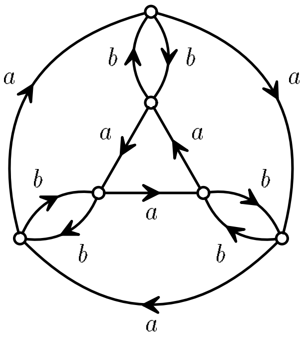


Graph-TEX User's Manual

John Pliam[1]

Version 1.0β, February 9, 1994

1 Introduction

If you are anal retentive like me, there are certain things you expect to see in a graph: smoothly curved edges, spade-like arrowheads which follow the curve of the edge, TEX labels, etc. Graph-TEX tries to meet these expectations by performing tedious calculations from high school trigonometry.

Graph-TEX is actually a *Perl* library (`graphtex.pl`) along with a preprocessor (`gt`) which can be used to generate *TEXdraw* commands producing graphs and related diagrams inside a TEX document. A graph can be "typeset" by writing `.gt` source file which resembles a matrix environment [2].

If your graph can be drawn with all its vertices lying on a rectilinear grid, you needn't be aware of the Perl interface aside from carrying out the

`.gt` → `.tex`

preprocessing. On the other hand, some graphs, such as the Petersen graph, are not naturally drawn with their vertices on any rectilinear grid. A simple Perl script including both the position and adjacency information, can produce desired `.tex` file (see §4 below).

2 Installation

First, get the latest compressed tar file via anonymous ftp from `ima.umn.edu`. It is in the directory `/pub/graphtex`. In order for Graph-TEX to work, you need to have the following packages installed: TEX, Perl, TEXdraw, and dvips (any `.dvi` to `.ps` program which respects `\special` for postscript insertion).

If you can write to the system-wide *bin* and *lib* directories and wish to install Graph-TEX right away, attend to definitions `PERLLIB`, `PERLEXEC` and `BINDIR` in `Makefile`, and type

```
make install
```

If you only wish to test Graph-TEX attend to the `PERLEXEC` definition in `Makefile` and type

```
make test-install
```

This assumes that your system supports the "magic number" mechanism for script programs, i.e. first lines of the form `#!`*path* (if not, you'll have to hack your own). A good way to test things is to rebuild the manual by typing

```
make
```

Note that because Graph-TEX relies on TEXdraw, the documents you create must include the TEXdraw macros, for example.

[1] Your special friend: pliam@ima.umn.edu

[2] This idea is an outright plagiarism of XY-pic, a GNU-protected package which is great for commutative diagrams, but which seemed inadequate for my graphs. Graph-TEX even started life as a set of macros for XY-pic.

```
\input texdraw
```

for TEX, and

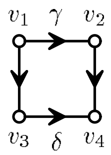
```
\documentstyle[texdraw]{book}
```

for LATEX.

3 Using the Preprocessor, gt

3.1 A Simple Example

The graph



was typeset by the following TEX commands:

```
The graph
$$
\input{examp1.tex}
$$
was typeset by ...
```

The file `examp1.tex` was automatically generated by the command line

```
gt examp1.gt
```

where the extension "`.gt`" is optional. The ultimate source file, `examp1.gt` looks like this:

```
%
% examp1.gt
%
\ver^{v_1} \rto^\gamma \dto & \ver^{v_2} \dto \\
\ver_{v_3} \rto_\delta & \ver_{v_4}
```

Notice that all commands and comments resemble TEX input even though they are processed by a Perl script. The array-like environment defines both the position of vertices and the adjacency structure of the graph. The vertices lie on a rectilinear grid whose spacing is `\versep` (10 mm by default). The edges must be defined in the cell corresponding to the vertex from which they emanate. The destination of an edge is usually defined by a string of relative displacement characters {`l`, `r`, `u`, `d`} (short for *left*, *right*, *up*, *down*) preceding the edge commands: `to`, `edge`.

3.2 Command Summary

Below is a table of graphics commands recognized by Graph-TEX. Notice that the vertices are enlarged, and the edges are shown along with vertices to indicate their relative positions.

