Chapter 4

Combinatorics of Trees II

Now that we've developed one method of phylogenetic inference, parsimony, we turn back to combinatorial considerations of phylogenetic trees. Our focus will be on different ways we can view the information expressed by a tree, and how those ways lead to methods of summarizing a collection of trees on the same taxon set in a single *consensus* tree. This is often done when parsimony returns many trees with the same minimal parsimony score, to summarize their common features. However, it is improtant in many other situations as well.

We also briefly introduce a similar situation where one has a collection of trees on possibly *different* but overlapping sets of taxa. Then the goal is to combine the trees into a single *supertree* that shows the relationships that occur in all the given trees, or if that is not possible attempts to show many of the relationships.

4.1 Splits and Clades

Suppose T is an unrooted phylogenetic X-tree. Then if we delete any one edge e of T, we partition the taxa X into two subsets, according to the two connected components of the resulting graph. For instance, deleting the marked edge in the tree T of Figure 4.1 produces the partition $\{a, b, d\}, \{c, e, f\}$, which we call a *split*.

Definition. A *split* of X is any partition of X into two non-empty disjoint subsets. We write $X_0|X_1$ to denote the split with subsets X_0, X_1.

If T is a phylogenetic X-tree, and $e \in E(T)$, then the *split induced by* e is the partition of X obtained by removing the edge e from T and forming the subsets of X that appear on each of the two connected components of the resulting graph. We also say such a split is *displayed on* T.

In the split notation $X_0|X_1$, the order of the sets does not matter; $X_0|X_1$ is the same as $X_1|X_0$. Also, if $X_0|X_1$ is a split, then $X_1 = X \setminus X_0$ is the complement of X_0. Thus to specify a split, we really only need to give one of

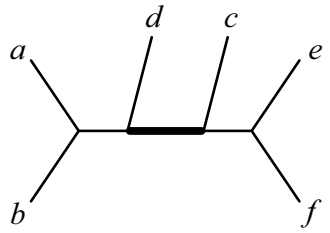

Figure 4.1: An edge inducing the split $\{a, b, d\}|\{c, e, f\}$.

the subsets. For instance, we might choose to give the smaller of the two, or the one that contains a particular taxon.

For a given set X of n taxa, there are $2^{n-1} - 1$ splits (see Exercise 2). However, a binary tree has only $2n - 3$ edges, and non-binary trees have even fewer. Thus at most $2n - 3$ splits will be displayed on any one tree.

A split induced by a pendant edge of a tree is particularly simple: In a tree relating n taxa, if a leaf is labelled by taxon S_1, then the split associated to that edge is $\{S_1\}|\{S_2, S_3, \ldots, S_n\}$. This split reflects nothing more about a tree than that S_1 labels a leaf. For that reason, a split $X_0|X_1$ in which one of the X_i has only a single element is said to be *trivial*. While trivial splits tell us essentially nothing interesting about a tree, the non-trivial splits induced by internal edges carry more information about the tree.

Suppose now that T is a phylogenetic X-tree and we consider the splits induced by two different edges e_1, e_2 of T. Removing both e_1 and e_2 from T produces three connected components, and thus a partition of X into three disjoint subsets X_1, X_2, X_3. With appropriate numbering of these sets, the split induced by e_1 is $X_1|X_2 \cup X_3$ and that induced by e_2 is $X_1 \cup X_2|X_3$. This shows splits induced from edges of a phylogenetic X-tree will have the following pairwise property.

Definition. Two splits $Y_0|Y_1$ and $Y_0'|Y_1'$ of a set Y are said to be *compatible* if for some i, j we have $Y_i \cap Y_j' = \emptyset$.

It's not hard to convince yourself that if you were given a pair of compatible splits, then you could find a tree displaying both of them. But what if we have a larger collection of splits? More broadly, to what extent does a collection of pairwise compatible X-splits correspond to a phylogenetic tree?

To answer this question, we need a slight generalization of a phylogenetic tree. While binary phylogenetic trees remain the biologists' ideal, sometimes the best tree we can produce has higher-degree vertices, some labels on internal vertices (which you might think of as leaves which have not yet been resolved from the interior of the tree, and multiple labels on some vertices (which you might also think of as a result of not yet resolving the branching of all taxa from one another). All that we will need is captured in the following definition.

