{b, a\}$ by canonicalising all pairs. It then computes the extended support supp_R^+ and the reflexive, transitive, cross-consistent closure R^+ of R as defined in [1]. It then checks if the R conditions $X1$ and $X2$ defined in Section 2 hold.

If either condition is violated, Algorithm 1 returns False together with an identifying label (`X1` or `X2`) and the falsifying constraint, and the pipeline reports that R is not realisable. In practice, since each $\trianglelefteq_{T_i}$ is itself always realisable [1], non-realizability can only arise when the union of relations from two or more trees is inconsistent.

If both conditions hold, Algorithm 1 proceeds to constructs the *canonical DAG* G_R and the *canonical network* N_R. By [1], N_R is regular and phylogenetic, with $[ab] = \mathrm{lca}_{N_R}(ab)$ for all $ab \in \mathrm{supp}_R^+$ with $a \neq b$.

Comparing N_R to the Input Trees

Once G_R and N_R have been constructed, the pipeline checks whether N_R is

isomorphic to any of the input trees $T_1, \ldots, T_n$. Recall from Section 2 that two networks N_1 and N_2 on the same leaf set X are isomorphic, written $N_1 \simeq N_2$, if there is a graph isomorphism $\varphi : V(N_1) \to V(N_2)$ that is the identity on X.
With N_R and the input trees available, one can perform whatever other comparison we might be interested in. However, we leave as an avenue for future research since this paper is primarily focused on examining the consequences of imposed conditions on the trees rather than performing post-generated analysis on N_R.

Example: Testing $\mathscr{C}(T) \subseteq \mathscr{C}(T') \iff T \simeq N_R$

To verify the condition $\mathcal{C}(T) \subseteq \mathcal{C}(T') \iff T \simeq N_R$ computationally, we run the following pipeline for each leaf set size $|X| = \ell$, starting at a chosen minimum $\ell_{\min} \geq 2$ and incrementing by one until a maximum $\ell_{\max}$ is reached. For each value of ℓ, the pipeline is repeated a fixed number of times.

Step 1 (Tree generation). Generate two random phylogenetic trees T and T' on a common leaf set X with $|X| = \ell$. We then verify that $\mathscr{C}(T) \subseteq \mathscr{C}(T')$; if this condition does not hold, we discard both trees and generate a new pair. This step repeats until a valid pair is found.

Step 2 (Extracting R). We apply ExtractR to both T and T' independently, obtaining the relations $\trianglelefteq_T$ and $\trianglelefteq_{T'}$. These are concatenated and deduplicated to form the union

$$R = \trianglelefteq_T \cup \trianglelefteq_{T'} .$$

Step 3 (Applying Algorithm 1). The consolidated relation R is passed to Algorithm 1 of Lindeberg et al. [1], which constructs the canonical DAG G_R and canonical network N_R. Since $\mathcal{C}(T) \subseteq \mathcal{C}(T')$ is enforced by construction in Step 1, Algorithm 1 is expected to return a valid network in every run; a non-realisable result would indicate an error in the pipeline.

Step 4 (Comparison). We check whether $N_R \simeq T$. The run is recorded as a PASS if $N_R \simeq T$ and as a FAIL otherwise.

Steps 1–4 are repeated for a fixed number of runs at each leaf set size, after which ℓ is incremented by one and the process restarts.

We arbitrarily test this pipeline for ℓ_{min}=2 and $\ell_{max} = 10$, repeated 100 times for each leaf set size.

Results. For every single run in which $\mathscr{C}(T) \subseteq \mathscr{C}(T')$ holds, the runs pass. With this, we can say with some confidence that the condition will hold for all phylogenetic trees T, T'. However, we still need to verify this with a proper proof as found in the next section.