\ver

\solver

\dblver

\box

\edge

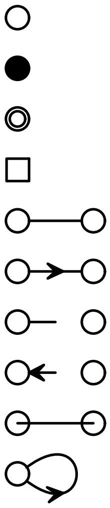
\to

\stub

\from

\stick

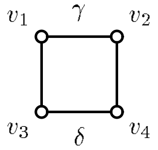
\loop



Additionally, objects can have labels (§3.3), edges can curve (§3.4), and most drawing attributes can be changed with commands `\gtset{}{}` and `\gtsetglb{}{}` (§3.5).

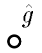
3.3 Labels

Edge and vertex labels also resemble the corresponding TEX commands, but they are more flexible. Consider the following variant of `examp1.gt`.

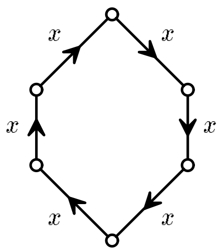



```
\ver^<{v_1} \redg^\gamma \dedg & \ver^>{v_2} \dedg \\
\ver_<{v_3} \redg_\delta & \ver_>{v_4}
```

Though the text offset is defined by the variable `\txtoff` the angle is defined by the sequence of characters from set { `<`, `>`, `^`, `v`, `_` }. For example, `\ver^^>{\hat{g}}` will produce a vertex labeled by a "$\hat{g}$". The label starts at a point `\txtoff` away from the vertex and at an angle of $\tan^{-1}(2)$, and "grows" in that direction.

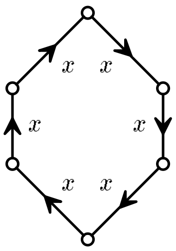


Edge labels can only have an "up" or "down" orientation defined by a single character in the set { `^`, `_` }. The actual position of the edge label is a function of the relative position of the source and destination vertices along with prescribed orientation. In other words, "up" may mean "down." The emphasis here is on symmetry; If you get one label right and stick with its orientation, you'll probably get all the orientations right. The next graph has all its labels outside:




```
& \ver \drto^x & \\
\ver \urto^x & & \ver \dto^x \\
\ver \uto^x & & \ver \dlto^x \\
& \ver \ulto^x &
```

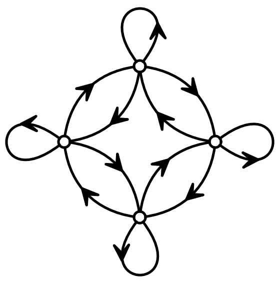
The following graph has all its labels inside, and is derived from the previous by changing `^` to `_`.




```
& \ver \drto_x & \\
\ver \urto_x & & \ver \dto_x \\
\ver \uto_x & & \ver \dlto_x \\
& \ver \ulto_x &
```

3.4 Loops and Curved Edges

Graph-TEX allows for loops and Bezier curved edges as in the following graph.

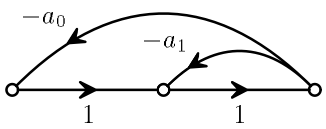


```
& \ver \loop(^) \ldto(+) \rdto(+) & \\
\ver \loop(<) \urto(+) \drto(+) & &
    \ver \loop(>) \luto(+) \ldto(+) \\
& \ver \loop(v) \luto(+) \ruto(+) &
```

The orientation of a loop is similar to that of a vertex label. A sequence of characters in the set { `<`, `>`, `^`, `v`, `_` } specifies the looping direction. The specification of an edge's bend is similar to that of its label. A single character in the set {`+`, `-`}, along with the relative position of the endpoints, determines the bending direction.

3.5 Changing the Attributes

Consider the graph




```
\ver \rrto_{1} & &
\ver \rrto_{1} & &
\ver \gtset{\edgmu}{0.80}
     \gtset{\labelmu}{0.80}
     \gtset{\bezfact}{1.0}
\llto(-)^{-a_1} \llllto(-)^{-a_0}
```