Definition. An *X-tree* is a tree T together with a labeling map $\phi : X \to V(T)$ such that for each $v \in V(T)$ of degree 1 or 2 is labelled; *i.e.*, $\phi^{-1}(v) \neq \emptyset$ for all v of degree at most 2.

Obviously we can refer to splits of X induced by edges of an X-tree, just as for phylogenetic trees. Moreover, any two splits induced by edges in a single X-tree will be compatible.

We now establish an important theorem giving the relationship between compatible splits and trees.

Theorem 9. (Splits Equivalence Theorem) Let S be a collection of splits of X. Then there is an X-tree whose induced splits are precisely those of S if, and only if, the splits in S are pairwise compatible. Furthermore, this X-tree is unique, up to isomorphism.

Proof. That an X-tree induces pairwise compatible splits has been discussed, so suppose we have a set of pairwise compatible splits S and wish to construct such an X-tree. We proceed by induction on the size of S. The case $|S| = 1$ is straightforward, since a one-edge tree whose two vertices are labeled by the taxa in the two split sets has the desired property, and no other X-tree does.

Let $S = S' \cup \{X_0|X_1\}$ where $X_0|X_1 \notin S'$. Then by induction, there is an X-tree T' whose induced splits are precisely those of S'. Color red all those vertices of T' that are labeled by elements of X_0, and color blue all those labeled by elements of X_1, so every vertex is either uncolored, red, blue, or both red and blue.

Our first goal is to show there is a unique vertex v in T' whose removal results in connected components all of whose colored vertices have the same color. To do this, let T_1 be the minimal spanning tree of all blue vertices in T', and T_2 the minimal spanning tree of all red vertices in T'.

Observe that T_1 and T_2 cannot have an edge in common since if they did that edge of T' would induce a split not compatible with $X_0|X_1$: There would be both red and blue vertices in both components of the graph left by the edge's removal. However, if T_1 and T_2 are disjoint then they are joined by some path in T', but picking any edge on that path induces the split $X_0|X_1$, and since $X_0|X_1 \notin S'$ we have a contradiction. Thus T_1 and T_2 must have only vertices in common. In fact, they can have only one vertex in common, since having more vertices in common would imply they have edges in common, and that has been ruled out. Call this vertex v.

Note that when v is removed from T in each of the resulting connected components all colored vertices have the same color. Indeed, if there were a component with both red and blue vertices in it, then the red and blue minimal spanning trees would have in common the edge from v leading into this component, and we know a shared edge is impossible.

Furthermore v is the unique vertex with this property, since removing any other vertex will leave all of the minimal spanning tree of one color in a single component, and at least part (and hence some colored nodes) of the other in the same one.

Now we modify T' to get T by replacing v by two vertices v_0 and v_1 connected by an edge e, reconnecting the edges of T' incident to v so that red components join to v_0 and blue components to v_1. We modify the labeling map for T' to get a labeling map for T by reassigning all of v's labels from X_0 to v_0, and all of v's labels from X_1 to v_1. We now see that the new edge e of T induces the split $X_0|X_1$, while all other edges of T induce precisely the splits of S'.

That T is uniquely determined by S we leave as Exercise 7. □

Notice that the argument given in this proof can easily be formulated as an algorithm, called *Tree-Popping*, for constructing an X-tree from a collection of splits. To illustrate this, we show only one step which might be done in the middle of a longer process. Suppose the graph on the left of Figure 4.2 has already been constructed from some splits, each of which was compatible with the split $\{de\}|\{abcf\}$. Then we color the vertex labeled de red, and the other

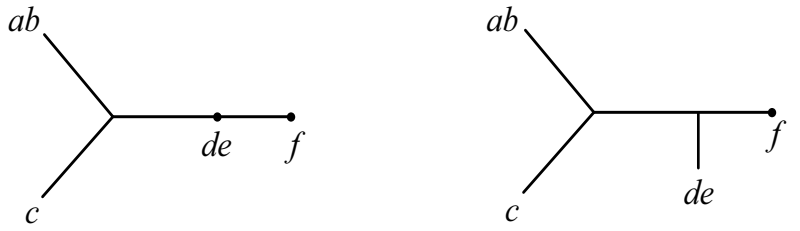

Figure 4.2: Incorporating the split $\{de\}|\{abcf\}$ into a tree following the proof of Theorem 9.

labeled vertices blue. If there were a red-blue vertex, that would have to be the node we seek, since it would lie on both minimal spanning trees. But since there is no red-blue vertex, to distinguish the vertex v that will be replaced, we must find the minimal spanning trees of the vertices of each color. For red, this just a single vertex, while for the blue, it is the entire tree. These intersect at the vertex labeled de. We thus remove this vertex, replacing it by an edge with all red vertices joined at one end, and all blue at the other. This gives the tree on the right of Figure 4.2.

