Chapter 7

SUPERTREE ALGORITHMS FOR NESTED TAXA

Philip Daniel and Charles Semple

Abstract: Most supertree algorithms combine collections of rooted phylogenetic trees with overlapping leaf sets into a single rooted phylogenetic tree. It is implicit in all these algorithms that the leaves of the rooted phylogenetic trees in the input collection, as a whole, represent non-nested taxa. Thus, for example, the "domestic dog" and "mammal" cannot be represented by two distinct leaves in such a collection because the former is nested inside the latter. In practice, however, one often wants to combine rooted trees in which taxa are nested. In other words, to combine rooted trees in which the leaves as well as some of the interior vertices are labeled. These interior labels represent taxa at a level higher than that of their descendants (e.g., families versus genera, or genera versus species). Moreover, it could happen that a leaf of one of the input trees represents a taxon that is represented by an interior label of another tree. In this chapter, we describe two supertree algorithms for combining rooted trees in which the leaves as well as some of the interior vertices are labeled. Called "rooted semi-labeled trees", these trees are more general than rooted phylogenetic trees in that not only are their leaves labeled, but some of their interior vertices might be as well. Both algorithms are polynomial-time in the size of the input and are motivated by a problem posed by Page in an earlier chapter called "Taxonomy, Supertrees, and the Tree of Life".

Keywords: BUILD; interior labels; leaf labels; nested taxa; taxonomy

1. Introduction

Throughout the chapter, we will assume that the reader is familiar with the basics of phylogenetic trees.

Bininda-Emonds, O. R. P. (ed.) Phylogenetic Supertrees: Combining Information to Reveal the Tree of Life, pp. 1–21. Computational Biology, volume 3 (Dress, A., series ed.).

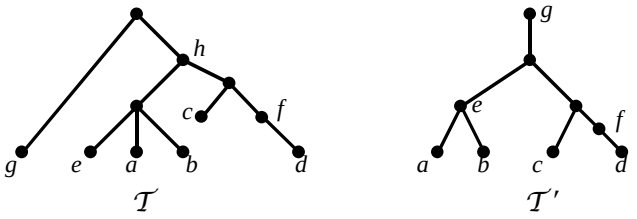

Figure 1. Two rooted semi-labeled trees.

A *rooted semi-labeled tree* (*on X*) is an ordered pair $(T; \phi)$ consisting of a rooted tree T with vertex set V and root vertex ρ, and a map $\phi : X \to V$ with the properties that, for all $v \in V - \{\rho\}$ of degree at most two, $v \in \phi(X)$ and, if ρ has degree zero or one, $\rho \in \phi(X)$. Viewing a rooted tree with edges directed away from the root, every vertex of out-degree 0 or 1 is labeled by an element of X. Intuitively, a rooted semi-labeled tree is simply a rooted tree in which the leaves and some of the interior vertices are labeled. We could avoid some of the technicalities of the formal definition, but then some of the generality is also lost. Rooted semi-labeled trees on X are also called *rooted X-trees*. Two rooted semi-labeled trees are shown in Figure 1. Rooted X-trees extend the notion of rooted phylogenetic X-trees; in the case of the latter, ϕ is a bijective map from X into the set of leaves of T, and the root has degree at least two.

Let $\mathcal{T} = (T; \phi)$ be a rooted X-tree and let X' be a subset of X. The *restriction* of $\mathcal{T}$ to X', denoted $\mathcal{T}|X'$, is the rooted X'-tree that is obtained from the minimal rooted subtree of $\mathcal{T}$ induced by the elements of the set $\phi(X')$ by suppressing all vertices of out-degree 0 or 1 not in $\phi(X')$. We say a rooted X-tree $\mathcal{T}$ *displays* a rooted X' tree $\mathcal{T}'$ if $X' \subseteq X$ and $\mathcal{T}|X'$ is a refinement of $\mathcal{T}'$. For example, in Figure 1, $\mathcal{T}$ displays $\mathcal{T}'$. A collection $\mathcal{P}$ of rooted semi-labeled trees is *compatible* if there exists a rooted semi-labeled tree $\mathcal{T}$ that displays every tree in $\mathcal{P}$, in which case, *T displays* $\mathcal{P}$.

One of the first supertree methods is due to Aho *et al.* (1981). Intended originally for relational databases, their method (called BUILD) provides a polynomial-time solution to the following problem: given a collection $\mathcal{P}$ of rooted phylogenetic trees, does there exist a rooted phylogenetic tree that displays $\mathcal{P}$ and, if so, can we construct such a rooted phylogenetic tree? In terms of evolutionary biology, where one views a rooted phylogenetic tree as representing the ancestral relationships of a set of present-day species, a rooted phylogenetic tree $\mathcal{T}$ displays $\mathcal{P}$ if, up to "polytomies", all the ancestral relationships of every tree in $\mathcal{P}$ is preserved in $\mathcal{T}$. As noted in Semple and Steel (2003), BUILD can be extended easily to determine the compatibility of a collection $\mathcal{P}'$ of rooted semi-labeled trees in polynomial time, and, if they

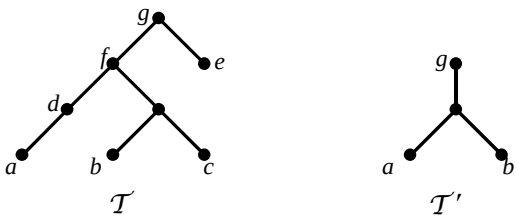

Figure 2. $\mathcal{T}$ perfectly displays $\mathcal{T}'$.

are compatible, to construct a rooted semi-labeled tree that displays this collection. However, under this notion of compatibility, it is possible that $\mathcal{P}'$ is compatible, but the only rooted semi-labeled trees that display $\mathcal{P}'$ all have a particular label labeling a leaf that labeled an interior vertex of a tree in $\mathcal{P}'$ originally. Hence, this notion of compatibility does not preserve descendancy and so, for practical purposes, it appears that it might not be of much use for collections of rooted semi-labeled trees.

In this chapter, we present two polynomial-time supertree algorithms for rooted semi-labeled trees. The first algorithm provides a solution to a problem posed by Page (2004) in another chapter of this book. We call this problem HIGHER TAXA COMPATIBILITY. A rooted X-tree $\mathcal{T}$ *perfectly displays* a rooted X'-tree $\mathcal{T}'$ if $X' \subseteq X$ and $\mathcal{T}|X'$ is isomorphic to $\mathcal{T}'$. In Figure 2, $\mathcal{T}$ perfectly displays $\mathcal{T}'$. Furthermore, a collection $\mathcal{P}$ of rooted semi-labeled trees is *perfectly compatible* if there exists a rooted semi-labeled tree $\mathcal{T}$ that perfectly displays every tree in $\mathcal{P}$, in which case, $\mathcal{T}$ *perfectly displays* $\mathcal{P}$. Note that if $\mathcal{T}$ perfectly displays $\mathcal{T}'$, then $\mathcal{T}$ displays $\mathcal{T}'$. However, the converse does not hold necessarily.

Problem: HIGHER TAXA COMPATIBILITY

Instance: A collection $\mathcal{P}$ of rooted semi-labeled trees.
Question: Does there exist a rooted semi-labeled tree that perfectly displays $\mathcal{P}$ and, if so, can we construct such a rooted semi-labeled tree?

The motivation for HIGHER TAXA COMPATIBILITY arises when one wants to use a supertree method for combining evolutionary trees the internal vertices of which as well as their leaves are labeled. Such trees contain taxa at different taxonomic levels. Indeed, it could happen that a higher taxon labels an internal vertex in one of the trees we want to combine, but it labels a leaf in another. The algorithm that we present to solve this problem is called SEMI-LABELEDBUILD.

Our first response in trying to solve HIGHER TAXA COMPATIBILITY was to use the extension of BUILD for determining the compatibility of a collection of rooted semi-labeled trees in some way. However, despite SEMI-LABELEDBUILD having a similar description to this extension, it seems that no straightforward modification solves this problem.