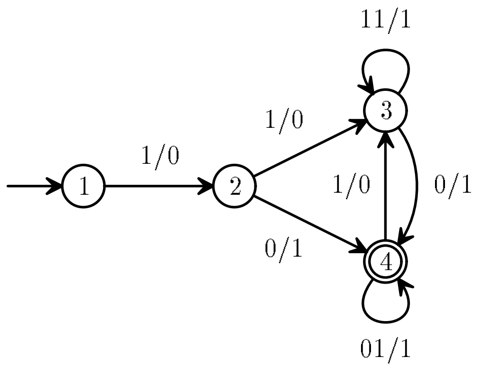
where it was necessary to locally change the variables `\edgmu` and `\labelmu`, which are the parameters defining how far to move along the Bezier curve before drawing the arrowhead and label, respectively. This was accomplished with the command `\gtset{}{}` which changes a variable only within the `&`'s (i.e. within the current *cell*). To change an attribute globally, the `\gtsetglb{}{}` command can be used before any other command. This is particularly useful when the picture as a whole has a different style than the typical graph. For example, the following exact sequence redefines `\versep`, `\verrad`, `\edgmu`, and `\bezfact`.

$$\begin{tikzcd} 0 \arrow[r] & M \arrow[r] & N \arrow[r] & P \arrow[r] \arrow[l, bend right] & 0 \end{tikzcd}$$

```
\gtsetglb{\linewd}{0.2}
\gtsetglb{\arrowdp}{0.5}
\gtsetglb{\versep}{15}
```

```
\gtsetglb{\verrad}{3}
\gtsetglb{\edgmu}{1}
\gtsetglb{\bezfact}{1.0}
0 \rto & M \rto & N \rto & P \rto \lto(-) & 0
```

The following graph is drawn in a style common in Automata Theory.




```
\gtsetglb{\verrad}{2.8}
\gtsetglb{\edgmu}{1}
\gtsetglb{\loopmu}{1}
& & & & 3 \ver \ddto(+)_{0/1} \loop(^)_{11/1} \\
1 \ver \llfrom \rrto^{1/0} & & 2 \ver \rruto^{1/0} \rrdto_{0/1} & & \\
& & & & 4 \dblver \uuto^{1/0} \loop(v)_{01/1}
```

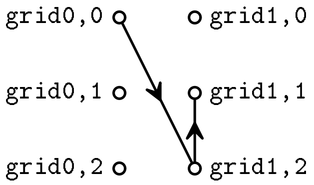
For a complete list of Graph-TEX variables see §5.1.

3.6 Absolute Form of Edge Commands

So far we have only shown the relative form of the edge commands, e.g. `\rruto`. However the destination of an edge can be defined in an absolute way by referring explicitly to its name inside `<` and `>`. When using `gt`, all vertices are given names of the form

$$\texttt{grid}x, y$$

where x and y are column and row integers, respectively, starting from `grid0,0` at the upper lefthand corner of the array environment. So for example, the following graph



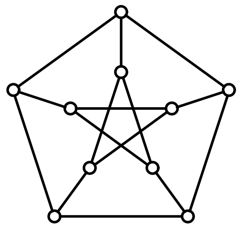

```
\ver \to<grid1,2> & \ver \\
\ver & \ver \\
\ver & \ver \to<grid1,1>
```

could have just as easily been created from the commands below.

```
\ver \rddto & \ver \\
\ver & \ver \\
\ver & \ver \uto
```

4 Using The Perl Library

In this section we describe the Perl library `graphtex.pl` which can be used to generate graphs whose vertex spacing is not rectilinear. Consider the Petersen graph below:



Since the graph possesses symmetries (i.e. it's automorphism group contains the dihedral group D_5), it is didactic to draw the graph with it's vertices lying on circles with an angular spacing of $2\pi/5$ radians. A placement of the vertices on a grid cannot express the symmetries. For this reason, Graph-TEX was built up from a library of Perl functions to facilitate the drawing of graphs with arbitrary vertex positions and adjacency.

To write a Perl script to generate TEXdraw commands for a graph, the following steps must be taken:

- Require `graphtex.pl`.
- Define the vertex positions in the associative array `%posit`.
- Define the adjacency structure in the associative array `%connect`.
- Open the file handle `OUT`.
- Call `&parse`.