It is instructive to work through a more complete example, such as in Exercise 9.

We also note that if the collection of splits includes all trivial ones (which, by Exercise 3, are always compatible with the other splits), then Tree Popping will produce a phylogenetic X-tree, where each of the taxa is located at a distinct leaf.

The notion of a split is useful for unrooted trees. Its analog for rooted trees is that of a *clade*:

Definition. If X is a set of taxa, then a *clade* is simply a non-empty subset of X.

If T^ρ is a rooted phylogenetic X- tree, and $v \in V(T)$, then the subset $U \subseteq X$ of all taxa descended from v on T^ρ is called the *clade induced by* v. We also say the clade U is *displayed* on T^ρ.

The *trivial* clades are those with only one element (which are induced by a vertex that is a leaf of the tree), and the full set X (which is induced by the root of the tree). Clades are sometimes called *clusters*, or if they are induced by a vertex on a rooted tree, *monophyletic groups*.

There is a simple relationship between clades on rooted trees and splits on unrooted trees, provided we are careful about how we relate the rooted and unrooted trees: Given an n-taxon rooted tree T^ρ, create an $(n+1)$-taxon unrooted tree $\tilde{T}$ by attaching an additional edge at the root, and labeling the new leaf this introduces with a new taxon name, say r. (If the tree was originally rooted by the use of an outgroup which was then deleted, this process simply reintroduces the outgroup.) Then each clade U displayed on T^ρ determines a split $U|V$ on $\tilde{T}$ with $V = X \cup \{r\} \smallsetminus U$. Conversely, each split $U|V$ on $\tilde{T}$ determines a clade on T^ρ, by simply choosing which ever of the sets U, V does not contain r. We will take advantage of this fact in discussing consensus trees, since a method based on using splits from unrooted trees will automatically give a method using clades on rooted trees.

Notice that this correspondence between splits and clades requires that we change the number of taxa. We are *not* simply ignoring the root location in passing from a rooted tree to an unrooted one, but rather encoding it in a special way. This has implications for the notion of a consensus tree below: ignoring the root location and constructing a consensus tree using splits may give a different result than constructing a consensus tree using clades (encoded as splits on a larger taxon set) and then ignoring the root. (See Exercise 18.)

The notion of compatibility of splits carries over to clades, capturing the ability of two clades to be displayed on the same tree:

Definition. Two clades U_1, U_2 of X are said to be *compatible* if either $U_1 \subseteq U_2$, $U_2 \subseteq U_1$ or $U_1 \cap U_2 = \emptyset$.

4.2 Refinements and Consensus Trees

To illustrate the utility of the splits viewpoint on trees, we give two applications.

First, we can use Theorem 9 to determine when two X-trees can be 'combined' into a more refined X-tree. This is useful in a biological setting if we have several non-binary X-trees that we've inferred, perhaps from different data sets, that have managed to resolve different parts of some unknown binary tree. We want to combine the information they contain to give a tree that shows more resolution. For example, in Figure 4.3, we see two trees which have a common minimal refinement, in a sense that can be made precise through splits.

More formally, we say a tree T_2 is a *refinement* of T_1 if every split displayed by T_1 is also displayed by T_2.

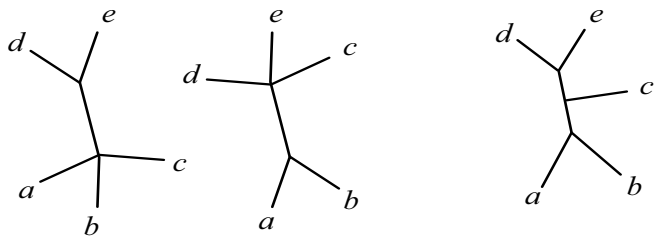

Figure 4.3: Two trees and their common refinement

By Theorem 9 we easily see the existence of a common refinement of two (or more) trees is equivalent to the compatibility of the splits of the two trees.