The notion of perfectly displays is very restrictive in the sense that a collection $\mathcal{P}$ of semi-labeled trees is perfectly compatible precisely if there is a rooted semi-labeled tree that preserves all the most recent common ancestor relationships described by $\mathcal{P}$. Thus, perfectly displays does not allow for the resolution of any polytomies in $\mathcal{P}$. To accommodate the possible resolution of polytomies, but still preserve all the descendancy relationships in $\mathcal{P}$, we introduce a second notion of displays called "ancestrally displays" in the second half of this chapter. In comparison with the other two notions of displays, ancestrally displays is weaker than perfectly displays because it might not preserve all the most recent common ancestor relations, but it is stronger than the usual notion of displays because it preserves descendancy. The second algorithm in this chapter determines the compatibility of $\mathcal{P}$ under this second notion.

Unless otherwise stated, the notation and terminology in this chapter follow Semple and Steel (2003). The chapter is organized as follows. In Section 2, we describe some necessary preliminaries. In Section 3, we present SEMI-LABELEDBUILD and show that it does indeed provide a polynomial-time solution to HIGHER TAXA COMPATIBILITY. The last section defines ancestrally displays formally and presents a polynomial-time algorithm for solving the associated compatibility problem.

2. Preliminaries

Let $\mathcal{T} = (T; \phi)$ be a rooted semi-labeled tree. The tree T and the map ϕ are called the *underlying tree* and *labeling map* of $\mathcal{T}$. The domain of ϕ is the *label set* of $\mathcal{T}$ and is denoted $\mathcal{L}(\mathcal{T})$. We shall often refer to the elements of $\mathcal{L}(\mathcal{T})$ as *labels*. If v is a vertex of T, we say that the elements of $\phi^{-1}(v)$ *label* v. If ρ is the root of T, then the elements of $\phi^{-1}(\rho)$ are called *root labels*. Furthermore, T is *fully labeled* if $\phi^{-1}(v)$ is non-empty for all vertices v of T. For a collection $\mathcal{P}$ of rooted semi-labeled trees, we denote the set $\cup_{\mathcal{T} \in \mathcal{R}} \mathcal{L}(\mathcal{T})$ by $\mathcal{L}(\mathcal{P})$.

For a rooted tree T, a particularly useful partial order $\leq_T$ on the vertex set V of T is obtained by setting $u \leq_T v$ if the path from the root of T to v includes u. If $u \leq_T v$, we say that v is a *descendant* of u or, alternatively, u is an *ancestor* of v. Furthermore, u and v are *comparable* under $\leq_T$ if either $u \leq_T v$ or $v \leq_T u$; otherwise u and v are *not comparable*. Observe that the partial

order $\leq_T$ has the property that, for every pair of elements, the greatest lower bound exists. The greatest lower bound of x and y under $\leq_T$ is called the *most recent common ancestor* of x and y and is denoted $\mathrm{mrca}_T(x, y)$.

The above partial order extends naturally to the label set of a rooted semi-labeled tree as follows. Let $\mathcal{T} = (T; \phi)$ be a rooted X-tree and let $a, b \in X$. Then, $a \leq_{\mathcal{T}} b$ if $\phi^{-1}(a) \leq_T \phi^{-1}(b)$, in which case, b is a *descendant label* of a or, alternatively, a is an *ancestor label* of b. Furthermore, for all $a, b \in X$, we let

$$\mathrm{mrca}_{\mathcal{T}}(a, b) = \phi^{-1}\left(\mathrm{mrca}_T(\phi(a), \phi(b))\right).$$

Note that, because $\mathcal{T}$ is only semi-labeled, this set could be empty. However, if $\mathcal{T}$ is fully-labeled, then this set is non-empty.

Let $\mathcal{T} = (T; \phi)$ be a rooted semi-labeled tree and let u be a vertex of T. An element a of $\mathcal{L}(\mathcal{T})$ is a *descendant label* of u if $u \leq_T \phi^{-1}(a)$. The set of descendant labels of a non-root vertex v of T is called a *cluster* and we often refer to it as the *cluster of T corresponding to v*. The collection of clusters of $\mathcal{T}$ is denoted by $\mathcal{H}(\mathcal{T})$. Up to isomorphism of the underlying trees, no two rooted semi-labeled trees have the same collection of clusters; thus, a rooted semi-labeled tree $\mathcal{T}$ is determined completely by its set of clusters (see Theorem 3.5.2 in Semple and Steel, 2003). Indeed, $\mathcal{T}$ can be constructed quickly and easily from $\mathcal{H}(\mathcal{T})$.

In Section 1, we defined the restriction of a rooted X-tree $\mathcal{T}$ to a subset X' of X and denoted it by $\mathcal{T}|X'$. An equivalent definition of $\mathcal{T}|X'$ is the rooted X'-tree for which

$$\mathcal{H}(\mathcal{T}|X') = \{C \cap X' : C \in \mathcal{H}(\mathcal{T}) \text{ and } C \cap X' \neq \emptyset\}$$

This equivalence will prove useful in this chapter.

3. The SEMI-LABELEDBUILD algorithm

In this section, we describe the algorithm SEMI-LABELEDBUILD. We begin by defining a particular graph that will play a prominent role in the algorithm. A vertex of a graph is *isolated* if it is incident with no edges. Let $\mathcal{P}$ be a collection of rooted fully-labeled trees. The *cluster and* root-label graph of $\mathcal{P}$, and denoted $G(\mathcal{P})$, has vertex set $\mathcal{L}(\mathcal{P})$ and an edge set consisting of three types of edges that are added sequentially as follows:

(i) First, two (not necessarily distinct) vertices are joined by a blue edge if they appear in the same cluster of a fully-labeled tree in $\mathcal{P}$. Note

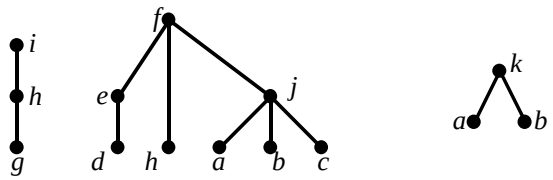

Figure 3. A collection $G(\mathcal{P})$ of rooted fully-labeled trees.

this implies that, for any element of $\mathcal{L}(\mathcal{P})$ that is in a cluster, there is a loop joining this element to itself.

(ii) Second, if a is a root label of a fully-labeled tree, $\mathcal{T}$ say, in $\mathcal{P}$ and a is not isolated, then join a to every other label in $\mathcal{L}(\mathcal{T})$ by a blue edge.

(iii) Third, once all of the edges associated with (i) and (ii) have been added, do the following:

(a) For all isolated vertices c, if there is a tree $\mathcal{T}$ in $\mathcal{P}$ with labels $a, b \in \mathcal{L}(\mathcal{T})$ such that $c \in \text{mrca}_{\mathcal{T}}(a, b)$, join a and b with a red edge labeled c. This labeled edge can be in parallel with a blue edge or other red edges (see remark below).

(b) For any two vertices joined by a red edge labeled c, if there is a path connecting them consisting of just blue edges, delete every red edge labeled c and join c to each $d \in \mathcal{L}(\mathcal{P})$ with a blue edge if both c and d are in the label set of some particular tree in $\mathcal{P}$.

(c) Repeat (b) until there are no pairs of vertices joined by red edges that are connected by a path consisting of just blue edges.

(d) Delete all remaining red edges.

We call a blue edge of $G(\mathcal{P})$ a *type (i), (ii),* or *(iii)* edge depending upon whether it is added at Step (i), (ii), or (iii), respectively, in the construction of $G(\mathcal{P})$.

Remark. In the construction of $G(\mathcal{P})$, once a blue edge has been added to join two, not necessarily distinct, vertices, no further blue edge need be added in parallel with this edge. However, red edges with distinct labels are added in parallel because each such edge plays a particular role in the construction.