4 Clustering systems of phylogenetic trees restricting and restricted by Canonical Network.

Our goal in this section is to prove that $\mathscr{C}(T) \subseteq \mathscr{C}(T') \iff T \simeq \mathrm{N}_R$ holds for any two phylogenetic trees over a leaf set X. In the previous section, this condition was verified computationally for hundreds of randomly generated tree pairs with $|X|$ ranging from 2 to 10. We now establish it in full generality via two auxiliary lemmas. The first shows that $\mathscr{C}(T) \subseteq \mathscr{C}(T')$ if and only if $\trianglelefteq_{T'} \subseteq \trianglelefteq_T$. The second shows that $T \simeq \mathrm{N}_R$ if and only if $R^+ = \trianglelefteq_T$, where R^+ denotes the +-closure of $R = \trianglelefteq_T \cup \trianglelefteq_{T'}$ as defined in [1, Def. 4.3]. Throughout this section, isomorphism between two phylogenetic networks on the common leaf set X is taken to be leaf-labelled, meaning that the underlying bijection φ restricts to the identity on X. Both lemmas draw on results from Hellmuth et al. [2] and Lindeberg et al. [1].

Lemma 4.1. *Let T and T' be phylogenetic trees over a non-empty leaf set X. Then $\mathscr{C}(T) \subseteq \mathscr{C}(T')$ if and only if $\trianglelefteq_{T'} \subseteq \trianglelefteq_T$.*

Proof. We make use of the characterisation of regular networks defined in Section 2: a network N is regular if and only if $\mathcal{C}(u) \subseteq \mathcal{C}(v) \iff u \preceq_N v$ for all $u, v \in V(N)$. Since phylogenetic trees are regular, this equivalence applies to both T and T' throughout the proof.

($\Rightarrow$) Assume $\mathscr{C}(T) \subseteq \mathscr{C}(T')$. Let $(ab, cd) \in \trianglelefteq_{T'}$, so that $\mathrm{lca}_{T'}(ab) \preceq_{T'} \mathrm{lca}_{T'}(cd)$. By regularity of T', this ancestor relation is equivalent to cluster containment: $\mathcal{C}_{T'}(\mathrm{lca}_{T'}(ab)) \subseteq \mathcal{C}_{T'}(\mathrm{lca}_{T'}(cd))$.

Since $\mathscr{C}(T) \subseteq \mathscr{C}(T')$, the cluster $\mathcal{C}_T(\mathrm{lca}_T(cd)) \in \mathscr{C}(T)$ also appears in $\mathscr{C}(T')$, say as $\mathcal{C}_{T'}(w)$ for some $w \in V(T')$. Both c and d belong to this cluster, so w is a common ancestor of c and d in T'. By minimality of $\mathrm{lca}_{T'}(cd)$ we have $\mathrm{lca}_{T'}(cd) \preceq_{T'} w$, and by regularity of T' this gives $\mathcal{C}_{T'}(\mathrm{lca}_{T'}(cd)) \subseteq \mathcal{C}_{T'}(w) = \mathcal{C}_T(\mathrm{lca}_T(cd))$.

Combining, both a and b lie in

$$\mathcal{C}_{T'}(\mathrm{lca}_{T'}(ab)) \subseteq \mathcal{C}_{T'}(\mathrm{lca}_{T'}(cd)) \subseteq \mathcal{C}_T(\mathrm{lca}_T(cd)),$$

so $\mathrm{lca}_T(cd)$ is a common ancestor of a and b in T. By minimality of $\mathrm{lca}_T(ab)$ we get $\mathrm{lca}_T(ab) \preceq_T \mathrm{lca}_T(cd)$, so $(ab, cd) \in \trianglelefteq_T$. Since (ab, cd) was arbitrary, $\trianglelefteq_{T'} \subseteq \trianglelefteq_T$.

($\Leftarrow$) Assume $\trianglelefteq_{T'} \subseteq \trianglelefteq_T$. Let $\mathcal{C}_T(v) \in \mathscr{C}(T)$ be an arbitrary cluster of T. If $\mathcal{C}_T(v)$ is a singleton $\{x\}$ or equals all of X, it trivially belongs to $\mathscr{C}(T')$, since every clustering system contains all singletons and X itself. We therefore consider the case where $\mathcal{C}_T(v)$ is a proper non-singleton subset of X.

Let $v' = \mathrm{lca}_{T'}(\mathcal{C}_T(v))$. Since phylogenetic trees are strong lca-networks [2, Lem. 7.11] and $|\mathcal{C}_T(v)| > 1$, there exist leaves $a, b \in \mathcal{C}_T(v)$ with $\mathrm{lca}_{T'}(\{a, b\}) = v'$. Every leaf of $\mathcal{C}_T(v)$ is a descendant of v' in T', giving the containment $\mathcal{C}_T(v) \subseteq \mathcal{C}_{T'}(v')$.