Corollary 10. Suppose $T_1, T_2, \ldots, T_n$ are X-trees. Then all induced edge splits from these trees are pairwise compatible if, and only if, there exists an X-tree T whose displayed splits are precisely those of the T_i. Furthermore, if this X-tree T exists, it is unique.

In the example of Figure 4.3, one could list the splits from the trees on the left (there is only one non-trivial split for each tree), check that they are pairwise compatible, and then apply the Tree Popping algorithm to produce their common refinement on the right. Of course this example is small, and the correct common refinement is clear without being so formal, but this gives a procedure that will produce common refinements in more complicated settings, with many larger trees.

A more common situation, though, is to have a collection of X-trees which have some incompatible splits, which therefore can not all be displayed on a single tree. Then we might want to combine them into a single *consensus tree* that only shows features common to all or most, and somehow ignores the incompatibilities. For instance, this is frequently done when parsimony has been used to infer trees, since many trees may tie for most parsimonious. Other situations where this may be desirable include when different genes have been used to infer highly resolved trees for the same set of taxa, but the trees are not in complete agreement.

There are several different versions of what should be called a consensus tree. We define these only for unrooted trees, using splits. For analogous versions using clades, one simply introduces an extra taxon to encode the root, as discussed earlier.

Definition. Suppose X-trees $T_1, T_2, \ldots, T_n$ are given. Then the *strict consensus tree* is the tree displaying precisely the splits that are displayed on every T_i. The *majority-rule consensus tree* is the tree displaying precisely the splits displayed on more than half of the T_i.

Note that the splits displayed on all T_i must be pairwise compatible since every pair is displayed on a single tree (in fact, on any of the T_i). Thus by

the Splits Equivalence Theorem, a strict consensus tree exists and is easily constructed by Tree Popping. Since the splits we use are a subset of those displayed on any of the T_i, the result will be a tree that has every T_i as a refinement. In practice, a strict consensus tree is often very poorly resolved; if there is a single T_i not displaying a particular split, then that split is not displayed by the strict consensus tree. The strict consensus approach effectively gives every tree veto power over the splits shown on the others.

The majority-rule consensus tree is defined to weaken the requirement of complete agreement. The 50% cutoff for a split 'winning' the vote is due to the following.

Theorem 11. Any two splits that are each displayed on more than 50% of the trees in a collection must be pairwise compatible.

Proof. If two splits each occur on more than half the trees, then, by the pigeonhole principle, they must both occur on at least one tree. But splits occurring on the same tree are compatible. □

Because of this theorem, we know the splits occurring with frequency above 50% across the trees are pairwise compatible, and thus by the Splits Equivalence Theorem they determine a tree. Thus the majority-rule tree actually exists, and can be found by Tree Popping.

Another possibility is the *loose consensus tree*. Of the splits displayed on any of the given trees, it displays exactly those that are compatible with all the trees. It will therefore display every split that the strict consensus tree does, but often some more, and thus be a refinement of it. It need not be a refinement of the majority-rule consensus tree, though, since a split displayed on most trees that is incompatible with a single tree will not be displayed on the loose consensus tree, but will be on the majority-rule one. This is used less commonly than majority rule, perhaps because a single 'error' in one tree can effectively veto a correct split in all the others.

One could also consider consensus trees between the strict and majority rule levels, by choosing a parameter q between $1/2$ and 1. Taking all the splits displayed on more than the fraction q of the trees, we again obtain a pairwise compatible set, and thus determine a unique tree displaying precisely those splits.

It's also possible to consider cut-offs lower than $1/2$, but then we cannot be sure all such splits will be compatible. To get around this issue, one can rank the splits displayed on all the trees, from highest to lowest, by the number of trees on which they appear. We then accept the first split in the list. Proceeding down the list, we accept each split that is compatible with all those previously accepted. If we continue this process through the full list, and then construct a tree displaying all accepted splits, we have a *greedy consensus tree*. The motivation behind this is that, once we pass below the 50% frequency level, we are using compatibility with the more frequent splits to help us choose which of the less frequent splits we should display.

