DOT language

From Wikipedia, the free encyclopedia

DOT is a plain text graph description language. It is a simple way of describing graphs that both humans and computer programs can use. DOT graphs are typically files that end with the *.dot* extension.

Various programs exist to process DOT files. These programs can read DOT files and render them, or provide an interface to manipulate the graphs. One such program, *dot*, is used by the source code documentation generator doxygen. *dot* is part of the Graphviz package.

Contents

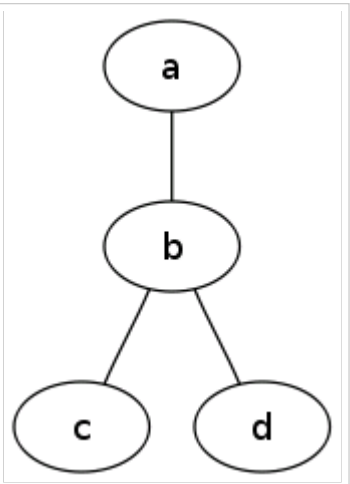
Syntax

Graph types

Undirected graphs

At its simplest, DOT can be used to describe an undirected graph. An undirected graph shows simple relations between objects, such as friendship between people. The *graph* keyword is used to begin a new graph, and nodes are described within curly braces. A double-hyphen (--) is used to show relations between the nodes.

```
graph graphname {
    a -- b -- c;
    b -- d;
}
```

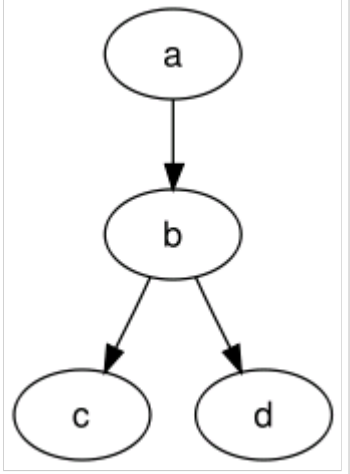


An undirected graph

Directed graphs

Similar to undirected graphs, DOT can describe directed graphs, such as flowcharts and dependency trees. The syntax is the same as for undirected graphs, except the *digraph* keyword is used to begin the graph, and an arrow (->) is used to show relationships between nodes.

```
digraph graphname {
    a -> b -> c;
    b -> d;
}
```

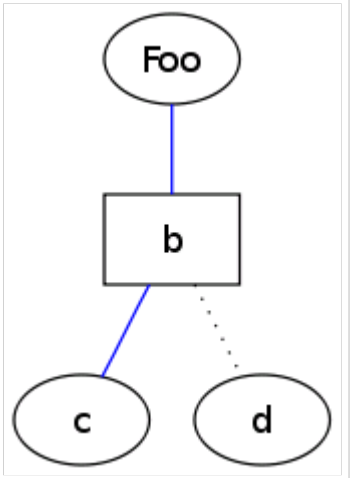


A directed graph

Attributes

Various attributes can be applied to nodes and edges in DOT files. These attributes can control aspects such as color, shape, and line styles. One or more attribute-value pairs are placed in square brackets ([]) after a statement and before the semicolon. Multiple attributes are separated by a comma and a space. Node attributes are placed after a statement containing only the name of the node, and no relations.

```
graph graphname {
    // The label attribute can be used to change the label of
    a [label="Foo"];
    // Here, the node shape is changed.
    b [shape=box];
    // These edges both have different line properties
    a -- b -- c [color=blue];
    b -- d [style=dotted];
}
```



A graph with attributes

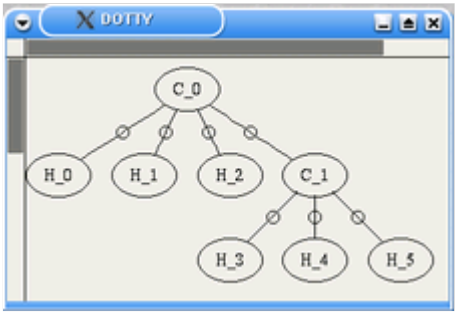
Comments

Dot supports C and C++ style single line and multiple line comments. In addition, it ignores lines with a number sign symbol (#) as their first character.

```
// This is a single line comment.
/* This is a
   multiple line
   comment. */
# Lines like this are also ignored.
```

A simple example

Following is an example script that describes the bonding structure of an ethane molecule. This is an undirected graph and contains node attributes as explained above.



