together. This is the usual role for subgraphs and typically specifies semantic information about the graph components. It can also provide a convenient shorthand for edges. An edge statement allows a subgraph on both the left and right sides of the edge operator. When this occurs, an edge is created from every node on the left to every node on the right. For example, the specification

```
A -> {B C}
```

is equivalent to

```
A -> B
A -> C
```

In the second role, a subgraph can provide a context for setting attributes. For example, a subgraph could specify that blue is the default color for all nodes defined in it. In the context of graph drawing, a more interesting example is:

is equivalent to

```
subgraph {
    rank = same; A; B; C;
}
```

This (anonymous) subgraph specifies that the nodes A, B and C should all be placed on the same rank if drawn using dot.

The third role for subgraphs directly involves how the graph will be laid out by certain layout engines. If the name of the subgraph begins with cluster, Graphviz notes the subgraph as a special cluster subgraph. If supported, the layout engine will do the layout so that the nodes belonging to the cluster are drawn together, with the entire drawing of the cluster contained within a bounding rectangle. Note that, for good and bad, cluster subgraphs are not part of the DOT language, but solely a syntactic convention adhered to by certain of the layout engines.

3.2.2 Lexical and Semantic Notes

A graph must be specified as either a *digraph* or a *graph*. Semantically, this indicates whether or not there is a natural direction from one of the edge's nodes to the other.

Lexically, a digraph must specify an edge using the edge operator -> while a undirected graph must use --. Operationally, the distinction is used to define different default rendering attributes. For example, edges in a digraph will be drawn, by default, with an arrowhead pointing to the head node. For ordinary graphs, edges are drawn without any arrowheads by default.

A graph may also be described as *strict*. This forbids the creation of multi-edges, i.e., there can be at most one edge with a given tail node and head node in the directed case. For undirected graphs, there can be at most one edge connected to the same two nodes. Subsequent edge statements using the same two nodes will identify the edge with the previously defined one and apply any attributes given in the edge statement. For example, the graph

```
strict graph {
  a -- b
  a -- b
  b -- a [color=blue]
}
```

will have a single edge connecting nodes a and b, whose color is blue.

If a default attribute is defined using a *node*, *edge*, or *graph* statement, or by an attribute assignment not attached to a node or edge, any object of the appropriate type defined afterwards will inherit this attribute value. This holds until the default attribute is set to a new value, from which point the new value is used. Objects defined before a default attribute is set will have an empty string value attached to the attribute once the default attribute definition is made.

A subgraph receives the attribute settings of its parent graph at the time of its definition. This can be useful; for example, one can assign a font to the root graph and all subgraphs will also use the font. For some attributes, however, this property is undesirable. If one attaches a label to the root graph, it is probably not the desired effect to have the label used by all subgraphs. Rather than listing the graph attribute at the top of the graph, and the resetting the attribute as needed in the subgraphs, one can simply defer the attribute definition in the graph until the appropriate subgraphs have been defined.

If an edge belongs to a cluster, its endpoints belong to that cluster. Thus, where you put an edge can effect a layout, as clusters are sometimes laid out recursively.

There are certain restrictions on subgraphs and clusters. First, at present, the names of a graph and its subgraphs share the same namespace. Thus, each subgraph must have a unique name. Second, although nodes can belong to any number of subgraphs, it is assumed clusters form a strict hierarchy when viewed as subsets of nodes and edges.