One issue with a greedy consensus tree is that we may have several splits with equal ranking, and then the list ordering is chosen arbitrarily. A different ordering can then lead to acceptance of a different collection of splits, and hence to a different greedy consensus tree. Software implementations should at least warn of this situation, though not all do so.

There are other approaches to consensus trees, that seek to resolve conflicts using different criteria. One of these (MRP) we discuss below in the context of supertrees, though it is straightforward to use it unchanged to construct a consensus tree. Another is the *Adams consensus tree*, which exists only for rooted trees. It shows the 'nesting' of groups of taxa within others that appears in the clade structure of the input trees. Its construction should be explained in a later version of these notes.

4.3 Quartets

A possible approach to determining a phylogenetic tree for a large number of taxa is to try to determine trees for smaller subsets of taxa, and then piece these together. For this, we'll use unrooted trees, since the trees relating these smaller sets might not all contain a common vertex.

If we focus on small subsets, then *quartets* of 4 taxa are the smallest ones we should consider, as an unrooted tree relating 3 taxa gives us no information on a tree topology.

Definition. A *quartet tree* is a unrooted binary tree with 4 labeled leaves. We denote the tree by $ab|cd$ if it induces the split $\{a,b\}|\{c,d\}$.

Now any phylogenetic X-tree T induces a collection $\mathcal{Q}(T)$ of quartet trees:

$$\mathcal{Q}(T) = \{ab|cd \ : \text{ for some } X_0|X_1 \text{ displayed by } T,\ a,b \in X_0 \text{ and } c,d \in X_1\}.$$

If T is binary, then for every 4-element subset $\{a,b,c,d\}$ of X, exactly one of the quartets $ab|cd$, $ac|bd$, or $ad|bc$ is in $\mathcal{Q}(T)$, so $\mathcal{Q}(T)$ has $\binom{|X|}{4}$ elements. For non-binary $|X|$, of course, it has fewer.

If using quartets to deduce phylogenies of larger collections of taxa is to have a chance of working, we must have the following.

Theorem 12. The collection $\mathcal{Q}(T)$ determines T for a binary tree.

Proof. If $|X| \leq 4$ the result is clear, and so we proceed by induction.

If $X = \{a_1, \ldots, a_n\}$, let $\mathcal{Q}'$ be those quartets in $\mathcal{Q}(T)$ that only involve taxa in $X' = \{a_1, a_2, \ldots, a_{n-1}\}$. Then $\mathcal{Q}' = \mathcal{Q}(T')$ for a tree T' obtained from T in the obvious way, by deleting a_n and its incident edge, and conjoining the two other edges that edge meets, to get a binary tree. Thus $\mathcal{Q}'$ determines T' by the induction hypothesis.

To determine T, we need only determine the conjoined edge of T' to which the edge leading to a_n should attach. So considering each edge e of T' in turn, suppose e induces the split $X'_0|X'_1$ of X'. Assuming this split is non-trivial, so

each of the sets in the split has at least two elements, then if for all $a, b \in X_0'$ and $c, d \in X_1'$ both $ab|ca_n$ and $aa_n|cd$ are in $\mathcal{Q}(T)$, e is the edge we seek. If the split $X_0'|X_1'$ is trivial, then whether an edge leading to a_n should be attached along e can be determined by a similar test, the formulation of which we leave as Exercise 13. □

This theorem can be extended to non-binary trees as well.

As you'll see in the exercises, a subset of $\mathcal{Q}(T)$ may be enough to identify T. In fact only $|X| - 3$ well-chosen quartets are needed, though not every collection of quartets of that size is sufficient, and precisely what quartets are needed depends on the tree.

Of course the real difficulty with determining a phylogenetic tree from quartets is usually not the issue of having insufficiently many quartets. Rather, whatever inference method is used to infer quartets is likely to give some wrong results, and so produce a set of quartets that are incompatible. What is needed is a construction of a tree that somehow reconciles as much information as possible from inconsistent sources. Such *quartet methods* for tree construction have been explored in a number of research papers in recent years.

4.4 Supertrees

Assembling a tree from quartets is an example of a more general problem of constructing a *supertree*. We might have obtained a collection of different trees, with overlapping but different sets of taxa, which we wish to combine into a single tree. Doing this might enable us to see relationships between species which do not appear together on any of the individual trees.