In order to define a graph, a name must be assigned to each vertex, such as `in1`,...,`in5`,`out1`,...,`out5` for the Petersen graph. Then the position and adjacency information is attached to the name via the associative arrays[3] `%posit` and `%connect`. A `%posit` value is a string consisting of the x and y coordinates separated by white space. A `%connect` value is a string consisting of Graph-TEX commands as if they were in any cell of a `.gt` file. Of course in a `%connect` array, the absolute form of the edge commands (see §3.6) is the most useful. As an example we list the entire Perl script `petersen`:

```
#!/usr/local/bin/perl
# petersen - generates a TeXdraw commands for the Petersen graph
#
# $Id: petersen,v 1.1 1994/02/09 23:17:00 pliam Exp $

require 'graphtex.pl';


#
# set up %posit array for objects in1..in5, out1..out5
#
$inrad = 7;   # radius of inner pentagon
$outrad = 15; # radius of outer pentagon
$theta = $pi/2;
foreach $i (1..5) {
   $x = $inrad*cos($theta);
   $y = $inrad*sin($theta);
   $posit{"in$i"} = "$x $y";
```

[3] Associative arrays in Perl allow array values to be indexed by *strings* (as opposed to the more common *integer* indices).

```
   $x = $outrad*cos($theta);
   $y = $outrad*sin($theta);
   $posit{"out$i"} = "$x $y";
   $theta += 2*$pi/5;
}

#
# set up connectivity array %connect
#
%connect = (
   'in1', '\ver \edg<in3>',
   'in2', '\ver \edg<in4>',
   'in3', '\ver \edg<in5>',
   'in4', '\ver \edg<in1>',
   'in5', '\ver \edg<in2>',
   'out1', '\ver \edg<out2> \edg<in1>',
   'out2', '\ver \edg<out3> \edg<in2>',
   'out3', '\ver \edg<out4> \edg<in3>',
   'out4', '\ver \edg<out5> \edg<in4>',
   'out5', '\ver \edg<out1> \edg<in5>'
);

#
# parse away, printing to standard out
#
open(OUT, ">&STDOUT");
&parse;
```

5 Reference Guide

In this section we provide a pedantic description of the Graph-TEX variables and commands.

5.1 Variables

Certain variables define the attributes of graph drawing, such as edge curvature. Graph-TEX treats the sequence of commands appearing between the `&`'s as a subroutine call in the sense that changes in attributes (effected by the `\gtset{}{}` command) have scope limited to objects emanating from the current vertex. Of course global definitions can be declared using the `\gtsetglb{}{}` command.

The following table gives the Graph-TEX variables, their defaults, and their meanings.

`\drawdim`	mm	drawing dimension
`\linewd`	0.3	line width
`\txtrefh`	*null string*	override horizontal text reference parameter (L, C, R)
`\txtrefv`	*null string*	override vertical text reference command (T, C, B)
`\vertstr`	1.0	vertical stretch
`\arrowwd`	2.0	arrowhead width
`\arrowln`	2.0	arrowhead length
`\arrowdp`	1.0	arrowhead depth (length where wings meet)
`\verrad`	0.75	vertex radius
`\versep`	10	vertex separation
`\txtoff`	2.0	text offset
`\bezfact`	0.75	Bezier curve stretch factor
`\edgmu`	0.5	how far to move along edge before arrow
`\labelmu`	0.5	how far to move along edge before label
`\loopln`	10	dimensions of rectangle enclosing loop
`\loopwd`	15	"
`\loopmu`	0.18	how far to move along loop before arrow
`\llabelmu`	0.5	how far to move along loop before label
`\stubrat`	0.5	fraction of `\versep` to draw `\stub` or `\from`
`\inverrat`	0.75	ratio of inner to outer radius for `\dblver`
`\boxwd`	1.5	width of a `\box`
`\boxht`	1.5	height of a `\box`

5.2 Command Syntax

The following is a stylized BNF of the syntax of a `.gt` file:

$$
\begin{array}{rcl}
\langle\text{gt file}\rangle & \rightarrow & \langle\text{global decl.}\rangle\ \langle\text{gt environment}\rangle \\
\langle\text{global decl.}\rangle & \rightarrow & \varepsilon \\
 & | & \texttt{\textbackslash gtsetglb\{}\ \langle\text{variable ident.}\rangle\ \texttt{\}\{}\ \langle\text{PERL EXPR.}\rangle\ \texttt{\}}\ \langle\text{global decl.}\rangle \\
\langle\text{variable ident.}\rangle & \rightarrow & \backslash\ \langle\text{STRING}\rangle \\
\langle\text{gt environment}\rangle & \rightarrow & \langle\text{line}\rangle\ \backslash\backslash\ \langle\text{line list}\rangle \\
\langle\text{line list}\rangle & \rightarrow & \varepsilon\ |\ \langle\text{line}\rangle\ \backslash\backslash\ \langle\text{line list}\rangle \\
\langle\text{line}\rangle & \rightarrow & \langle\text{cell}\rangle\ \texttt{\&}\ \langle\text{cell list}\rangle \\
\langle\text{cell list}\rangle & \rightarrow & \varepsilon\ |\ \langle\text{cell}\rangle\ \texttt{\&}\ \langle\text{cell list}\rangle
\end{array}
$$

