Pedigree and genealogy tree diagrams are proven tools to visualize genetic and relational connections between individuals. The naming for mathematical tree structures with parent nodes and child nodes is traded from historical family diagrams. However, even the smallest family entity consisting of two parents and several children is no mathematical tree but a more general graph.

The genealogytree package provides a set of tools to typeset such genealogy trees or, more precisely, to typeset a set of special graphs for the description of family-like structures. The package uses an auto-layout algorithm which can be customized to e.g. prioritize certain paths.

Contents

1 Introduction ........................................ 9
  1.1 Genealogy Trees .................................. 9
  1.2 Package Design Principles and Philosophy .......... 10
  1.3 Comparison with Other Packages .................. 11
  1.4 Installation ...................................... 12
  1.5 Loading the Package .............................. 12
  1.6 Libraries ......................................... 13
  1.7 How to Get Started ................................ 13

2 Tutorials ............................................. 15
  2.1 Tutorial: First Steps (Ancestor Tree) .............. 15
    2.1.1 Document Setup ............................... 15
    2.1.2 Creation of a Basic Ancestor Diagram ........ 16
    2.1.3 Applying options .............................. 18
    2.1.4 Growing the Tree .............................. 19
    2.1.5 Prioritize and Colorize a Path ................. 22
    2.1.6 Changing the Timeflow ........................ 24
  2.2 Tutorial: Diagram Manipulation by ID values (Descendant Tree) .... 25
    2.2.1 Creation of a Basic Descendant Diagram .......... 25
    2.2.2 Growing the Tree .............................. 26
    2.2.3 Separating Diagram Data and Diagram Drawing ... 29
    2.2.4 Emphasizing a Relationship Path ............... 30
    2.2.5 Coloring Subtrees .............................. 32
  2.3 Tutorial: A Database Family Diagram (Sand Clock) .... 34
    2.3.1 Creation of a Basic Sand Clock Diagram ......... 34
    2.3.2 Node Content in Database Format ............... 35
5.11 Ignoring Input .......................... 117
5.12 Inserting Input .......................... 121
5.13 Phantom Nodes and Subtrees .......... 125
5.14 Childless Families ...................... 128
5.15 Autofill Parent Graphs (Ancestors) .. 132
5.16 Special and Auxiliary Options ........ 136

6 Node Data (Content) Processing ........ 137
6.1 Setting a Node Data Processing and Processor ........................................ 138
6.2 Predefined Non-Interpreting Processings .................................................. 139
   6.2.1 \texttt{fit} .................................. 139
   6.2.2 \texttt{tcolorbox} ......................... 143
   6.2.3 \texttt{tcbbox} ............................ 146
   6.2.4 \texttt{tcbbox*} ......................... 149
   6.2.5 \texttt{tikznode} ......................... 152
6.3 Creating a Customized Non-Interpreting Processor ................................ 154
6.4 Content Interpreters ...................... 155

7 Database Processing ..................... 161
7.1 Database Concept ........................ 162
7.2 Example Settings ........................ 163
7.3 Data Keys ................................ 165
7.4 Input Format for Dates .................. 172
7.5 Formatting the Node Data ................ 174
7.6 Formatting Names ......................... 184
7.7 Formatting Dates ......................... 186
7.8 Formatting Places ....................... 190
7.9 Formatting Events ....................... 191
7.10 Formatting Lists of Events ............. 193
7.11 Formatting Comments ................... 195
7.12 Formatting Professions .................. 196
7.13 Formatting Lists of Information ....... 197
7.14 Formatting Sex ........................... 198
7.15 Formatting Images ....................... 199
7.16 Formatting Ages ......................... 201

8 Edges .................................. 203
8.1 Edge Settings ............................ 204
8.2 Edge Types ................................ 208
8.3 Edge Parameters ......................... 213
8.4 Edge Labels .............................. 217
8.5 Edge Labels Database .................... 219
8.6 Adding and Removing Nodes from Edge Drawing ................ .................. 221
8.7 Extra Edges ............................... 226
8.8 Edge Shifting ............................. 231

9 Genealogy Symbols ...................... 237
9.1 Symbol Color Settings ................... 237
   9.1.1 Global Color Settings ................. 237
   9.1.2 Local Color Settings .................. 238
9.2 List of Symbols .......................... 239
9.3 Legend to Symbols ........................ 243
9.3.1 Printing a Legend ........................................ 243
9.3.2 Description Texts and Language Settings ............. 244