As an example of the above construction, let $\mathcal{P}$ be the collection of rooted fully-labeled trees shown in Figure 3. Then, omitting loops, the construction of $G(\mathcal{P})$ to the end of Step (ii) is shown in Figure 4. At Step (iii)(a), red edges are added between pairs $\{a, b\}$, $\{d, a\}$, $\{d, b\}$, $\{d, c\}$, $\{d, j\}$, $\{e, a\}$, $\{e, b\}$, $\{e, c\}$, $\{e, j\}$, $\{h, a\}$, $\{h, b\}$, $\{h, c\}$, $\{h, j\}$, $\{h, d\}$ and $\{h, e\}$. The red

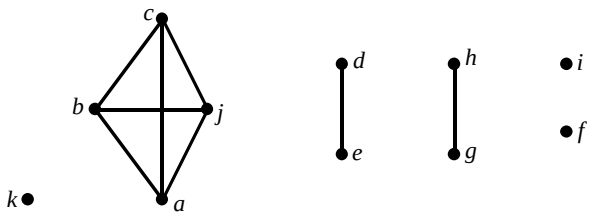

Figure 4. Construction of $G(\mathcal{P})$ at the end of steps (i) and (ii).

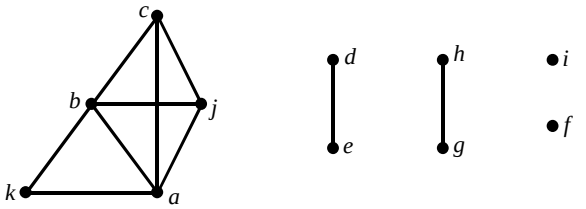

Figure 5. Completed construction of $G(\mathcal{P})$.

edge between *a* and *b* gives blue edges between *k* and *a* and between *k* and *b* at Step (iii)(b). All other red edges are deleted at Step (iii)(d) without having affected the graph. With loops omitted, the final construction of $G(\mathcal{P})$ is shown in Figure 5.

Before describing SEMI-LABELEDBUILD, we need to define one further construction. Let $\mathcal{T} = (T; \phi)$ be a rooted semi-labeled tree on X, where T has vertex set V. We say that a rooted fully-labeled tree $\mathcal{T}_1 = (T; \phi_1)$ on X_1, where $X \subseteq X_1$, has been obtained from $\mathcal{T}$ by *adding distinct new labels* if, for all distinct $u, v \in V$, the following properties are satisfied:

1. If $\phi^{-1}(v)$ is non-empty, then $\phi_1^{-1}(v) = \phi^{-1}(v)$.
2. If $\phi^{-1}(v)$ is empty, then $\left|\phi_1^{-1}(v)\right| = 1$.
3. If $\phi^{-1}(u)$ and $\phi^{-1}(v)$ are both empty, then $\phi_1^{-1}(u) \neq \phi_1^{-1}(v)$.

Intuitively, $\mathcal{T}_1$ has been obtained from $\mathcal{T}$ by labeling non-labeled vertices of $\mathcal{T}$ singularly with distinct new labels. For a collection $\mathcal{P}$ of rooted semi-labeled trees, we say that $\mathcal{P}_1$ has been obtained from $\mathcal{P}$ by *adding distinct new labels* if it has been obtained by adding distinct new labels to every tree in $\mathcal{P}$ so that, for any pair of trees, no two new labels are the same.

Essentially, all the work in SEMI-LABELEDBUILD is done by a subroutine called FULLY-LABELEDBUILD. We describe each in turn.

Algorithm: SEMI-LABELEDBUILD($\mathcal{P}$, $\mathcal{T}$)

Input: A collection $\mathcal{P}$ of rooted semi-labeled trees.
Output: A rooted semi-labeled tree $\mathcal{T}$ that perfectly displays $\mathcal{P}$ or the statement *not perfectly compatible*.

1. Construct a collection $\mathcal{P}'$ of rooted fully-labeled trees from $\mathcal{P}$ by adding distinct new labels.
2. Call the subroutine FULLY-LABELEDBUILD($\mathcal{P}'$, v', $\mathcal{T}'$).
3. If FULLY-LABELEDBUILD returns *not perfectly compatible*, then return *not perfectly compatible*.
4. If FULLY-LABELEDBUILD returns a rooted fully-labeled tree $\mathcal{T}'$, then remove the added labels and return the resulting rooted semi-labeled tree $\mathcal{T}$.

Algorithm: FULLY-LABELEDBUILD($\mathcal{P}'$, v', $\mathcal{T}'$)

Input: A collection $\mathcal{P}'$ of rooted fully-labeled trees.
Output: A rooted fully-labeled tree $\mathcal{T}'$ with root vertex v' that perfectly displays $\mathcal{P}'$ or the statement *not perfectly compatible*.

1. Construct the cluster and root-label graph $G(\mathcal{P}')$ of $\mathcal{P}'$.
2. If $G(\mathcal{P}')$ has no isolated vertices, then halt and return *not perfectly compatible*.
3. Otherwise, let S_1, S_2, …, S_k denote the vertex sets of the connected components of $G(\mathcal{P}')$ not consisting of an isolated vertex, and let S_0 denote the set of isolated vertices of $G(\mathcal{P}')$.
4. Initialize $\mathcal{T}'$ with a single root vertex v' and assign all labels in S_0 to v'.
5. For each $i \in \{1, 2, \ldots, k\}$, call FULLY-LABELEDBUILD($\mathcal{P}_i'$, v', $\mathcal{T}_i'$), where $\mathcal{P}_i'$ is the collection of rooted fully-labeled trees obtained from $\mathcal{P}'$ by restricting each tree in $\mathcal{P}'$ to S_i. If FULLY-LABELEDBUILD($\mathcal{P}_i'$, v', $\mathcal{T}_i'$) returns a tree, then attach $\mathcal{T}_i'$ to v' via the edge $\{v'_i, v'\}$.

Intuitively, for a set $\mathcal{P}'$ of rooted fully-labeled trees, FULLY-LABELEDBUILD attempts to construct a rooted fully-labeled tree $\mathcal{T}'$ that perfectly displays $\mathcal{P}'$ by essentially constructing $\mathcal{H}(\mathcal{T}')$. This is done by beginning with $\mathcal{L}(\mathcal{P}')$ and breaking it down successively into disjoint subclusters. How clusters are broken up in this way is determined by the connected components of the associated cluster and root-label graph. Components consisting of isolated vertices are distinguished from those not consisting of isolated vertices. This process continues provided the

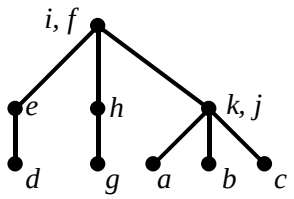

Figure 6. The rooted fully-labeled tree outputted by FULLY-LABELEDBUILD when applied to the collection of trees shown in Figure 3.

associated cluster and root-label graph has at least one isolated vertex at each iteration, in which case, FULLY-LABELEDBUILD returns a rooted fully-labeled tree. By contrast, if the associated cluster and root-label graph has no isolated vertices at some iteration, then FULLY-LABELEDBUILD returns "$\mathcal{P}'$ is not perfectly compatible".

Remark.

1. It is an immediate consequence of Lemma 3.2 below that, for all i in Step 4 of FULLY-LABELEDBUILD, $\mathcal{P}_i'$ is indeed a collection of rooted fully-labeled trees as indicated in this step.
2. Using the fact that FULLY-LABELEDBUILD considers proper restrictions of the input collection of rooted fully-labeled trees successively in Step 4 of the algorithm, it is seen easily that FULLY-LABELEDBUILD returns either "$\mathcal{P}'$is not perfectly compatible" or a rooted fully-labeled tree with label set $\mathcal{L}(\mathcal{P}')$. Consequently, SEMI-LABELEDBUILD returns either "$\mathcal{P}$ is not perfectly compatible" or a rooted semi-labeled tree with label set $\mathcal{L}(\mathcal{P})$.

To illustrate FULLY-LABELEDBUILD, the rooted fully-labeled tree shown in Figure 6 is the result of applying this algorithm to the collection of rooted fully-labeled trees shown in Figure 3.

The main result of this chapter is the following theorem.

Theorem 3.1. *Let $\mathcal{P}$ be a collection of rooted semi-labeled trees. Then* SEMI-LABELEDBUILD *applied to $\mathcal{P}$ either:*

(i) *returns a rooted semi-labeled tree that perfectly displays $\mathcal{P}$ if $\mathcal{P}$ is perfectly compatible, or*

(ii) *returns the statement $\mathcal{P}$* is not perfectly compatible *otherwise.*

To prove Theorem 3.1, we establish first some lemmas.