A rendering of the example script using the tool `dotty`

```
graph ethane {
    C_0 -- H_0 [type=s];
    C_0 -- H_1 [type=s];
    C_0 -- H_2 [type=s];
    C_0 -- C_1 [type=s];
    C_1 -- H_3 [type=s];
    C_1 -- H_4 [type=s];
    C_1 -- H_5 [type=s];
}
```

Layout programs

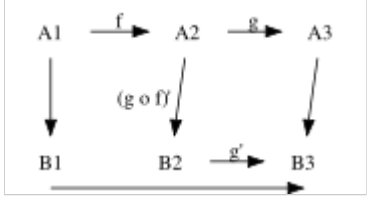
The DOT language defines a graph, but does not provide facilities for rendering the graph. There are several programs that can be used to render, view, and manipulate graphs in the DOT language:

- Graphviz - A collection of libraries and utilities to manipulate and render graphs
- Grappa - A Java based graph editor and viewer based on Graphviz
- Tulip (http://tulip.labri.fr/) can import dot files for analysis
- OmniGraffle can import a subset of DOT, producing an editable document. (The result cannot be exported back to DOT, however.)
- ZGRViewer, a GraphViz/DOT Viewer link (http://zvtm.sourceforge.net/zgrviewer.html)
- VizierFX, A Flex graph rendering library link (http://markandrewgoetz.com/vizierfx)

Limitations

It is possible to specify layout details with DOT, although not all tools that implement the DOT language pay attention to the position attributes. Thus, depending on the tools used, users must rely on automated layout algorithms (potentially resulting in unexpected output) or tediously hand-position nodes.

For example:



An image that seems improperly rendered

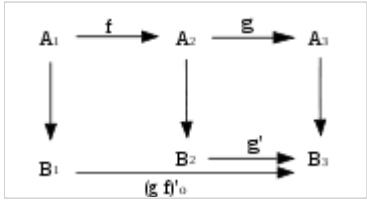
```
digraph g {
        node [shape=plaintext]
        A1 -> B1
        A2 -> B2
        A3 -> B3

        A1 -> A2 [label=f]
        A2 -> A3 [label=g]
        B2 -> B3 [label="g'"]
        B1 -> B3 [label="(g o f)'" tailport=s headport=s]

        { rank=same; A1 A2 A3 }
        { rank=same; B1 B2 B3 }
}
```

There are two problems in the image above. The square on the right is not a perfect square and the label "(g o f)'" is in the wrong place.

This can be fixed with Inkscape or other SVG editors. In some cases, this can also be fixed by using the *pos* attribute to specify a position.



After moving labels and arrows a bit, and changing font size of subscripts, the image looks correct.

See also

- Graphviz
- Graph (data structure)
- lisp2dot tool to convert Lisp programming language-like program trees to DOT language. Designed for use with genetic programming.

External links

- DOT tutorial and specification (http://www.graphviz.org/Documentation.php)
- Gallery of examples (http://www.graphviz.org/Gallery.php)
- Boost Graph Library (http://www.boost.org/libs/graph/doc/index.html)
- Grappa Drawing Package (http://www.research.att.com/~john/Grappa/)
- Online graph publisher in SVG, PNG, GIF, JPG (http://graph.gafol.net/)

Retrieved from "http://en.wikipedia.org/wiki/DOT_language"
Categories: Mathematical software | Graph description languages