10 Language and Text Settings ................................. 247
10.1 Preamble Settings ......................................... 247
10.2 Document Settings ......................................... 248

11 Debugging: Library .......................................... 249
11.1 Parser Debugging ........................................... 249
11.2 Processor Debugging ....................................... 252
11.3 Graphical Debugging ....................................... 261
11.4 Show Information .......................................... 265

12 Templates: Library ........................................... 269
12.1 Using Templates ............................................ 269
12.2 Template 'formal graph' .................................... 269
12.3 Template 'signpost' ......................................... 270
12.4 Template 'symbol nodes' ................................... 270
12.5 Template 'tiny boxes' ....................................... 271
12.6 Template 'tiny circles' ...................................... 271
12.7 Template 'directory' ........................................ 272
12.8 Template 'database pole' ................................... 273
12.9 Template 'database pole reduced' ......................... 275
12.10 Template 'database poleportrait' ......................... 276
12.11 Template 'database poleportrait reduced' .............. 278
12.12 Template 'database portrait' .............................. 279
12.13 Template 'database portrait reduced' .................... 281
12.14 Template 'database traditional' ......................... 282
12.15 Template 'database traditional reduced' ............... 283
12.16 Template 'database sideways' ............................ 284
12.17 Template 'database sideways reduced' ................. 286
12.18 Template 'database sidewaysportrait' ................... 287
12.19 Template 'database sidewaysportrait reduced' ........ 288
12.20 Template 'database relationship' ....................... 289
12.21 Template 'ahnentafel 3' .................................. 290
12.22 Template 'ahnentafel 4' ................................... 292
12.23 Template 'ahnentafel 5' ................................... 294
12.24 Template 'a4paper 3' ...................................... 296
12.25 Template 'a4paper 4' ...................................... 297
12.26 Template 'a3paper 5' ...................................... 298
12.27 Template 'letterpaper 3' .................................. 299
12.28 Template 'letterpaper 4' .................................. 299
12.29 Template 'display 16:9 3' ................................. 300
12.30 Template 'display 16:9 4' ................................ 301
12.31 Template 'display 16:9 5' ................................ 302
12.32 Template 'display 16:10 3' ............................... 303
12.33 Template 'display 16:10 4' ............................... 303
12.34 Template 'display 16:10 5' ............................... 303
12.35 Border Size Options ...................................... 304
12.36 Predefined Colors of the Library ....................... 306
12.37 Auxiliary Control Sequences ............................. 307
13 Fancharts: Library

13.1 Fan Chart Diagrams ........................................ 309
13.2 Geometry Options ........................................... 312
13.3 Segment Text Options ....................................... 318
13.4 Color and Style Options .................................... 320
13.5 Templates ................................................... 333
  13.5.1 Template 'spartan' ...................................... 333
  13.5.2 Template 'malefemale sober' ........................... 334
  13.5.3 Template 'malefemale relation' .......................... 334
  13.5.4 Template 'colorwheel sober' ............................ 335
  13.5.5 Template 'colorwheel serious' .......................... 335
  13.5.6 Template 'colorwheel malefemale' ..................... 336
  13.5.7 Template 'colorwheel rich' .............................. 336
  13.5.8 Template 'colorwheel opulent' ......................... 337
  13.5.9 Template 'wave sober' .................................. 337
  13.5.10 Template 'wave serious' ............................... 338
  13.5.11 Template 'wave malefemale' ........................... 338
  13.5.12 Template 'wave rich' .................................. 339
  13.5.13 Template 'wave opulent' ................................ 339
  13.5.14 Template 'radial sober' ............................... 340
  13.5.15 Template 'radial serious' ............................. 340
  13.5.16 Template 'radial malefemale' ......................... 341
  13.5.17 Template 'radial rich' ................................ 341
  13.5.18 Template 'radial opulent' ............................. 342
13.6 Advanced Customization ..................................... 343