Lemma 3.2. *Let $\mathcal{P}$ be a collection of rooted fully-labeled trees and consider the cluster and root-label graph $G(\mathcal{P})$ of $\mathcal{P}$. Let $\mathcal{T}$ be an element of $\mathcal{P}$ and let S_0 denote the set of isolated vertices of $G(\mathcal{P})$. Then the following hold:*

(i) *If $\mathcal{L}(\mathcal{P}) \cap S_0$ is non-empty, then all the elements of this set are root labels of $\mathcal{T}$. Furthermore, if d is a root label of $\mathcal{T}$ and $d \in S_0$, then all root labels of $\mathcal{T}$ are elements of S_0.*

(ii) *If no root label of $\mathcal{T}$ is an element of S_0, then $\mathcal{L}(\mathcal{T})$ is a subset of the vertex set of some connected component of $G(\mathcal{P})$.*

(iii) *Suppose that a root label of $\mathcal{T}$ is an element of S_0. Let A and B be distinct maximal clusters of $\mathcal{T}$. Then A and B are subsets of the vertex sets of distinct connected components of $G(\mathcal{P})$.*

Proof. Using the construction of $G(\mathcal{P})$, the proofs of (i), (ii), and (iii) are straightforward. We omit the details. □

Lemma 3.3. *Let $\mathcal{T}_1$ be a rooted fully-labeled tree on X_1 and let $\mathcal{T}_2$ be a rooted semi-labeled tree on X_2 such that $X_1 \subseteq X_2$. Then $\mathcal{T}_2$ perfectly displays $\mathcal{T}_1$ if and only if, for all $a, b \in X_1$,*

$$\mathrm{mrca}_{\mathcal{T}_1}(a, b) = \mathrm{mrca}_{\mathcal{T}_2}(a, b) \,|\, X_1.$$

Proof. For each $i \in \{1, 2\}$, let $\mathcal{H}_i$ denote the collection of clusters of $\mathcal{T}_i$ plus X_i. Suppose that $\mathcal{T}_2$ perfectly displays $\mathcal{T}_1$. Let $a, b, c \in X_1$ and suppose that $c \in \mathrm{mrca}_{\mathcal{T}_1}(a, b)$. Then, for any $C_1 \in \mathcal{H}_1$, $\{a, b\}$ is a subset of C_1 if and only if $c \in C_1$. Because $\mathcal{H}_1 = \mathcal{H}_2 \,|\, X_1$, it follows that, for any $C_2 \in \mathcal{H}_2$, $\{a, b\}$ is a subset of C_2 if and only if $c \in C_2$. Therefore, c is a common ancestor of a and b in $\mathcal{T}_2$, and no common ancestor of a and b in $\mathcal{T}_2$ is a proper descendent of c in $\mathcal{T}_2$. Thus, $c \in \mathrm{mrca}_{\mathcal{T}_2}(a, b) \,|\, X_1$, and, more generally, $\mathrm{mrca}_{\mathcal{T}_1}(a, b) \subseteq \mathrm{mrca}_{\mathcal{T}_2}(a, b) \,|\, X_1$. For all x in $X_1 - \mathrm{mrca}_{\mathcal{T}_1}(a, b)$, there is either an element of $\mathcal{H}_1$ containing x but not c, or there is an element of $\mathcal{H}_1$ containing c but not x. Thus, because $\mathcal{H}_1 = \mathcal{H}_2 \,|\, X_1$, there is either an element of $\mathcal{H}_2$ containing x but not c or there is an element of $\mathcal{H}_2$ containing c but not x. Therefore, c and x do not label the same vertex of $\mathcal{T}_2$, and so $x \notin \mathrm{mrca}_{\mathcal{T}_2}(a, b) \,|\, X_1$. This shows that $\mathrm{mrca}_{\mathcal{T}_1}(a, b) = \mathrm{mrca}_{\mathcal{T}_2}(a, b) \,|\, X_1$.

Now suppose that, for all $a, b \in X_1$, we have $\mathrm{mrca}_{\mathcal{T}_1}(a, b) = \mathrm{mrca}_{\mathcal{T}_2}(a, b) \cap X_1$, but $\mathcal{H}_1 \neq \mathcal{H}_2 \,|\, X_1$. If $\mathcal{H}_2 \,|\, X_1$ is a proper subset of $\mathcal{H}_1$, then $\mathcal{T}_1$ is a proper refinement of $\mathcal{T}_2 \,|\, X_1$ and it is checked easily that there is a pair of distinct elements $a, b \in X_1$ such that $\mathrm{mrca}_{\mathcal{T}_1}(a, b) \neq \mathrm{mrca}_{\mathcal{T}_2}(a, b) \,|\, X_1$. Therefore, we can assume that there is an element, C_2 say, of $\mathcal{H}_2 \,|\, X_1$ that is not an element of $\mathcal{H}_1$. Let C_1 be the minimal cluster of $\mathcal{T}_1$ that contains C_2 and let x be an element of $C_1 - C_2$. If x is a label of the vertex of $\mathcal{T}_1$ that corresponds to C_1, then, by the minimality of C_1, there is a pair of distinct

elements $a, b \in C_1$ such that $x \in \text{mrca}_{\mathcal{T}_1}(a, b)$, but $x \notin \text{mrca}_{\mathcal{T}_2}(a, b) \mid X_1$. It follows that we can assume also that no element of $C_1 - C_2$ labels the vertex of $\mathcal{T}_1$ corresponding to C_1. But then, if c is such a label, it is seen easily that $\text{mrca}_{\mathcal{T}_1}(x, c) \neq \text{mrca}_{\mathcal{T}_2}(x, c) \mid X_1$. This completes the proof of Lemma 3.3. □

Lemma 3.4. Let $\mathcal{P}$ be a collection of rooted semi-labeled trees. Let $\mathcal{P}'$ be a set of rooted fully-labeled trees obtained from $\mathcal{P}$ by adding distinct new labels. Then $\mathcal{P}$ is perfectly compatible if and only if $\mathcal{P}'$ is perfectly compatible. Moreover, if $\mathcal{T}'$ is a rooted semi-labeled tree that perfectly displays $\mathcal{P}'$, then $\mathcal{T}'$ perfectly displays $\mathcal{P}$.

Proof. Suppose that $\mathcal{P}$ is perfectly compatible and let $\mathcal{T}$ be a rooted semi-labeled tree that perfectly displays $\mathcal{P}$. For each element $c' \in \mathcal{L}(\mathcal{P}') - \mathcal{L}(\mathcal{P})$, there is a unique rooted fully-labeled tree, $\mathcal{T}_1'$ say, in $\mathcal{P}'$ for which $c' \in \mathcal{L}(\mathcal{T}_1')$. Furthermore, because c' is one of the added labels, c' labels a vertex of $\mathcal{T}_1'$ of degree at least three and so there exist labels a and b of $\mathcal{T}_1$ such that $\text{mrca}_{\mathcal{T}_1'}(a, b) = \{c'\}$, where $\mathcal{T}_1$ is the tree in $\mathcal{P}$ corresponding to $\mathcal{T}_1'$, because all leaves of $\mathcal{T}_1$ must be labeled.

Now let $\mathcal{T}'$ be the rooted semi-labeled tree obtained from $\mathcal{T}$ by adding the labels of $\mathcal{L}(\mathcal{P}') - \mathcal{L}(\mathcal{P})$ so that if $c' \in \mathcal{L}(\mathcal{P}') - \mathcal{L}(\mathcal{P})$, $c' \in \mathcal{T}_1'$, and $\text{mrca}_{\mathcal{T}_1'}(a, b) = \{c'\}$ for some labels a and b of $\mathcal{T}_1$, then $c' \in \text{mrca}_{\mathcal{T}'}(a, b)$. By the previous paragraph and the fact that $\mathcal{T}$ perfectly displays $\mathcal{P}$, it is seen easily that, for all $a, b \in \mathcal{L}(\mathcal{T}_1')$,

$$\text{mrca}_{\mathcal{T}_1'}(a, b) = \text{mrca}_{\mathcal{T}'}(a, b) \mid \mathcal{L}(\mathcal{T}_1')$$

It now follows by Lemma 3.3 that $\mathcal{T}'$ perfectly displays $\mathcal{T}_1'$ and hence $\mathcal{P}'$.

