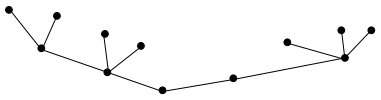
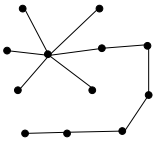
Trees

1 Trees defined

There are several ways to define trees. First, we can consider (undirected) trees to be specialized from (undirected) graphs. Thus a *tree* is a connected graph with no circuits. The following are trees:

trees as specialized graphs





Theorem 1 *A tree is simple.*

Proof. Suppose $G = (V, E)$ is a tree.

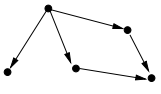
First suppose that G has a self-loop $e \in E$, and let v the endpoint of e. Then G has the walk v, e, v, which is a circuit. Thus G is not a tree, contradiction. Hence G has no self-loops.

Next suppose that G has parallel edges $e_1, e_2 \in E$. Let v_1 and v_2 be the endpoints of e_1 and e_2. Then G has the walk v_1, e_1, v_2, e_2, v_1, which is a circuit. Thus G is not a tree, contradiction. Hence G has no parallel edges.

Therefore G is simple.□

More familiar and intuitive to us is a *rooted tree*, which is a tree with a designated vertex as a root. Look at the trees pictured above and imagine that each edge is attached to its vertices with a hinge or something flexible. Convince yourself that any vertex could be the root by imagining picking up the trees by any vertex and letting the edges fall into the familiar tree shape.

Notice that this approach to defining trees does not work on *directed* graphs and trees because it would allow a directed acyclic graph like



directed trees as specialized directed graphs

Thus, a directed graph $G = (V, E)$ is a *tree* if there exists a vertex $v_0 \in V$ such that for all $v \in V$, there exists exactly one path from v_0 to v in G. Such a v_0 is called a *root* of the tree.

Theorem 2 *A directed tree has exactly one root.*

> **Proof.** Suppose $G = (V, E)$ is a directed tree, and let $v_0 \in E$ be a root. Further suppose $v_1 \in E$, $v_1 \neq v_0$, is also a root. By definition of tree, there exists a path p_0 from v_0 to v_1. Likewise there exists a path p_1 from v_1 to v_0. Let p_3 be the concatenation of p_0 and p_1. p_3 is a path from v_0 to itself. v_0 alone is also a (trivial) path to itself. Hence there exist two paths from v_0 to v_0. Contradiction. □

Hand-waving alert: The above proof never shows that p_3 is itself a path. It can be shown, but it's a pain.

Trees as relations

Since trees are simple, they have no parallel edges. A directed graph with no parallel edges can be viewed as a set (the vertices) and a relation (the edges). Thus we can consider trees to be a kind of relation. When we say "T is a tree on a set A," we mean that A is the set of vertices, and T is the set of edges, that is, T is a relation.

Theorem 3 *If T is a tree on a set A, then T is irreflexive, asymmetric, and untransitive.*

> **Proof.** (T is irreflexive.) Suppose $a \in A$, and further suppose $(a, a) \in T$. Let v_0 be the root of T, and let p be the path from v_0 to a, which must exist by definition of tree. If p contains the edge (a, a), then the path p' which is constructed by removing the edge (a, a) from p' is also a path from v_0 to a. If p does not contain the edge (a, a), then the path p'' which is constructed by appending the edge (a, a) to the end of p is also a path from v_0 to a. Hence there are two paths in T from v_0 to a, and T is not a tree. Contradiction.
>
> (T is asymmetric.) Suppose $a, b \in A$, and $(a, b), (b, a) \in T$. Let v_0 be the root of T, and let p be the path in from v_0 to b, which must exist by definition of tree.
>
> Case 1: Suppose p contains the edge (a, b). Then it contains a subpath p' from v_0 to a. The path p'' which is constructed by appending (b, a) to the end of p is also a path from v_0 to a.
>
> Case 2: Suppose p does not contain the edge (a, b). Then the path p''' which is constructed by appending $(b, a)(a, b)$ to the end of p is also a path from v_0 to b.
>
> Either way there is more than one path from v_0 to another vertex, and so T is not a tree. Contradiction.
>
> (T is untransitive.) See exercise.