14 Auto-Layout Algorithm ........................................ 347

14.1 Preliminaries ............................................... 347
  14.1.1 Aesthetic Properties .................................... 347
  14.1.2 Genealogy Trees ........................................ 348
  14.1.3 Graph Grammar ......................................... 348
14.2 Requirements ............................................... 349
  14.2.1 Parent and Child Alignment ............................. 349
  14.2.2 Patchwork Families ..................................... 350
  14.2.3 Graph Growing Direction ............................... 351
14.3 Algorithmic Steps ........................................... 352
  14.3.1 Recursive Family and Node Placement .................. 352
  14.3.2 Contours ............................................... 352
  14.3.3 Combining Subtrees ..................................... 353
14.4 Known Problems ............................................. 354

15 Example Graph Files .......................................... 357

15.1 example.option.graph ....................................... 357
15.2 example.database.graph ..................................... 358
15.3 example.formal.graph ....................................... 359
15.4 example.neumann.graph .................................... 360

16 Stack Macros .................................................. 365

16.1 Creating a Stack ............................................ 365
16.2 Push to a Stack ............................................. 365
16.3 Pop from a Stack ............................................ 366
16.4 Peek into a Stack .......................................... 366
1.1 Genealogy Trees

The naming for mathematical tree structures with parent nodes and child nodes is traded from historical family diagrams. But, creating a family diagram for medical and sociological studies or family research can become surprisingly difficult with existing tools for tree visualization. The simple reason is, that a mathematical tree has only one parent node for its direct children nodes.

With reverse logic, this can be used to visualize ancestor diagrams starting from an individual to its predecessors:

However, even the smallest family entity consisting of two parents and several children is no mathematical tree but a more general graph:

The genealogytree package aims to support such graphs which will be denoted genealogy trees in the following. The graphs to be processed cannot become arbitrarily complex. A set of special graphs for the description of family-like structures is supported by the package algorithms. From at theoretical point of view, these graphs can be seen as a sort of annotated mathematical trees.
1.2 Package Design Principles and Philosophy

The emphasis of a genealogy tree is not the node or individual, but the family. A family is a set of arbitrarily many parents and children. From an algorithmic point of view, there could be more than two parents in a family.

A single family

A node is either a parent or a child to a specific family. A node can also be child to one family and parent to another (or many) families. Such a node is called a g-node (genealogy node) in the following.

Two families

The main restriction of the graph grammar is that there is exactly one g-node which connects its enclosing family to another one. In the example above, the father node is the g-node in the grandparents family. It is linked to the family with mother and children.

A strong driving force for elaborating and creating this package was to balance two contradictory goals for diagram generation: automatism and customization. In the ideal case, a diagram would be constructed automatically by data provided from an external data source and would also be freely customizable in any thinkable way starting changing colors, lines, shapes, node positioning, etc. In the real world, a trade-off between these goals has to be found.

Automatism:

- For a set of genealogy trees described by a grammar, see Chapter 4 on page 63, an auto-layout algorithms computes the node positioning.

- The graph grammar is family-centric and supports ancestors and descendants diagrams. For the later, multiple marriages can be used to a certain degree.

- The graph data can be written manually, but the package design is adjusted to process automatically generated data. There are many genealogy programs which manage family related data. The general idea is that such programs export selected diagram data in a text file using the provided grammar. Processing GEDCOM files directly by the package is not possible.

- While manipulations like coloring a single node can be done directly at node definition, the package design makes a lot of efforts to allow manipulations aside from the actual place of data definition, see Section 5.1.1 on page 78 and Section 5.1.2 on page 79. The idea is that automatically generated data has not to be edited, but can be manipulated from outside. Also, an automatically or manually generated data set can be used for several distinct diagrams; e.g. the graph data in Section 15.1 on page 357 is used numerous times inside this document for different diagrams.

GEDCOM (GEnealogical Data COMmunication) is a widely used data exchange format.
• The auto-layout algorithm is implemented in pure \TeX/\LaTeX{} (without Lua). This imposes all programming restrictions of this macro language on the implementation, but makes the package independent of external tools and fosters \LaTeX{} customization.

**Customization:**

• The auto-layout algorithm can be customized to e.g. prioritize certain paths or exclude selected subtrees. Also, several node dimensions and distances can be changed globally or locally.

• The appearance of a node can be customized with all capabilities of Ti\kZ{} [5] and \texttt{tcolorbox} [4]. Also, the node text can be processed.

• For the node content, a database concept can be used, see Chapter 7 on page 161. This gives a high degree of customizing the data visualization inside the node.

• The geometry of edges between nodes is not considered by the auto-layout algorithm, but edges can also be customized in many ways, see Chapter 8 on page 203.