Though there are a number of different approaches, here we describe only one, called Matrix Representation with Parsimony (MRP).

First, we need to extend the idea of parsimony to encompass characters with missing state information.

Suppose we only know the states some character assigns to *some* of the taxa in X. For the remaining taxa, we have no information on what the state is, though we believe there is some state. Then in a parsimony analysis we can treat these taxa for which we have no information as we treat internal vertices on the tree. We consider extensions of the character from the subset of taxa with known states to all internal vertices and taxa with unknown states. The standard definitions of parsimony scores can then be used to choose the most parsimonious tree(s).

If the Fitch-Hartigan algorithm is used to compute parsimony scores for trees, then only a simple modification needs to be made to deal with missing information. If the character has state space S, then to any taxon with missing information we assign the full set S. Taxa for which the state is known are assigned that state as usual. Then one works up the tree from the leaves to the root, assigning sets of states to each vertex and counting changes using the

same rules as when no information is missing. (A similar modification can be made for the Sankoff algorithm. See Exercise 22 of section 3.8.)

Now given a collection of trees $\{T_i\}$, where T_i is a phylogenetic X_i-tree and the taxon sets X_i may be different but overlapping, let $X = \cup X_i$. For each split displayed on each T_i, define a two-state character on X_i by assigning states in accord with the partition sets. For instance, if an edge e of T_i induces a split $X_{i,1}|X_{i,2}$ of X_i, we might define the character $\chi_{T_i,e}$ to assign state 1 to those taxa in $X_{i,1}$ and the state 0 to those in $X_{i,2}$. For any other taxa in X (*i.e.*, for those not on T_i), we treat the character value as missing. The MRP supertree is then the maximum parsimony tree for this character sequence encoding all the splits on all the T_i. This method thus seeks to find trees in which each split on each input tree is displayed (or come close to being displayed) when one ignores the taxa not on that input tree.

Of course as with any use of parsimony, the MRP tree may not be unique. If there are multiple trees that are most parsimonious, it is common to take a consensus tree of those.

The terminology 'Matrix Representation with Parsimony' may seem natural to a biologist, but strange to a mathematician. It comes from the use of the word 'matrix' to describe character information that has been placed in a table, with rows corresponding to taxa, columns to characters, and entries specifying states, and has little to do with a mathematicians notion of a matrix. In MRP, the trees we wish to combine are 'represented' by a matrix encoding characters corresponding to their splits.

4.5 Final Comments

All the concepts and methods discussed in this section have been combinatorial in nature. Even though the problem of reconciling different trees, or summarizing them in a single tree may arise naturally in a biological study, we have used no biological ideas or models in developing consensus and supertree methods. This should be kept firmly in mind when these methods are used. They may be reasonable in some circumstances, but not in others.

As an example, the gene tree/species tree problem of using many gene trees to infer a single species tree is one that can be approached with a biologically-motivated model. In that setting the performance of various consensus methods can be studied, and detailed results on the extent to which various consensus methods are reliable can be obtained. We'll return to this in a later chapter.

For now though, we have given no justification that in any particular biological setting it makes good sense to construct a consensus tree. Even though it may be common to report a consensus tree when parsimony leads to multiple most-parsimonious trees, one may question how this procedure actually fits with the parsimony criterion: a consensus tree of several most-parsimonious trees may in fact require *more* state changes, and thus not be most-parsimonious

itself. This does not mean that such a consensus tree should not be considered, but just that one should understand the limitations.

Finally, Bryant [Bry03] gives an excellent survey on consensus trees, which includes many more types than have been presented here. More material on supertrees can be found in [SS03].