The rest of the proof of Lemma 3.4 is straightforward and omitted. □

Lemma 3.5. *Let $\mathcal{P}$ be a collection of rooted fully-labeled trees. If $\mathcal{P}$ is perfectly compatible, then there exists a rooted fully-labeled tree that perfectly displays $\mathcal{P}$ with label set $\mathcal{L}(\mathcal{P})$.*

Proof. Suppose that $\mathcal{P}$ is perfectly compatible and let $\mathcal{T} = (T; \phi)$ be a rooted semi-labeled tree that perfectly displays $\mathcal{P}$ and has label set $\mathcal{L}(\mathcal{P})$. Now suppose that among all rooted semi-labeled trees that perfectly display $\mathcal{P}$ and have label set $\mathcal{L}(\mathcal{P})$, the tree T has the least number of unlabeled vertices. If T has no unlabeled vertices, then the lemma is proved. Therefore, assume that there is a vertex, u say, of T that is unlabeled. Because $\mathcal{T}$ is a rooted semi-labeled tree, u has out-degree at least two. Furthermore, because $\mathcal{T}$ perfectly displays $\mathcal{P}$ and $\mathcal{P}$ is a collection of rooted fully-labeled trees, it follows by Lemma 3.3 that there is no tree $\mathcal{T}_1$ in $\mathcal{P}$ with labels a and b such

that $\mathrm{mrca}_{\mathcal{T}_1}(\phi(a), \phi(b)) = u$. By Lemma 3.3 again, this in turn implies that if $v_1, v_2, \ldots, v_n$ are immediate descendants of u in T, then the rooted semi-labeled tree obtained from T by contracting $\{u, v_i\}$ for all i and labeling the identified vertex with $\cup_{i\in\{1,\ldots,n\}} \phi_1^{-1}(v_i)$ perfectly displays $\mathcal{P}$. But the latter tree has one less unlabeled vertex than $\mathcal{T}$. This contradiction completes the proof of the lemma. □

It follows from Lemma 3.4 and the description of SEMI-LABELEDBUILD that Theorem 3.1 is an immediate consequence of the following theorem.

Theorem 3.6. *Let $\mathcal{P}$ be a collection of rooted fully-labeled trees. Then* FULLY-LABELEDBUILD *applied to $\mathcal{P}$ either:*

(i) *returns a rooted fully-labeled tree that perfectly displays $\mathcal{P}$ if $\mathcal{P}$ is perfectly compatible, or*

(ii) *returns the statement* $\mathcal{P}$ is not perfectly compatible *otherwise.*

Proof. First suppose that $\mathcal{P}$ is perfectly compatible. Under this assumption, we show that FULLY-LABELEDBUILD applied to $\mathcal{P}$ outputs a rooted fully-labeled tree. If this is not the case, then FULLY-LABELEDBUILD outputs *not perfectly compatible*, in which case, the associated cluster and root-label graph, G say, has no isolated vertices at some iteration of the algorithm. Let S denote the vertex set of G. Because $\mathcal{P}$ is perfectly compatible, $\mathcal{P}|S$ is perfectly compatible and so, by Lemma 3.5, there exists a rooted fully-labeled tree $\mathcal{T}$ with labeled set S that perfectly displays $\mathcal{P}|S$. Let c be a root label of $\mathcal{T}$. Then, because $\mathcal{T}$ perfectly displays $\mathcal{P}|S$, every tree of $\mathcal{P}|S$ in which c is a label has the property that c is a root label. Furthermore, because G has no isolated vertices, c must be joined to an element of $S - \{c\}$ in G by some edge and this edge must be a type (iii) edge; for otherwise, c is a non-root label of some tree in $\mathcal{P}|S$. It now follows that there exists a tree $\mathcal{T}_1$ in $\mathcal{P}|S$ such that a, b, c are distinct vertices of $\mathcal{L}(\mathcal{T}_1)$, $c \in \mathrm{mrca}_{\mathcal{T}_1}(a, b)$, and, in G, there is a path joining a and b. We next show that the existence of this path implies that a and b are elements of a cluster of $\mathcal{T}$.

Let G_0 denote the graph with vertex set S and the edge set of which consists of all type (i) and (ii) edges of G. Let u and v be any two vertices of G_0. If $\{u, v\}$ is an edge of this graph, then, using the fact that $\mathcal{T}$ perfectly displays $\mathcal{P}$, it is checked easily that u and v must be in the same maximal cluster of $\mathcal{T}$. Clearly, being in the same maximal cluster of a given rooted semi-labeled tree is a transitive relation and so if there is a path in G_0 joining two vertices, then these two vertices are in the same maximal cluster of $\mathcal{T}$. Now G is obtained from G_0 by adding sets of edges iteratively that join a particular root label of trees in $\mathcal{P}|S$ to all its descendant labels. Let $E_1, E_2, \ldots, E_k$ denote the corresponding sequence of these added sets of edges, and

let $z_1, z_2, \ldots, z_k$ denote the associated sequence of particular root labels. Let E_0 denote the edge set of G_0 and, for all $i \in \{1, 2, \ldots k\}$, let G_i denote the graph with vertex set S and edge set $E_0 \cup E_1 \cup E_2 \cup \ldots \cup E_i$.

Consider the graph G_1. By the construction of G_1, there is a rooted fully-labeled tree T_1 in $\mathcal{P}|S$ with root label z_1 and distinct proper descendant labels x_1 and y_1 such that $z_1 \in \text{mrca}_{\mathcal{T}_1}(x_1, y_1)$ and, in G_0, there is a path joining x_1 and y_1. By the existence of this path and the previous paragraph, x_1 and y_1 are in the same maximal cluster of $\mathcal{T}$. Because $\mathcal{T}$ perfectly displays $\mathcal{P}|S$ and $z_1 \in \text{mrca}_{\mathcal{T}_1}(x_1, y_1)$, it follows that z_1 must also be in this particular maximal cluster of $\mathcal{T}$. Because all the edges in E_1 contain z_1 and because G_0 has the transitive property mentioned in the last paragraph, G_1 also has the property that if there is a path joining two vertices in G_1, then these two vertices are in the same maximal cluster of $\mathcal{T}$. Continuing in this way for G_2, G_3, …, G_{k-1} and lastly for G_k, we deduce that G_k, and hence $G(\mathcal{P}|S)$, also have this edge transitive property. But then, because a and b are joined by a path in $G(\mathcal{P}|S)$, a and b are in the same maximal cluster of $\mathcal{T}$ and so $c \notin \text{mrca}_{\mathcal{T}}(a, b)$. This contradiction shows that FULLY-LABELEDBUILD does indeed output a rooted fully-labeled tree if $\mathcal{P}$ is compatible.

Now suppose that FULLY-LABELEDBUILD outputs a rooted fully-labeled tree $\mathcal{T}$. Here, we show that $\mathcal{T}$ perfectly displays $\mathcal{P}$. Let $\mathcal{T}_1$ be a rooted semi-labeled tree in $\mathcal{P}$ with label set X_1 and let $a, b \in X_1$. By Lemma 3.3, it suffices to show that $\text{mrca}_{\mathcal{T}_1}(a, b) = \text{mrca}_{\mathcal{T}}(a, b)|X_1$.

We show first that if $c \in \text{mrca}_{\mathcal{T}_1}(a, b)$, then $c \in \text{mrca}_{\mathcal{T}}(a, b)|X_1$. Throughout this part of the proof, we freely use the fact that, because $c \in \text{mrca}_{\mathcal{T}_1}(a, b)$, we have $c \in \text{mrca}_{\mathcal{T}_1}(a, c)$ and $c \in \text{mrca}_{\mathcal{T}_1}(b, c)$. Let S be the minimal cluster of $\mathcal{T}$ that contains a, b, and c, and consider the graph $G(\mathcal{P}|S)$. If c is not isolated, then either a and c are in the same cluster of a fully-labeled tree in $\mathcal{P}$ or c is the root label of a fully-labeled tree in $\mathcal{P}$. In both cases, it follows that a and c are in the same connected component of $G(\mathcal{P}|S)$. Similarly, b and c are in the same connected component of $G(\mathcal{P}|S)$. It now follows that a, b, and c must be in the same connected component of $G(\mathcal{P}|S)$. This implies that S is not the minimal cluster of $\mathcal{T}$ that contains a, b, and c. Therefore, we can assume that c is an isolated vertex of $G(\mathcal{P}|S)$ and, moreover, that it labels the vertex of $\mathcal{T}$ corresponding to S.