• Several genealogy tree diagrams can be interconnected manually to form a more complex total diagram.

On the technical side, the package is based on *The TikZ and PGF Packages* [5] and uses *The tcolorbox package* [4] for the nodes. Since all processing is done in \TeX/\LaTeX{} without Lua and external tools, one should expect a lot of processing time for complex diagrams. Therefore, using an externalization of the diagrams is recommended.

### 1.3 Comparison with Other Packages

This is not really a comparison, but more a hinting to other possibilities for graph drawing. I am not aware of another package with focus on genealogy trees as defined here, but surely there are excellent other graph drawing packages. The first to name is Ti\kZ{} itself. There, one will find a bunch of graph drawing tools with different algorithms, partly implemented in Lua. The second one is the \texttt{forest} package which is also very powerful and does not need Lua.
1.4 Installation

Typically, `genealogytree` will be installed as part of a major LaTeX distribution and there is nothing special to do for a user.

If you intend to make a local installation manually, you have to install not only `tcolorbox.sty`, but also all `*.code.tex` files in the local `texmf` tree.

1.5 Loading the Package

The base package `genealogytree` loads the package `tcolorbox` with its skins, fitting, and external libraries. This also loads several other packages are loaded, especially `tikz` with its `arrows.meta` and `fit` libraries.

`genealogytree` itself is loaded in the usual manner in the preamble:

\usepackage{genealogytree}

The package takes option keys in the key-value syntax. Alternatively, you may use these keys later in the preamble with \gtruselibrary\textsuperscript{P.13} (see there). For example, the key to use debug code is:

\usepackage[debug]{genealogytree}
1.6 Libraries

The base package \texttt{genealogytree} is extendable by program libraries. This is done by using option keys while loading the package or inside the preamble by applying the following macro with the same set of keys.

\begin{verbatim}
\gtruselibrary{(key list)}
\end{verbatim}

Loads the libraries given by the \textit{(key list)}.

\begin{verbatim}
\gtruselibrary{all}
\end{verbatim}

The following keys are used inside \texttt{\gtruselibrary} respectively \texttt{\usepackage} without the key tree path /\texttt{gtr/library}/.

\begin{itemize}
  \item \texttt{/gtr/library/debug} \texttt{\{debug\}}
    Loads additional code for debugging a \textit{genealogy tree}. This is also useful for displaying additional informations during editing a graph; see Chapter 11 on page 249.
  \item \texttt{/gtr/library/templates} \texttt{\{templates\}}
    Loads additional code for templates. These are styles to set various options by one key; see Chapter 12 on page 269.
  \item \texttt{/gtr/library/fanchart} \texttt{\{fanchart\}}
    Loads additional code to draw ancestor fancharts. Therefor, the auto-layout algorithm is replaced by a specialized other algorithm; see Chapter 13 on page 309.
  \item \texttt{/gtr/library/all} \texttt{(style, no value)}
    Loads all libraries listed above.
\end{itemize}

For the curious readers: There are additional \textit{core} libraries which are loaded automatically and which are not mentioned here. Also, languages are special libraries which are loaded by \texttt{\gtrloadlanguage}. Third party libraries (denoted external libraries) can also be loaded using \texttt{\gtruselibrary}, if they follow the file naming scheme \texttt{gtrlib.(key).code.tex}

\begin{verbatim}
% Loading 'gtrlib.foobar.code.tex'
\gtruselibrary{foobar}
\end{verbatim}

Note that such external libraries are not version-checked as internal libraries are.

1.7 How to Get Started

You don’t have to read this long document to start creating your first \textit{genealogy tree}. A good starting point is to browse through the tutorials in Chapter 2 on page 15 and simply try some of them on your computer. The package provides a lot of options and allows many adjustments to node setting, but you do not need to know them in advance to create the first examples.

You should also take a look at Chapter 12 on page 269, where template examples are shown which could be useful instantly.

For using advanced features, it is not harmful to know at least the basics of Ti\textit{k}Z \cite{tikz} and \texttt{tcolorbox} \cite{tcolorbox}, since \texttt{genealogytree} is based on both.
2.1 Tutorial: First Steps (Ancestor Tree)

2.1.1 Document Setup

Most examples in this documentation will display some code snippets which one can use in a document with proper set-up. This very basic tutorial will create a tiny full document. If this does not work on your system, there is probably some installation problem. Typically, this can be solved by just updating the \TeX{} distribution.