Suppose for contradiction that this is a strict containment, so there exists $z \in \mathcal{C}_{T'}(v') \setminus \mathcal{C}_T(v)$. Since both a and z are descendants of v' in T', and $v' = \mathrm{lca}_{T'}(a, b)$, we have $\mathrm{lca}_{T'}(a, z) \preceq_{T'} \mathrm{lca}_{T'}(a, b)$, so $(az, ab) \in \trianglelefteq_{T'}$. By assumption, $(az, ab) \in \trianglelefteq_T$, meaning $\mathrm{lca}_T(a, z) \preceq_T \mathrm{lca}_T(a, b)$.

Since $a, b \in \mathcal{C}_T(v)$, both are descendants of v, so $\mathrm{lca}_T(a,b) \preceq_T v$. Now $\mathrm{lca}_T(a,z)$ and v are both ancestors of a in T, and in a tree the ancestors of any fixed vertex form a totally $\preceq_T$-ordered chain. Hence either $\mathrm{lca}_T(a,z) \preceq_T v$ or $v \prec_T \mathrm{lca}_T(a,z)$. In the first case, $z \preceq_T \mathrm{lca}_T(a,z) \preceq_T v$ would give $z \in \mathcal{C}_T(v)$, contradicting $z \notin \mathcal{C}_T(v)$. Therefore $v \prec_T \mathrm{lca}_T(a,z)$. Combining with $\mathrm{lca}_T(a,b) \preceq_T v$, we obtain $\mathrm{lca}_T(a,b) \prec_T \mathrm{lca}_T(a,z)$, contradicting $\mathrm{lca}_T(a,z) \preceq_T \mathrm{lca}_T(a,b)$. Therefore $\mathcal{C}_T(v) = \mathcal{C}_{T'}(v')$, so $\mathcal{C}_T(v) \in \mathscr{C}(T')$.

Since $\mathcal{C}_T(v)$ was arbitrary, $\mathscr{C}(T) \subseteq \mathscr{C}(T')$. □

Lemma 4.2. *Let T and T' be phylogenetic trees over a non-empty leaf set X, let $R = \trianglelefteq_T \cup \trianglelefteq_{T'}$, and let N_R be the canonical network built from R. Then $T \simeq \mathrm{N}_R$ if and only if $R^+ = \trianglelefteq_T$.*

Proof. Since T and T' are phylogenetic trees on X, $\mathrm{lca}_T(ab)$ and $\mathrm{lca}_{T'}(ab)$ are well-defined for every $ab \in \mathcal{P}_2(X)$. Consequently, $\mathrm{supp}_{\trianglelefteq_T} = \mathrm{supp}_{\trianglelefteq_{T'}} = \mathcal{P}_2(X)$, and so $\mathrm{supp}_R^+ = \mathcal{P}_2(X)$.

($\Leftarrow$) Assume $R^+ = \trianglelefteq_T$. Since [1, prop.4.4], $R \subseteq R^+ = \trianglelefteq_T$ and $\trianglelefteq_T \subseteq R$ by construction of R, we have $R = \trianglelefteq_T$. By [1, Lem. 3.5], T realises $\trianglelefteq_T = R$, so R is realisable and N_R is well-defined [1, Thm. 5.10]. By [1, Prop. 6.4], N_R is regular and phylogenetic, with $[ab] = \mathrm{lca}_{N_R}(ab)$ for all $ab \in \mathrm{supp}_R^+$ with $a \neq b$, and $x = \mathrm{lca}_{N_R}(xx)$ for all $x \in X$. Furthermore, by [1, Lem. 7.2(2a)], the ancestor relation in N_R is completely determined by R^+: for all $ab, cd \in \mathrm{supp}_R^+$,

$$ab \; R^+ \; cd \iff \mathrm{lca}_{N_R}(ab) \preceq_{N_R} \mathrm{lca}_{N_R}(cd). \qquad (*)$$