If a, b, and c label the same vertex of $\mathcal{T}_1$, then, by symmetry, a, b, and c are all isolated vertices of $G(\mathcal{P}|S)$ and so a, b, and c label the vertex of $\mathcal{T}$ corresponding to S. Hence, in this case, $c \in \text{mrca}_{\mathcal{T}}(a, b)|X_1$.

Now assume that a and b do not label the same vertex of $\mathcal{T}_1$, but that b and c do. Then, because c is isolated, it follows by the argument above that b is also an isolated vertex of $G(\mathcal{P}|S)$. Furthermore, b also labels the vertex of T corresponding to S. Because $c \in \text{mrca}_{\mathcal{T}_1}(a, b)$, a is not isolated in $G(\mathcal{P}|S)$ and so $c \in \text{mrca}_{\mathcal{T}}(a, b)|X_1$.

Lastly, assume that no pair of a, b, and c label the same vertex of $\mathcal{T}_1$. Because c is isolated, c must have been isolated after Step (ii) in the construction of $G(\mathcal{P}|S)$, and so the relation $c \in \text{mrca}_{\mathcal{T}_1}(a, b)$ implies that a and b are joined by a red edge labeled c in Step (iii)(a) of this construction. Moreover, because c remains isolated at the end of the construction, a and b must be in separate connected components of $G(\mathcal{P}|S)$. Therefore, $c \in \text{mrca}_{\mathcal{T}}(a, b)|X_1$.

We have now established $\text{mrca}_{\mathcal{T}_1}(a, b) \subseteq \text{mrca}_{\mathcal{T}}(a, b)|X_1$. It follows from Lemma 3.2 that, for all $a, b \in \mathcal{L}(\mathcal{T}_1)$, a and b label the same vertex of $\mathcal{T}$ precisely if a and b label the same vertex of $\mathcal{T}_1$. By the argument in the preceding paragraph, $\text{mrca}_{\mathcal{T}_1}(a, b) \cap \text{mrca}_{\mathcal{T}}(a, b)|X_1$ is non-empty. Hence, $\text{mrca}_{\mathcal{T}_1}(a, b) = \text{mrca}_{\mathcal{T}}(a, b)|X_1$. □

We now consider the running time of SEMI-LABELEDBUILD applied to a collection $\mathcal{P}$ of rooted semi-labeled trees. Because it is more than likely that there is a faster method for determining if $\mathcal{P}$ is perfectly compatible, a detailed analysis is omitted. The point is to show that there exists a polynomial-time algorithm (in the size of $\mathcal{L}(\mathcal{P})$) for determining perfect compatibility of $\mathcal{P}$.

Let $\mathcal{P}'$ be a collection of rooted fully-labeled trees obtained from $\mathcal{P}$ by adding distinct new labels. Because all vertices of degree at most two are labeled in a rooted semi-labeled tree, the only possible unlabeled vertices are interior vertices with degree at least three. The number of such interior vertices is at most one less than the number of leaves. Summing over all trees in $\mathcal{P}$, this implies that $|\mathcal{L}(\mathcal{P}')| - |\mathcal{L}(\mathcal{P})| \leq |\mathcal{L}(\mathcal{P})| - 1$. Thus, it suffices to show that the running of SEMI-LABELEDBUILD is polynomial in $|\mathcal{L}(\mathcal{P}')|$. Clearly, the construction of the cluster and root-label graph at each iteration of FULLY-LABELEDBUILD can be done in such a time. Furthermore, because we consider only proper restrictions of the input collection of rooted fully-labeled trees at Step 4 of FULLY-LABELEDBUILD, the number of iterations of FULLY-LABELEDBUILD is bounded by $\mathcal{L}(\mathcal{P}')$. It now follows that the running time of SEMI-LABELEDBUILD is polynomial in the size of $\mathcal{L}(\mathcal{P}')$.

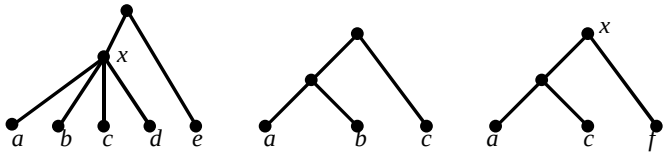

Figure 7. A collection of semi-labeled trees.

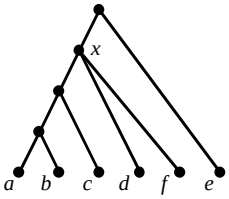

Figure 8. The rooted semi-labeled tree outputted by ANCESTRALBUILD when applied to the rooted semi-labeled trees in Figure 7.

4. Ancestrally displays

If $\mathcal{T}$ is a rooted semi-labeled tree that perfectly displays a collection $\mathcal{P}$ of rooted semi-labeled trees, then $\mathcal{T}$ preserves all the most recent common ancestor relationships described by $\mathcal{P}$. As a consequence, no polytomies in $\mathcal{P}$ are resolved in $\mathcal{T}$. Thus, as mentioned in Section 1, the notion of perfectly compatible is very strong. In this section, we introduce a notion of compatibility that allows the resolution of polytomies, but still maintains all the descendancy relationships of a collection of rooted semi-labeled trees. Moreover, we present a polynomial-time algorithm for determining if a collection of rooted semi-labeled trees is compatible under this new notion.

Let $X' \subseteq X$. A rooted semi-labeled tree $\mathcal{T}$ on X *ancestrally displays* a rooted semi-labeled tree $\mathcal{T}'$ on X' if $\mathcal{T} \mid X'$ refines $\mathcal{T}'$, and for all $a, b \in X'$, the following hold:

1. if $a <_{\mathcal{T}'} b$, then $a <_{\mathcal{T}} b$, and
2. if a is not comparable to b in $\mathcal{T}'$ under $<_{\mathcal{T}'}$, then a is not comparable to b in $\mathcal{T}$ under $<_{\mathcal{T}}$.

Intuitively, (1) and (2) imply that $\mathcal{T}$ preserves the ancestor-descendant relationships of $\mathcal{T}'$, but might not preserve the most recent common ancestor relationships of $\mathcal{T}'$, which is required for the notion of perfectly displays. Consequently, perfectly displays is a stronger notion than ancestrally displays. Each of the rooted semi-labeled trees in Figure 7 are ancestrally displayed by the rooted semi-labeled tree in Figure 8. However, the first tree

in Figure 7 is not perfectly displayed by the rooted semi-labeled tree in Figure 8. In comparison with the standard notion of displays, ancestrally displays is stronger because the former does not preserve descendancy. A collection $\mathcal{P}$ of rooted semi-labeled trees is *ancestrally compatible* if there is a rooted semi-labeled tree $\mathcal{T}$ that ancestrally displays every tree in $\mathcal{P}$, in which case, $\mathcal{T}$ *ancestrally displays* $\mathcal{P}$.

In this section, we present a polynomial-time algorithm (called ANCESTRALBUILD) for solving the following problem.

Problem: HIGHER TAXA ANCESTOR COMPATIBILITY

Instance: A collection $\mathcal{P}$ of rooted semi-labeled trees.
Question: Does there exist a rooted semi-labeled tree that ancestrally displays $\mathcal{P}$ and, if so, can we construct such a rooted semi-labeled tree?