The very first document just tests, if the package is installed:

\begin{verbatim}
\documentclass{article}
\usepackage[all]{genealogytree}
\begin{document}
\section{First Test}
Package loaded but not used yet.
\end{document}
\end{verbatim}
2.1.2 Creation of a Basic Ancestor Diagram

Now, we start with the very first genealogy tree. Such trees are family-centric. So, let us begin with a family consisting of mother and father and three children. Chapter 4 on page 63 tells us, that there are different kinds of families; the two main ones are parent and child. For a single family, the choice is quite irrelevant. Here, we think about extending the example to grandparents. Therefore, we take the parent construct.

Before the details are discussed, let us try a full example:

```
\documentclass{article}
\usepackage[all]{genealogytree}
\begin{document}
\section{First Test}
\begin{tikzpicture}
\genealogytree
  parent{
    g{first child}
    c{second child}
    c{third child}
    p{father}
    p{mother}
  }
\end{tikzpicture}
\end{document}
```

The environment `tikzpicture` is the main picture environment from the Ti\kZ \cite{tikz} package. `\genealogytree` can only be used inside such an environment.

When testing this example, be very sure about setting all braces properly. The internal parser will react very sensitive on input errors. Of course, this is nothing new for a \TeX user, but larger trees will have a lot of hierarchical braces and error messages will likely not be very talkative about where the error is.

The `genealogytree` package uses `{}` pairs for structuring and `[]` pairs for options like `typical` \LaTeX does.
In the following, we will not see full documents but code snippets and their output. Note that the full example used the all option to load all libraries of genealogytree, see Section 1.6 on page 13. You should also add all libraries for testing the examples. Later, you may choose to reduce the libraries.

Let us look at our example again with focus on the relevant part:

\begin{tikzpicture}
genealogytree{
  parent{
    g{first child}
    c{second child}
    c{third child}
    p{father}
    p{mother}
  }
}
\end{tikzpicture}

Our parent family has two parents denoted by \texttt{p} and three children, two of them denoted by \texttt{c} as expected. But one child, not necessarily the first one, is denoted by \texttt{g}. This is the \texttt{g}-node which connects a family uplink to another family. Here, we have a single family which is the root family where no uplink exists. Nevertheless, a \texttt{g}-node has to be present.
2.1.3 Applying options

Certainly, the size and distance of the nodes can be changed. A quick way to adapt the graph is to use preset values from a given /gtr/template on P. 269. We put this to the option list of \genealogytree on P. 55.

```latex
\begin{tikzpicture}
\genealogytree[template=signpost]{
  parent{
    g{first child}
    c{second child}
    c{third child}
    p{father}
    p{mother}
  }
}
\end{tikzpicture}
```

Options can also be set for families and nodes. We enhance our genealogy tree by giving /gtr/male on P. 101 and /gtr/female on P. 101 settings to the nodes:

```latex
\begin{tikzpicture}
\genealogytree[template=signpost]{
  parent{
    g[female]{first child}
    c[male]{second child}
    c[female]{third child}
    p[male]{father}
    p[female]{mother}
  }
}
\end{tikzpicture}
```
2.1.4 Growing the Tree

As next step, the father node shall get a grandfather and a grandmother. For this, the father node has to become a $g$-node which links the grandparents family to the root family:

\begin{tikzpicture}
genealogytree[template=signpost]{
parent{
  g[female]{first child}
  c[male]{second child}
  c[female]{third child}
  parent{
    g[male]{father}
  }
  p[female]{mother}
}
}\end{tikzpicture}

Visually, nothing happened. But, the father node is now $g$-node of a new family. As in our root family, we can add parents $p$ and even other children $c$. Of course, these other children are the siblings of the father node:

\begin{tikzpicture}
genealogytree[template=signpost]{
parent{
  g[female]{first child}
  c[male]{second child}
  c[female]{third child}
  parent{
    c[female]{aunt}
    g[male]{father}
    c[male]{uncle}
    p[male]{grandfather}
    p[female]{grandmother}
  }
  p[female]{mother}
}
}\end{tikzpicture}

One could replace all parents $p$ by \texttt{parent} families with a single $g$-node. This would increase the expense, but can be a good thing when editing and compiling a tree step by step.