Below is the BNF of the syntax of a ⟨cell⟩. These can of course occur between &'s in a `.gt` file, or inside the `%connect` array of a Perl script.

$$
\begin{array}{rcl}
\langle\text{cell}\rangle & \rightarrow & \langle\text{command}\rangle\ \langle\text{command list}\rangle \\
\langle\text{command list}\rangle & \rightarrow & \varepsilon \mid \langle\text{command}\rangle\ \langle\text{command list}\rangle \\
\langle\text{command}\rangle & \rightarrow & \langle\text{local decl.}\rangle \\
 & \mid & \langle\text{math expr}\rangle \\
 & \mid & \langle\text{vertex}\rangle \\
 & \mid & \langle\text{edge}\rangle \\
 & \mid & \langle\text{stub}\rangle \\
 & \mid & \langle\text{box}\rangle \\
\langle\text{local decl.}\rangle & \rightarrow & \texttt{\textbackslash gtset\{}\ \langle\text{variable ident.}\rangle\ \texttt{\}\{}\ \langle\text{PERL EXPR.}\rangle\ \texttt{\}} \\
\langle\text{math expr}\rangle & \rightarrow & \langle\text{STRING}\rangle \\
\langle\text{vertex}\rangle & \rightarrow & \langle\text{vertex type}\rangle\ \langle\text{ver. label}\rangle \\
\langle\text{vertex type}\rangle & \rightarrow & \texttt{\textbackslash ver} \mid \texttt{\textbackslash dblver} \mid \texttt{\textbackslash solver} \\
\langle\text{ver. label}\rangle & \rightarrow & \varepsilon \mid \langle\text{ver. orientation}\rangle\ \texttt{\{}\ \langle\text{math expr}\rangle\ \texttt{\}} \\
\langle\text{ver. orientation}\rangle & \in & \{\ \texttt{<}, \texttt{>}, \texttt{\^{}}, \texttt{v}, \texttt{\_}\ \}^* \\
\langle\text{edge}\rangle & \rightarrow & \langle\text{relative edge}\rangle\ \langle\text{edge options}\rangle \\
 & \mid & \langle\text{absolute edge}\rangle\ \langle\text{edge options}\rangle \\
\langle\text{relative edge}\rangle & \rightarrow & \backslash\ \langle\text{displacement}\rangle\ \langle\text{edge type}\rangle \\
\langle\text{absolute edge}\rangle & \rightarrow & \backslash\ \langle\text{edge type}\rangle\ \texttt{<}\ \langle\text{STRING}\rangle\ \texttt{>} \\
\langle\text{edge type}\rangle & \rightarrow & \texttt{to} \mid \texttt{edg} \mid \texttt{stick} \\
\langle\text{displacement}\rangle & \in & \{\texttt{l}, \texttt{r}, \texttt{u}, \texttt{d}\}^* \\
\langle\text{edge options}\rangle & \rightarrow & \langle\text{bend}\rangle\ \langle\text{edge label}\rangle \\
\langle\text{bend}\rangle & \rightarrow & \varepsilon \mid \texttt{(+)} \mid \texttt{(-)} \\
\langle\text{edge label}\rangle & \rightarrow & \varepsilon \mid \langle\text{edge orientation}\rangle\ \texttt{\{}\ \langle\text{math expr}\rangle\ \texttt{\}} \\
\langle\text{edge orientation}\rangle & \rightarrow & \texttt{\^{}} \mid \texttt{\_} \\
\langle\text{stub}\rangle & \rightarrow & \backslash\ \langle\text{displacement}\rangle\ \langle\text{stub type}\rangle \\
\langle\text{stub type}\rangle & \rightarrow & \texttt{stub} \mid \texttt{from} \\
\langle\text{box}\rangle & \rightarrow & \texttt{\textbackslash box}
\end{array}
$$