Before describing ANCESTRALBUILD and its subroutine DESCENDANT, we need first to define a particular graph and a construction. This graph consists of a mixture of arcs (directed edges) and edges. Let $\mathcal{P}$ be a collection of rooted fully-labeled trees. This graph, called the *descendancy graph of* $\mathcal{P}$ and denoted $D(\mathcal{P})$, is defined as follows. The vertex set of $D(\mathcal{P})$ is $\mathcal{L}(\mathcal{P})$. The arc set $A(\mathcal{P})$ of $D(\mathcal{P})$ is

$$\{(c, a) : c <_{\mathcal{T}} a \text{ for some } \mathcal{T} \text{ in } \mathcal{P}\},$$

and the edge set $E(\mathcal{P})$ of $D(\mathcal{P})$ is

$$\{\{a, b\} : a \text{ is not comparable to } b \text{ under } \leq_{\mathcal{T}} \text{ for some } \mathcal{T} \text{ in } \mathcal{P}\}.$$

As an example of a descendancy graph, let $\mathcal{P}$ be the collection of fully-labeled trees formed from the trees in Figure 7 by adding u_1 to the root of the leftmost tree, u_3 to the root and u_2 to the other unlabeled vertex of the middle tree, and u_4 to the unlabeled vertex of the rightmost tree. Figure 9 shows the descendancy graph of $\mathcal{P}$ where, to avoid clutter, only edges and arcs from parents to immediate descendants are shown.

The descendancy graph plays an important role in ANCESTRALBUILD. However, unlike SEMI-LABELEDBUILD, where a cluster and root-label graph is constructed at each iteration, the descendancy graph for $\mathcal{P}$ is constructed just once and then successive iterations consider particular restrictions of it. To this end, we will denote the subgraph of $D(\mathcal{P})$ that is induced by a subset S of the vertex set $\mathcal{L}(\mathcal{P})$ by $D(\mathcal{P}) \mid S$; that is, $D(\mathcal{P}) \mid S$ denotes the subgraph of $D(\mathcal{P})$ obtained by deleting all vertices of $\mathcal{L}(\mathcal{P}) - S$ and their incident arcs and edges. In association with $D(\mathcal{P})$ (or any of its vertex-induced subgraphs), the

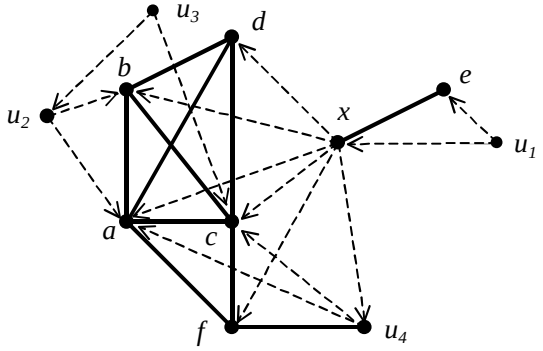

Figure 9. The descendancy graph of $\mathcal{P}$. Arcs are indicated by dashed lines, edges by solid lines.

in-degree of a vertex a is the number of arcs directed into a (edges are ignored), and an *arc component* is a connected component of the graph obtained by deleting all edges.

Lastly, we define our construction. Let $\mathcal{T} = (T; \phi)$ be a rooted semi-labeled tree. We say that a rooted semi-labeled tree $\mathcal{T}_1$ has been obtained from $\mathcal{T}$ by *adding descendants to leaves* if, for each multi-labeled leaf vertex u of T, we adjoin a new leaf vertex v to u by a new edge, and then label each new leaf vertex with a distinct new label. For a collection $\mathcal{P}$ of rooted semi-labeled trees, $\mathcal{P}_1$ has been obtained from $\mathcal{P}$ by *adding descendants to leaves* if it has been obtained by adding descendants to leaves to every tree in $\mathcal{P}$ so that all the new labels are distinct.

Algorithm: ANCESTRALBUILD($\mathcal{P}$, $\mathcal{T}$)

Input: Let $\mathcal{P}$ be a collection of rooted semi-labeled trees.
Output: A rooted semi-labeled tree $\mathcal{T}$ that ancestrally displays $\mathcal{P}$ or the statement $\mathcal{P}$ *is not ancestrally compatible.*

1. Construct a collection $\mathcal{P}'$ of rooted fully-labeled trees from $\mathcal{P}$ by adding descendants to leaves and then adding distinct new labels to the resulting collection.
2. Construct the descendancy graph $D(\mathcal{P}')$ of $\mathcal{P}'$.
3. Call the subroutine DESCENDANT($D(\mathcal{P}')$, v', $\mathcal{T}'$).
4. If DESCENDANT returns *no possible labeling*, then return $\mathcal{P}$ *is not ancestrally compatible.*
5. If DESCENDANT returns a rooted semi-labeled tree $\mathcal{T}'$, then remove the added labels and return the resulting rooted semi-labeled tree $\mathcal{T}$.

Algorithm: DESCENDANT($D(\mathcal{P}')$, v', $\mathcal{T}'$)

Input: The descendancy graph of a collection $\mathcal{P}'$ of rooted fully-labeled trees.
Output: A rooted fully-labeled tree $\mathcal{T}'$ with root vertex v' that ancestrally displays $\mathcal{P}'$ or the statement *no possible labeling.*

1. Let S_0 denote the set of vertices of $D(\mathcal{P}')$ that have in-degree zero and no incident edges.
2. If S_0 is empty, then halt and return *no possible labeling.*
3. Otherwise,
   (a) Delete the elements of S_0 (and their incident arcs) from $D(\mathcal{P}')$ and denote the resulting graph by $D(\mathcal{P}')\backslash S_0$.
   (b) Let $S_1, S_2, \ldots, S_k$ denote the vertex sets of the arc components of $D(\mathcal{P}')\backslash S_0$.
   (c) Delete all edges of $D(\mathcal{P}')\backslash S_0$ the end vertices of which are in distinct arc components of this graph.
   (d) For each element $i \in \{1, 2, \ldots, k\}$, call DESCENDANT($D(\mathcal{P}') \mid S_i$, v'_i, $\mathcal{T}'_i$). If DESCENDANT($D(\mathcal{P}') \mid S_i$, v'_i, $\mathcal{T}'_i$) returns a tree, then assign the labels in S_0 to v' and attach $\mathcal{T}'_i$ to v' via the edge $\{v'_i, v'\}$.

The general approach of the algorithm DESCENDANT is the same as that of FULLY-LABELEDBUILD. In particular, it attempts to construct a rooted fully-labeled tree that ancestrally displays $\mathcal{P}'$ beginning with the root and moving towards the leaves. To illustrate ANCESTRALBUILD, the rooted semi-labeled tree shown in Figure 8 is the result of applying this algorithm to the collection of rooted semi-labeled trees shown in Figure 7.

Remark.

1. Because DESCENDANT considers proper restrictions of $D(\mathcal{P})$ successively, it is clear that DESCENDANT returns either "no possible labeling" or a rooted semi-labeled tree. ANCESTRALBUILD consequently returns either "$\mathcal{P}$ is not ancestrally compatible" or a rooted semi-labeled tree.
2. Because every tree in $\mathcal{P}'$ is fully-labeled, it follows that the only labels in S_0 at any iteration are root labels of the corresponding restrictions of $\mathcal{P}'$. Thus, in regards to the last step of DESCENDANT, $\mathcal{P}' \mid S_i$ is a rooted fully-labeled tree for all i. This fact will be useful later.

Theorem 4.1. *Let $\mathcal{P}$ be a collection of rooted semi-labeled trees. Then* ANCESTRALBUILD *applied to $\mathcal{P}$ either:*

(i) *returns a rooted semi-labeled tree that ancestrally displays $\mathcal{P}$ if $\mathcal{P}$ is ancestrally compatible, or*

(ii) *returns the statement $\mathcal{P}$* is not ancestrally compatible *otherwise.*

The proof Theorem 4.1 makes use of the following lemma. The proof follows the approach used in the proof of Lemma 3.4 and is omitted.