4.6 Exercises

1. List all the splits displayed on the tree in Figure 4.1, including trivial ones.
2. Explain why a set of n taxa will have $2^{n-1} - 1$ splits.
3. Explain why a trivial split of X is compatible with any other split.
4. Show that if distinct splits $Y_0|Y_1$ and $Y_0'|Y_1'$ of a set Y are compatible then there is exactly one pair i, j with $Y_i \cap Y_j' = \emptyset$.
5. Show that two splits $Y_0|Y_1$ and $Y_0'|Y_1'$ of a set Y are compatible if and only if $Y_i \subseteq Y_j'$ for some i, j. Further show that $Y_i \subseteq Y_j'$ if and only if $Y_{1-i} \supseteq Y_{1-j}'$.
6. Problem Deleted.
7. Complete the proof of Theorem 9 by showing a tree T which displays exactly those splits in some collection S is in fact unique. (Hint: Use induction on the number of splits. Assume there are two such trees, and 'contract' the edge in each for one specific split.)
8. Elaborate on the given proof of Theorem 9 to explain why the T we construct has the X-tree property that all vertices of degree ≤ 2 are labeled.
9. For $X = \{a, b, c, d, e, f, g\}$ consider the pairwise compatible splits

$$a|bcdefg, eg|abcdf, b|acdefg, af|bcdeg, f|abcdeg.$$

   a. By Tree Popping, find an X-tree inducing precisely these splits.

   b. Use Tree Popping again, but with the splits in some other order, to again find the same tree.

10. List all the splits displayed on the two trees on the left of Figure 4.3, and then use Tree Popping to produce the tree on the right from them.
11. If T_1, T_2 are X-trees with T_2 a refinement of T_1, write $T_1 \leq T_2$. Show that '$\leq$' is a partial order on the set of X-trees. Then characterize X-trees that are maximal with respect to '$\leq$'.
12. List all 15 quartets displayed on the tree of Figure 4.1.
13. Complete the proof of Theorem 12, by addressing the issue in its final line.

14. A collection of quartets is said to be *compatible* if there is some tree T that displays them all.

    (a) Show that the quartets $ab|cd$ and $ac|be$ are compatible.

    (b) Explain why the quartets $ab|cd$, $ac|be$, and $bc|de$ are not compatible. (Note: every pair of these are compatible, so pairwise compatibility of quartets is not enough to show they can all be displayed on a single tree.)

15. For a 5-leaf binary phylogenetic X-tree T, $\mathcal{Q}(T)$ has 5 elements. Draw such a 5-leaf tree and give a set of only two of these quartets that still determine T. Give another set of two of these quartets that does not determine T.

16. Generalize the result of the last problem by showing that for any binary phylogenetic X-tree, there are $|X| - 3$ quartets that determine T.

17. Show that the definition of compatibility of clades given in the text is consistent with that for compatibility of splits, using the relationship of n-taxon rooted trees to $(n+1)$-taxon unrooted ones. (You may find it helpful to use the result in Exercise 4.)

18. Consider the three trees $(a,(b,(c,d)))$, $((a,b),(c,d))$, and $(((a,b),c),d)$ on taxa $X = \{a,b,c,d\}$.

    (a) Construct the strict consensus tree, treating these as rooted.

    (b) Construct the strict consensus tree, treating these as unrooted 4-taxon trees. Is your answer the same as the unrooted version of the tree from (a)?

    (c) Construct the majority-rule consensus tree, treating these as rooted.

19. Explain why the greedy consensus tree is a refinement of the loose consensus tree.

20. If an unrooted binary tree relates n taxa, how many quartets will it display? How many possible quartets (not necessarily compatible) are there for n taxa?

21. Consider the three trees shown below.

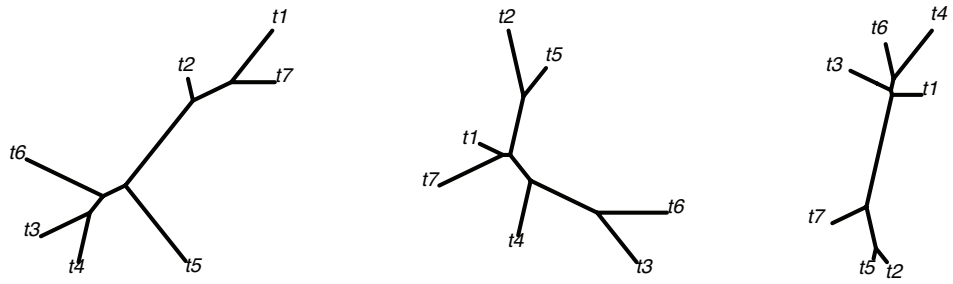

Figure 4.4: Trees T_1, T_2, T_3.

(a) For each of the three trees, list all splits displayed on the tree.

(b) Construct the strict consensus tree.

(c) Construct the majority rule consensus tree.