Trees defined recursively

Finally, we can define (rooted) trees recursively. Thus a *rooted tree* is

- a single vertex (the root) with no edges, or
- a vertex (the root) with edges to roots of rooted trees.

Formally,

- $((\{v\}, \{\}), v)$, or
- $((\{v\} \cup V_1 \cup V_2 \cup \ldots \cup V_n, \{(v, v_1), (v, v_2), \ldots, (v, v_n)\} \cup E_1 \cup E_2 \cup \ldots E_n), v)$, where $((V_1, E_1), v_1)$, $((V_2, E_2), v_2)$, …, $((V_n, E_n), v_n)$ are all rooted trees.

A recursive definition, like a recursive algorithm, has a base case and a recursive case.

2 Trees represented

The representation of trees in a programming language is based on the recursive definition. This is the case no matter what language or paradigm one uses (assuming, of course, the language supports structural recursion). This can be done particularly succinctly, however, in ML. If we distinguish between leaf and internal vertices, and store information only at the leaves, we can write

```
- datatype tree = Leaf of int | Internal of tree list;
```

If we want information stored at internal vertices/nodes, too, then we can eliminate the distinction, since the list of children can be empty.

```
- datatype tree = Node of (int * tree list);
```

If it's the familiar binary tree in which we're interested, then we can change the tree list into a pair of two trees, but we'd also need the special case for a null link.

```
- datatype tree = Null | Node of (int * tree * tree);
```

In a *full* binary tree (not the same as a *complete* binary tree), each node has either 0 or two children, never 1. It is modeled with the slight variation

```
- datatype tree = Leaf of int | Internal of (int * tree * tree);
```

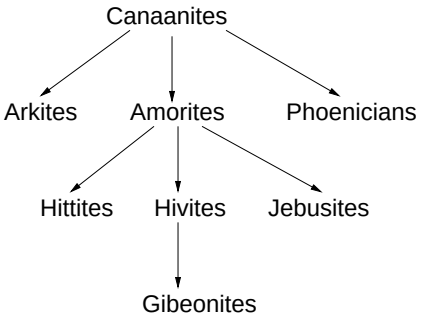
Of course, trees can have more specialized purposes

```
- datatype tree = Animal of string | Question of (string * tree * tree);
```

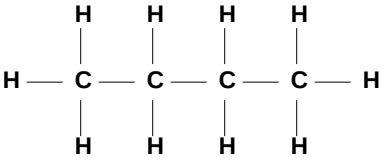
3 Trees applied

Trees have many purposes for structuring information in many fields. There are knowledge or decision trees as in the last example of the previous section.

Trees are used to show hierarchical relationships such as family trees, organizational charts, or subset relations.[1]

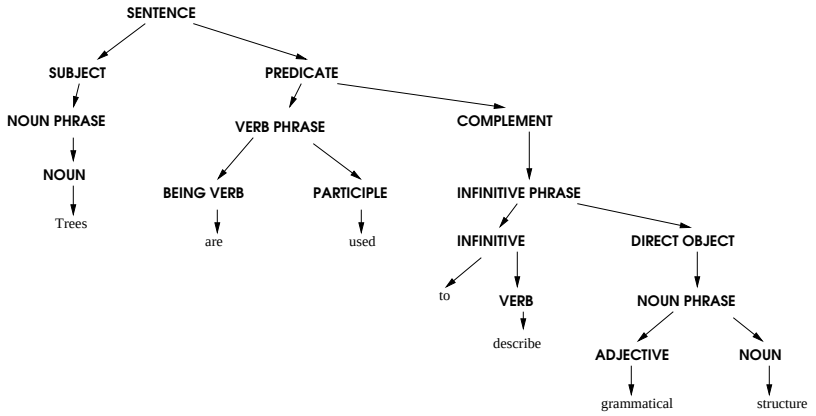


Trees can describe molecular structure



[1]This tree shows a possible scheme for what ethnic groups in Canaan were subgroups of others. This is inferred from various pieces of data in the Bible. For example, Joshua 9:7 implies that the Gibeonites were a specific tribe within the Hivites.

Trees are used to describe grammatical structure.



Exercise. Show that a tree, viewed as a relation, is untransitive. That is, let T is a tree on a set A. Show that for all $a, b, c \in A$, if $(a, b) \in T$ and $(b, c) \in T$, then $(a, c) \notin T$.