We now prepare our tree for expansion and replace mother, grandfather, and grandmother with
appropriate parent families.
Again, we populate the three added families with parents $p$ and children $c$. 

\begin{tikzpicture}
\genealogytree[template=signpost]{
    parent{
        g[female]{first child}
        c[male]{second child}
        c[female]{third child}
        parent{
            c[female]{aunt}
            g[male]{father}
            c[male]{uncle}
            parent{
                g[male]{grandfather}
                p[male]{great-grandfather}
                p[female]{great-grandmother}
            }
            parent{
                g[female]{grandmother}
                p[male]{great-grandfather 2}
                p[female]{great-grandmother 2}
            }
            c[male]{granduncle}
        }
    parent{
        c[male]{uncle 2}
        g[male]{mother}
        p[male]{grandfather 2}
        p[female]{grandmother 2}
    }
}
\end{tikzpicture}
2.1.5 Prioritize and Colorize a Path

After the tree has been grown to its final size, we want to influence the node positions. Let us assume that the lineage from first child to great-grandmother 2 has to be especially emphasized.

To prioritize a node, the `/gtr/pivot` option can be used. This will place a node centered in relation to its ancestors and/or descendants. If this option is used for several connected nodes, a straight lineage is generated. All other nodes are placed automatically to respect this lineage.

To emphasize this lineage further, the respective nodes should be colorized differently. With standard settings, every node is drawn as a `tcolorbox`. Box options are given by `/gtr/box`.

The options inside `/gtr/box` are `tcolorbox` options [4]. To add a yellowish background color and glow, we use:

```
\%...
g[pivot,box={colback=yellow!20,no shadow,fuzzy halo},female]{first child}
\%...
```

All option settings are `pgfkeys` options. So, it is easy to create a new option style `highlight` which can be used for each node in the lineage. This can be done by `\gtrset` or inside the option list of `\genealogytree`.

```
\gtrset{highlight/.style={pivot,box={colback=yellow!20,no shadow,fuzzy halo}}}
```

Now, `highlight` can be used to apply `/gtr/pivot` and `/gtr/box` settings with one key word:

```
\%...
g[highlight,female]{first child}
\%...
```
\begin{tikzpicture}
\genealogytree[template=signpost, highlight/.style={pivot,box={colback=yellow!20,no shadow,fuzzy halo}},]
\parent{
    g[highlight,female]{first child}
    c[male]{second child}
    c[female]{third child}
    \parent{
        c[female]{aunt}
        g[highlight,male]{father}
        c[male]{uncle}
        \parent{
            g[male]{grandfather}
            \parent{
                g[highlight,female]{grandmother}
                p[male]{great-grandfather}
                p[female]{great-grandmother}
            }
            \parent{
                g[highlight,female]{grandmother}
                p[male]{great-grandfather 2}
                \parent{
                    c[male]{granduncle}
                }
            }
        }
        \parent{
            c[male]{uncle 2}
            g[female]{mother}
            p[male]{grandfather 2}
            p[female]{grandmother 2}
        }
    }
}
\end{tikzpicture}
2.1.6 Changing the Timeflow

A genealogy tree may be grown in four directions depending on the given /gtr/timeflow→P.80. Now, we will let the time flow to the left. Additionally, we replace the /gtr/template→P.269 setting by individual settings for /gtr/processing →P.138, /gtr/level size →P.84, /gtr/node size from →P.86, and /gtr/box →P.98.

\begin{tikzpicture}
\genealogytree[
  timeflow=left,
  processing=tcolorbox,
  level size=3.3cm,node size from=5mm to 4cm,
  box={size=small,halign=center,valign=center,fontupper=\small\sffamily},
  highlight/.style={pivot,box={colback=yellow!20,no shadow,fuzzy halo}},
]{
  parent{
    g[highlight,female]{first child}
    c[male]{second child}
    c[female]{third child}
    parent{
      c[female]{aunt}
      g[male]{father}
      c[male]{uncle}
      parent{
        g[male]{grandfather}
        p[male]{great-grandfather}
        p[female]{great-grandmother}
      }
      parent{
        g[highlight,female]{grandmother}
        p[male]{great-grandfather 2}
        p[highlight,female]{great-grandmother 2}
        c[male]{granduncle}
      }
      parent{
        c[male]{uncle 2}
        g[female]{mother}
        p[male]{grandfather 2}
        p[female]{grandmother 2}
      }
    }
  }
}\end{tikzpicture}
2.2 Tutorial: Diagram Manipulation by ID values (Descendant Tree)

This tutorial shows how to set up and save a descendant diagram which is going to be manipulated without changing the base data.