We show $\mathscr{C}(T) = \mathscr{C}(\mathrm{N}_R)$. For any internal vertex $[ab]$ of N_R, a leaf $x \in X$ belongs to $\mathcal{C}_{N_R}([ab])$ if and only if $\mathrm{lca}_{N_R}(xx) \preceq_{N_R} \mathrm{lca}_{N_R}(ab)$, which by $(*)$ holds if and only if $xx \; R^+ \; ab$. Since $R^+ = \trianglelefteq_T$, this is equivalent to $x \preceq_T \mathrm{lca}_T(ab)$, i.e. $x \in \mathcal{C}_T(\mathrm{lca}_T(ab))$. Hence $\mathcal{C}_{N_R}([ab]) = \mathcal{C}_T(\mathrm{lca}_T(ab))$, so every cluster of N_R is a cluster of T. Conversely, since phylogenetic trees are 2-lca-relevant [1, Obs. 7.6], every internal vertex $v \in V(T)$ equals $\mathrm{lca}_T(ab)$ for some pair ab, so the identity above gives $\mathcal{C}_T(v) = \mathcal{C}_{N_R}([ab])$, meaning every cluster of T appears in N_R. The singleton clusters $\{x\}$ for $x \in X$ and the cluster X itself belong to every clustering system on X by definition. Together this gives $\mathscr{C}(T) = \mathscr{C}(\mathrm{N}_R)$.

Since T is a phylogenetic tree, its clustering system is a hierarchy and T is isomorphic to the Hasse diagram of $\mathscr{C}(T)$ [2, Cor. 3.47]. Since N_R is regular, it is likewise isomorphic to the Hasse diagram of $\mathscr{C}(\mathrm{N}_R)$ [2, Prop. 4.10]. Since $\mathscr{C}(T) = \mathscr{C}(\mathrm{N}_R)$, we obtain

$$T \;\simeq\; \mathcal{H}(\mathscr{C}(T)) \;=\; \mathcal{H}(\mathscr{C}(\mathrm{N}_R)) \;\simeq\; \mathrm{N}_R,$$

where each isomorphism is the identity on X.

($\Rightarrow$) Assume $T \simeq \mathrm{N}_R$ as defined in Section 2, $\varphi : V(T) \to V(\mathrm{N}_R)$, so φ is the identity on X. Since φ preserves the ancestor relation and is the identity on leaves, for any $a, b \in X$ the vertex $\varphi(\mathrm{lca}_T(ab))$ is a $\preceq_{\mathrm{N}_R}$-minimal common ancestor of a and b in N_R. By Proposition 6.4 of [1], $\mathrm{lca}_{\mathrm{N}_R}(ab)$ is well-defined, so $\varphi(\mathrm{lca}_T(ab)) = \mathrm{lca}_{\mathrm{N}_R}(ab)$ for

every $ab \in \mathcal{P}_2(X)$. For any $ab, cd \in \mathcal{P}_2(X) = \mathrm{supp}_R^+$, applying $(*)$ and then the two properties of φ in succession gives:

$$\begin{aligned} ab\ R^+\ cd &\iff \mathrm{lca}_{N_R}(ab) \preceq_{N_R} \mathrm{lca}_{N_R}(cd) \\ &\iff \varphi(\mathrm{lca}_T(ab)) \preceq_{N_R} \varphi(\mathrm{lca}_T(cd)) \\ &\iff \mathrm{lca}_T(ab) \preceq_T \mathrm{lca}_T(cd), \end{aligned}$$

which is precisely $ab \trianglelefteq_T cd$. Since ab and cd were arbitrary, $R^+ = \trianglelefteq_T$. □

Theorem 4.3. *Let T and T' be phylogenetic trees over a non-empty leaf set X, let $R = \trianglelefteq_T \cup \trianglelefteq_{T'}$, and let N_R be the canonical network built from R. Then*

$$\mathscr{C}(T) \subseteq \mathscr{C}(T') \iff T \simeq \mathrm{N}_R.$$

Proof. $(\Rightarrow)$ If $\mathscr{C}(T) \subseteq \mathscr{C}(T')$, then Lemma 4.1 gives $\trianglelefteq_{T'} \subseteq \trianglelefteq_T$, so $R = \trianglelefteq_T \cup \trianglelefteq_{T'} = \trianglelefteq_T$. The relation $\trianglelefteq_T$ is its own $+$-closure [1, Prop. 4.10], so $R^+ = \trianglelefteq_T$, and Lemma 4.2 gives $T \simeq \mathrm{N}_R$.