Lemma 4.2. *Let $\mathcal{P}$ be a collection of rooted semi-labeled trees. Let $\mathcal{P}'$ be a set of rooted fully-labeled trees obtained from $\mathcal{P}$ by adding descendants to leaves and then adding distinct new labels to the resulting collection. Then $\mathcal{P}$ is ancestrally compatible if and only if $\mathcal{P}'$ is ancestrally compatible. Moreover, if $\mathcal{T}'$ is a rooted semi-labeled tree that ancestrally displays $\mathcal{P}'$, then $\mathcal{T}'$ ancestrally displays $\mathcal{P}$.*

Proof of Theorem 4.1. By Lemma 4.2, it suffices to show that the theorem holds if $\mathcal{P}$ is a collection of fully-labeled trees with no multi-labeled leaf vertices. Suppose that $\mathcal{P}$ is ancestrally compatible, and let $\mathcal{T}$ be a semi-labeled tree that ancestrally displays $\mathcal{P}$. We show that under this assumption, ANCESTRALBUILD applied to $\mathcal{P}$ outputs a rooted semi-labeled tree. Assume that this is not the case. Then, at some iteration of ANCESTRALBUILD, there is subset S of $\mathcal{L}(\mathcal{P})$ for which all vertices of $D(\mathcal{P})\,|\,S$ either have in-degree greater than zero or are incident with an edge. Because $\mathcal{T}$ ancestrally displays $\mathcal{P}$, it is seen easily that $\mathcal{T}\,|\,S$ ancestrally displays $\mathcal{P}\,|\,S$. Let P be a path of $\mathcal{T}\,|\,S$ from the root to a leaf and consider the first label, y say, that is met on this path. In $D(\mathcal{P})\,|\,S$, either y does not have in-degree zero or it is incident with an edge. In the first case, this implies that there is another element, x say, of S such that in some tree of $\mathcal{P}$ we have x is a proper ancestor of y. But y was the first label met in P, and so x is not a proper ancestor of y in $\mathcal{T}\,|\,S$ and, in particular, in $\mathcal{T}$; a contradiction. Therefore, we can assume that, in $D(\mathcal{P})\,|\,S$, y has in-degree zero and is incident with an edge. But then, because $\mathcal{P}\,|\,S$ is a collection of rooted fully-labeled trees (see remark above), all trees in $\mathcal{P}\,|\,S$ in which y is a label has y as a root label. This means that, in $D(\mathcal{P})\,|\,S$, y cannot be incident with any edge. This last contradiction completes this direction of the proof.

For the converse, suppose that ANCESTRALBUILD outputs a rooted semi-labeled tree $\mathcal{T}$. We show that $\mathcal{T}$ ancestrally displays $\mathcal{P}$. Let $\mathcal{T}_1$ be a member of $\mathcal{P}$, and let a and b be elements of $\mathcal{L}(\mathcal{P})$. If $a <_{\mathcal{T}_1} b$, then, because a is an element of an arc component, there is an arc from a to b in the associated descendancy graph. Because ANCESTRALBUILD returns $\mathcal{T}$, there must be some iteration at which a is an element of S_0, but b is a vertex of an arc

component of the graph obtained by deleting the elements of S_0 including a. It now follows by the description of the descendancy graph that $a <_{\mathcal{T}} b$.

Next assume that a is not comparable to b in $\mathcal{T}_1$. Then, in $D(\mathcal{P})$, the vertices a and b are joined by an edge. Because ANCESTRALBUILD outputs $\mathcal{T}$, this edge is deleted eventually, but not until a and b are in separate arc components of some restriction of $D(\mathcal{P})$. This implies that, in $\mathcal{T}$, there is a cluster in which a is an element and not b, and there is a cluster in which b is an element and not a. In other words, a is not comparable to b in $\mathcal{T}$.

Lastly, let X_1 denote the label set of $\mathcal{T}_1$. We complete the converse and thus the proof by showing that $\mathcal{T} \,|\, X_1$ refines $\mathcal{T}_1$. Let C_1 be a cluster of $\mathcal{T}_1$. It suffices to show that C_1 is a cluster of $\mathcal{T} \,|\, X_1$. Let X'_1 be the subset of X_1 that labels the vertex u of $\mathcal{T}_1$ corresponding to C_1. Because $\mathcal{T}_1$ is fully-labeled, X'_1 is non-empty. Either X'_1 consists of a single element or u is not a leaf vertex. In the first case, this element is comparable trivially with itself. In the second case, for all $a, b \in X'_1$, there is a label c of $\mathcal{T}_1$ such that $a <_{\mathcal{T}_1} c$ and $b <_{\mathcal{T}_1} c$, and so, by an earlier argument, $a <_{\mathcal{T}} c$ and $b <_{\mathcal{T}} c$. Hence, a is comparable with b in $\mathcal{T}$. Furthermore, the same arguments imply that, for all $y \in C_1 - X'_1$, for all $x \in X'_1$, and for all $z \in X_1 - C_1$, we have x is a proper ancestor of y in $\mathcal{T}$, and either z is a proper ancestor of x or x and z are not comparable in $\mathcal{T}$. It now follows that C_1 is a cluster of $\mathcal{T} \,|\, X_1$. □

A very similar analysis to that used to show that the running time of SEMI-LABELEDBUILD is polynomial in the size of $\mathcal{L}(\mathcal{P})$ shows that the running time of ANCESTRALBUILD is also polynomial in the size of $\mathcal{L}(\mathcal{P})$. We leave the details to the reader.

Final Remarks.

1. Some extensions of the problems described in this chapter are considered by Daniel in his Masters thesis (in prep.). One in particular is the following. In Figure 6, two of the interior vertices are multi-labeled. For a variety of reasons, such as the labels representing taxa of different levels or the labels representing different taxa of the same rank (e.g., genera), it might have been predetermined that it is not possible for two such labels to label the same vertex. In some cases, such as the former, one way to resolve the problem is to include an additional rooted semi-labeled tree in the input consisting of a root vertex and a leaf where the higher taxon labels the root vertex. However, for many cases, no such resolution is be possible. Hence, a desirable extension to the original problem of HIGHER TAXA COMPATIBILITY is to include a collection of pairs of labels in the instance and then ask the question of whether there exists a rooted semi-labeled tree that perfectly displays $\mathcal{P}$ and has the

property for any such pair $\{a, b\}$, a and b label distinct vertices. Surprisingly, Daniel shows that the resulting problem is NP-complete.

2. Both the algorithms described in this chapter are "all-or-nothing" algorithms. Each algorithm returns either a rooted semi-labeled tree with certain properties if one exists or a statement that there is no such tree. In practice, this limits the use of these algorithms. However, we believe that there is a MINCUTSUPERTREE-type approach (see Semple and Steel, 2000; Page, 2002) to resolving this limitation, so that the two algorithms will alway output a rooted semi-labeled tree.

Acknowledgements

We thank Olaf Bininda-Emonds, Sebastian Böcker, and Rod Page for their valuable comments. The first author was supported by the New Zealand Institute of Mathematics and its Applications funded programme *Phylogenetic Genomics*, and the second author was supported by the New Zealand Marsden Fund.

References

AHO, A. V., SAGIV, Y., SZYMANSKI, T. G., AND ULLMAN, J. D. 1981. Inferring a tree from lowest common ancestors with an application to the optimization of relational expressions. *SIAM Journal on Computing* 10:405–421.

BRYANT, D., SEMPLE, C., AND STEEL, M. 2004. Supertree methods for ancestral divergence dates and other applications. In O. R. P. Bininda-Emonds (ed). *Phylogenetic Supertrees: Combining Information to Reveal the Tree of Life*, pp. xxx–xxx. Kluwer Academic, Dordrecht, the Netherlands.

PAGE, R. D. M. 2002. Modified mincut supertrees. In R. Guigó and D. Gusfield (eds), *Algorithms in Bioinformatics, Second International Workshop, WABI 2002, Rome, Italy, September 17–21, 2002, Proceedings*, pp. 537–552. Springer, Berlin.

PAGE, R. D. M. 2004. Taxonomy, supertrees, and the Tree of Life. In O. R. P. Bininda-Emonds (ed). *Phylogenetic Supertrees: Combining Information to Reveal the Tree of Life*, pp. xxx–xxx. Kluwer Academic, Dordrecht, the Netherlands.

SEMPLE, C. AND STEEL, M. 2000. A supertree method for rooted trees. *Discrete Applied Mathematics* 105:147–158.

SEMPLE, C. AND STEEL, M. 2003. *Phylogenetics*. Oxford University Press, Oxford.