2.2.1 Creation of a Basic Descendant Diagram

For a genealogy tree displaying a descendant lineage, we take the `child` construct. As a first step, we start with a single family. As always, this root family has to have a `g`-node which serves no important role for a root family, but stands for a parent here. The resulting genealogy tree will contain just small nodes without names to display some interconnection. For this, a preset value from a given `/gtr/template` \( \text{P.269} \) is used for quick setup.

```
\begin{tikzpicture}
\genealogytree[template=formal graph]{
  child{
    g[male]{a_1}
    p[female]{a_2}
    c[female]{a_3}
    c[male]{a_4}
    c[female]{a_5}
  }
}\end{tikzpicture}
```

The nodes of the diagram already have some options settings. To select and manipulate some or many nodes later without editing the data, the nodes and families can be given unique `/gtr/id` \( \text{P.92} \) values.

```
\begin{tikzpicture}
\genealogytree[template=formal graph]{
  child[id=fam_A]{
    g[id=na1,male]{a_1}
    p[id=na2,female]{a_2}
    c[id=na3,female]{a_3}
    c[id=na4,male]{a_4}
    c[id=na5,female]{a_5}
  }
}\end{tikzpicture}
```
2.2.2 Growing the Tree

The nodes $a_3$ and $a_4$ shall become parent of their own families. To proceed in small steps, we make them g-nodes of single-member child families which does not change the diagram. Both new families get their own /gtr/id values for later reference.

\begin{tikzpicture}
genealogytree[template=formal graph]{
  child[id=fam_A]{
    g[id=na1,male]{a_1}
    p[id=na2,female]{a_2}
    child[id=fam_B]{
      g[id=na3,female]{a_3}
    }
    child[id=fam_C]{
      g[id=na4,male]{a_4}
    }
    c[id=na5,female]{a_5}
  }
}
\end{tikzpicture}

Now, the new families are populated by a second parent and children.

\begin{tikzpicture}
genealogytree[template=formal graph]{
  child[id=fam_A]{
    g[id=na1,male]{a_1}
    p[id=na2,female]{a_2}
    child[id=fam_B]{
      p[id=nb1,male]{b_1}
      g[id=na3,female]{a_3}
      c[id=nb2,male]{b_2}
      c[id=nb3,female]{b_3}
    }
    child[id=fam_C]{
      g[id=na4,male]{a_4}
      p[id=nc1,female]{c_1}
      c[id=nc2,male]{c_2}
    }
    c[id=na5,female]{a_5}
  }
}
\end{tikzpicture}

As a specialty, a union construct can be used inside a child family. This represents a second husband or wife including children for the g-node of the current child family. A union does not get its own g-node but shares the g-node of the child family.
In our example, node $a_4$ gets a *union* which has to be placed inside the family with id value *fam_C*:

As the reader may note, for *union* constructs, the edges between the nodes are likely to overlap. Therefore, to attenuate the effect, the vertical positions of the edges for *fam_C* and *fam_D* are shifted automatically. Also, note the small visual separation at the cross-point of both family edges. This is generated by using `/gtr/edge/foreground`\hspace{4pt}$^\text{P. } 213$ and `/gtr/edge/background`\hspace{4pt}$^\text{P. } 214$ (here, as preset values).

In some context, *fam_C* and *fam_D* will be seen as a single aggregated family and will be called *patchwork* family.

The tree is now grown further following the previous construction pattern.
### 2.2.3 Separating Diagram Data and Diagram Drawing

For the second part of this tutorial, the final diagram data is now saved into a file `example.formal.graph`, see Section 15.3 on page 359. That is everything inside \genealogytree\textsuperscript{P.55} without the options of \genealogytree\textsuperscript{P.55}. Using the \texttt{input} construct, graph drawing is done simply by the following:

\begin{tikzpicture}
\genealogytree[template=formal graph]
{input{example.formal.graph}}
\end{tikzpicture}