$(\Leftarrow)$ If $T \simeq \mathrm{N}_R$, then Lemma 4.2 gives $R^+ = \trianglelefteq_T$. Since $R \subseteq R^+ = \trianglelefteq_T$ and $\trianglelefteq_T \subseteq R$ by construction of R, we have $R = \trianglelefteq_T$, so $\trianglelefteq_{T'} \subseteq \trianglelefteq_T$. Lemma 4.1 then gives $\mathscr{C}(T) \subseteq \mathscr{C}(T')$. □

Corollary 4.4. *Let T be an arbitrary phylogenetic tree over a non-empty leaf set X, let T' be the star tree on X, let $R = \trianglelefteq_T \cup \trianglelefteq_{T'}$, and let N_R be the canonical network built from R. Then $T' \simeq \mathrm{N}_R$.*

Proof. The clustering system of the star tree on X is $\mathscr{C}(T') = \{\{x\} : x \in X\} \cup \{X\}$. Every clustering system on X contains the singletons and X itself by definition, so $\mathscr{C}(T') \subseteq \mathscr{C}(T)$. Applying Theorem 4.3 with the roles of T and T' exchanged yields $T' \simeq \mathrm{N}_R$. □

Corollary 4.5. *Let $T_1, T_2, \ldots, T_n$ be n phylogenetic trees over a non-empty leaf set X, let $R = \bigcup_{i=1}^n \trianglelefteq_{T_i}$, and let N_R be the canonical network built from R. For each $i \in \{1, \ldots, n\}$,*

$$T_i \simeq \mathrm{N}_R \iff \mathscr{C}(T_i) \subseteq \bigcap_{\substack{j=1 \\ j \neq i}}^{n} \mathscr{C}(T_j).$$

Proof. Fix $i \in \{1, \ldots, n\}$.

$(\Rightarrow)$ Suppose $T_i \simeq \mathrm{N}_R$. By the same argument as the $(\Rightarrow)$ direction of Lemma 4.2 (whose proof uses only that $R = \bigcup_k \trianglelefteq_{T_k}$ and $\mathrm{supp}_R^+ = \mathcal{P}_2(X)$), we have $R^+ = \trianglelefteq_{T_i}$. Since $\trianglelefteq_{T_j} \subseteq R \subseteq R^+ = \trianglelefteq_{T_i}$ for every $j \neq i$, Lemma 4.1 gives $\mathscr{C}(T_i) \subseteq \mathscr{C}(T_j)$ for every $j \neq i$, hence $\mathscr{C}(T_i) \subseteq \bigcap_{j \neq i} \mathscr{C}(T_j)$.

$(\Leftarrow)$ Suppose $\mathscr{C}(T_i) \subseteq \bigcap_{j \neq i} \mathscr{C}(T_j)$. Then $\mathscr{C}(T_i) \subseteq \mathscr{C}(T_j)$ for every $j \neq i$, so Lemma 4.1 gives $\trianglelefteq_{T_j} \subseteq \trianglelefteq_{T_i}$ for every $j \neq i$. Therefore $R = \bigcup_{j=1}^n \trianglelefteq_{T_j} = \trianglelefteq_{T_i}$, and $R^+ = \trianglelefteq_{T_i}$ by [1, Prop. 4.10]. By the same argument as the $(\Leftarrow)$ direction of Lemma 4.2, $T_i \simeq \mathrm{N}_R$. □

5 Discussion and Conclusions

In this thesis, we have constructed an algorithmic pipeline for extracting lca-comparison relations from phylogenetic trees and feeding them into the canonical network construction of Lindeberg et al. [1]. The central component of this pipeline is the ExtractR algorithm, which takes a standardized phylogenetic tree T on a leaf set X and produces the full relation $\trianglelefteq_T$ by enumerating all pairwise lca-values and comparing their relative ancestor ordering. Given one or more trees, the individual relations are consolidated into a union $R = \bigcup_i \trianglelefteq_{T_i}$, which is passed to Algorithm 1 of Lindeberg et al. to produce the canonical DAG G_R and canonical network N_R.

Using this pipeline, we verified computationally that the condition

$$\mathcal{C}(T) \subseteq \mathcal{C}(T') \iff T \simeq N_R$$

holds without exception across hundreds of randomly generated tree pairs with leaf set sizes ranging from $|X| = 2$ to $|X| = 10$. We subsequently proved this equivalence in full generality in Theorem 4.3.

Directions for Future Research

Statistical Analysis on large sets of Canonical Networks. Through ExtractR or a similar algorithm, it is possible to perform hundreds or thousands of experiments on large-scale sets of generated trees. There may exist potential in finding patterns or trends that occur in such canonical networks, potentially leading to the discovery of additional conditions such as the one proven in Section 4.

Non-equal leaf sets. Theorem 4.3 and its corollaries are stated for phylogenetic trees sharing a common leaf set X. A natural generalisation is to consider trees $T_1, \ldots, T_n$ with distinct leaf sets $X_1, \ldots, X_n$. In this setting, the lca-comparison relations $\trianglelefteq_{T_i}$ are defined over different domains, and consolidating them into a single relation requires restricting to the shared ground set or working with a broader notion of compatibility. Whether an analogue of Theorem 4.3 holds in this setting is an open question.

Algorithmic complexity. The current implementation of ExtractR runs in $O(|X|^4)$ time due to the double loop over all $\binom{|X|}{2} + |X|$ lca-pairs. It is an open question whether a more efficient extraction procedure exists, particularly in light of the linear-time lca algorithms available for trees [1].

AI Usage

AI was used as a tool for the coding section of this project, primarily in adapting already written code into modular sections for ease of editing and structure. It was also used as a code checker in order to detect errors.
AI was not used to generate, write, or otherwise come up with the proofs within this project. It was used as a tool for correcting grammar, spelling, and as a tool to verify understanding of concepts explained in reference material. Finally, AI was used to check the correctness of the finished proof.

PSEUDOCODE

Algorithm 1 Equal-Leaf-Set Tree Generation

Require: Number of trees $N \geq 1$, leaf set size L, booleans AllowStar, AllowDuplicates, integer MaxAttempts

Ensure: A list $\mathcal{T}$ of N rooted phylogenetic trees, each on leaf set $X = \{0, 1, \ldots, L-1\}$

$\mathcal{T} \leftarrow []$
$seen \leftarrow \emptyset$
$attempts \leftarrow 0$
while $|\mathcal{T}| < N$ **do**
    if $attempts >$ MaxAttempts **then**
        warn only $|\mathcal{T}|$ trees could be generated
        break
    end if
    $T \leftarrow$ RandomTree(L)
    $attempts \leftarrow attempts + 1$
    if $\neg$AllowStar **and** height(T) $= 1$ **then**
        continue
    end if
    if $\neg$AllowDuplicates **then**
        $\sigma \leftarrow \mathcal{C}(T)$ ▷ Cluster system as signature
        if $\sigma \in seen$ **then**
            continue
        end if
        $seen \leftarrow seen \cup \{\sigma\}$
    end if
    $\mathcal{T}$.Append(T)
end while
return $\mathcal{T}$

Algorithm 2 Variable-Leaf-Set Tree Generation

Require: Number of trees $N \geq 1$, maximum leaf set size L, booleans AllowStar, AllowDuplicates, AllowSubgraphs, integer MaxAttempts

Ensure: A list $\mathcal{T}$ of N trees where $\mathcal{T}[0]$ is on L leaves and all others are on at most L leaves

$\mathcal{T} \leftarrow []$
$seen \leftarrow \emptyset$
$attempts \leftarrow 0$
while $|\mathcal{T}| < N$ **do**
    if $attempts >$ MaxAttempts **then**
        warn only $|\mathcal{T}|$ trees could be generated
        break
    end if
    $attempts \leftarrow attempts + 1$
    if $|\mathcal{T}| = 0$ **then**
        $\ell \leftarrow L$ ▷ First tree uses the full leaf set
    else
        $\ell \leftarrow \text{Uniform}\{2, \ldots, L\}$
    end if
    $T \leftarrow \text{RandomTree}(\ell)$
    if ¬AllowStar **and** height$(T) = 1$ **then**
        continue
    end if
    if ¬AllowDuplicates **then**
        $\sigma \leftarrow \mathcal{C}(T)$
        if $\sigma \in seen$ **then**
            continue
        end if
        $seen \leftarrow seen \cup \{\sigma\}$
    end if
    if ¬AllowSubgraphs **then**
        if $\exists T' \in \mathcal{T}$ such that T is a refinement of T' **then**
            continue
        end if
    end if
    $\mathcal{T}$.Append(T)
end while
return $\mathcal{T}$

Algorithm 3 Leaf Standardization

Require: A rooted phylogenetic tree T on L leaves
Ensure: T with leaves relabelled $0, 1, \ldots, L-1$ and internal nodes relabelled $v_0, v_1, \ldots$

$i \leftarrow 0$
for each leaf ℓ of T (in traversal order) **do**
    $\ell.label \leftarrow i$
    $i \leftarrow i + 1$
end for
$j \leftarrow 0$
for each internal node v of T (in preorder) **do**
    $v.label \leftarrow v_j$
    $j \leftarrow j + 1$
end for
$T.root.label \leftarrow$ `root`
return T

Algorithm 4 ExtractR

Require: A standardized phylogenetic tree T on leaf set $X = \{0, \ldots, L-1\}$, boolean Strict

Ensure: A list R of constraints (a, b, c, d) encoding $\trianglelefteq_T$ (or $\triangleleft_T$ if Strict)

Initialise an lca data structure Λ on T
pairs $\leftarrow []$
for $i = 0, \ldots, L-1$ **do**
    for $j = i, \ldots, L-1$ **do**
        pairs.Append$\big((i,\ j,\ \Lambda.\text{lca}(i, j))\big)$
    end for
end for
$R \leftarrow []$
for $k = 0, \ldots, |pairs| - 1$ **do**
    for $m = k, \ldots, |pairs| - 1$ **do**
        $(a_1, b_1, u) \leftarrow pairs[k]$
        $(a_2, b_2, v) \leftarrow pairs[m]$
        $\delta \leftarrow$ **nil**
        $\gamma \leftarrow$ **nil**
        if Strict **then**
            if $u \prec_T v$ **then**
                $\delta \leftarrow (a_1, b_1, a_2, b_2)$
            else if $v \prec_T u$ **then**
                $\delta \leftarrow (a_2, b_2, a_1, b_1)$
            end if
        else
            if $u \preceq_T v$ **then**
                $\delta \leftarrow (a_1, b_1, a_2, b_2)$
                if $v \preceq_T u$ **then** ▷ $u = v$: constraint holds in both directions
                    $\gamma \leftarrow (a_2, b_2, a_1, b_1)$
                end if
            else if $v \prec_T u$ **then**
                $\delta \leftarrow (a_2, b_2, a_1, b_1)$
            end if
        end if
        if $\delta \neq$ **nil then**
            R.Append(δ)
        end if
        if $\gamma \neq$ **nil then**
            R.Append(γ)
        end if
    end for
end for
return R

Bibliography

[1] Anna Lindeberg et al. “Inferring DAGs and Phylogenetic Networks from Least Common Ancestors”. In: *arXiv:2511.07965* (2025).

[2] Marc Hellmuth, David Schaller, and Peter F. Stadler. “Clustering systems of phylogenetic networks”. In: *Theory in Biosciences 142(4):301–358, DOI 10.1007/s12064-023-00398-w* (2023).

[3] David Schaller. *"GitHub - DavidSchaller/tralda · GitHub"*. Accessed: 2026-05-19. URL: `https://github.com/david-schaller/tralda` (visited on 05/19/2026).

[4] Anna Lindeberg and Anton Alfonsson. *"GitHub - AnnaLindeberg/RealLCA · GitHub"*. Accessed: 2026-05-15. URL: `https://github.com/AnnaLindeberg/RealLCA` (visited on 05/15/2026).

Datalogi
www.math.su.se

Beräkningsmatematik
Matematiska institutionen
Stockholms universitet
106 91 Stockholm