In our example, the given \texttt{/gtr/id}\textsuperscript{P.92} values are easy to remember since we choose them nearly identical to the node content. For a not-so-formal example, this will be different. To avoid digging into the data source for finding some \texttt{/gtr/id}\textsuperscript{P.92} value, the \texttt{/gtr/show id}\textsuperscript{P.265} setting from the \texttt{debug} library is useful:

\begin{tikzpicture}
\genealogytree[template=formal graph,show id]
{input{example.formal.graph}}
\end{tikzpicture}
2.2.4 Emphasizing a Relationship Path

For the given example data, we will emphasize the relationship between node $e_3$ and node $i_2$ in our graph. The diagram above exposes the id values along the relationship path as $ne_3$, $nb_3$, $na_3$, $na_1$ and $na_2$, $na_4$, $nd_4$, $ni_2$. For emphasizing, we dim the colors of all other nodes and brighten the colors for the nodes along this path.

All these manipulations are done inside the option list of `\genealogytree` \textsuperscript{P.55} without changing the diagram data directly.

1. `/gtr/box` \textsuperscript{P.98} sets options to wash out all nodes.
2. `/gtr/edges` \textsuperscript{P.204} sets options to wash out all edges.
3. `/gtr/options for node` \textsuperscript{P.95} sets box options to all nodes along the selected path to display them emphasized.
4. `/gtr/extra edges for families` \textsuperscript{P.227} sets extra edge options to all emphasized the connection line along the selected path.

\begin{tikzpicture}
\genealogytree
[template=formal_graph,
 box={colback=white,colupper=black!50,opacityframe=0.25},
 edges={foreground=black!25,background=black!5},
 options for node={ne3,nb3,na3,na1,na2,na4,nd4,ni2}]
 %

{box={colback=blue!50!red!20,colupper=black,opacityframe=1,fuzzy halo}},

extra edges for families={
  x={fam_E}{nb3}{ne3},x={fam_B}{na3}{nb3},
  x={fam_A}{na1,na2}{na3,na4},
  x={fam_D}{na4}{nd4},x={fam_I}{nd4}{ni2}
  }{foreground=blue!50!red,no background},
%
\]

{input{example.formal.graph}}
\end{tikzpicture}
Also, the parameters for the auto-layout algorithm can be changed using the known id values. Our selected relationship path is emphasized further by straightening the lineages. This is done by inserting \textit{\texttt{/gtr/pivot}}\textsuperscript{\textsuperscript{P.97}} values through \textit{\texttt{/gtr/options for node}}\textsuperscript{\textsuperscript{P.95}}.

\begin{tikzpicture}
\genealogytree[template=formal graph,
  box={colback=white,colupper=black!50,opacityframe=0.25},
  edges={foreground=black!25,background=black!5},
  options for node={ne3,nb3,na3,na1,na2,na4,nd4,ni2}]
  \node at (ne3) {$a_1$};
  \node at (na2) {$a_2$};
  \node at (na3) {$a_4$};

\end{tikzpicture}

All given \textit{\texttt{/gtr/id}}\textsuperscript{\textsuperscript{P.92}} values are also TikZ nodes. Therefore, a \textit{genealogy tree} can easily be annotated and extended by TikZ instructions.
2.2.5 Coloring Subtrees

For the given example data, the descendants of the root family should now be colored with three different colors. All in-law nodes should be visually separated from descendants of $a_1$ and $a_2$.

As a first step, the subtree denoted by fam_B is colored in red by /gtr/options for subtree. Analogously, fam_C is colored in blue. Node $a_5$ is a leaf node without own family and, therefore, is colored using /gtr/options for node. Also, the preset /gtr/male and /gtr/female styles are made ineffective for this drawing.

This gives a colored genealogy tree, but not only the direct descendents are colored, but all members of descendant families:

As can be inspected using /gtr/show type from the debug library, the nodes to be excluded are all p-nodes:

As a first step, the subtree denoted by fam_B is colored in red by /gtr/options for subtree. Analogously, fam_C is colored in blue. Node $a_5$ is a leaf node without own family and, therefore, is colored using /gtr/options for node. Also, the preset /gtr/male and /gtr/female styles are made ineffective for this drawing.

This gives a colored genealogy tree, but not only the direct descendents are colored, but all members of descendant families:

As can be inspected using /gtr/show type from the debug library, the nodes to be excluded are all p-nodes:
This node type is accessible by \texttt{\textbackslash gtrnodetype} or \texttt{\textbackslash gtrifpnode}. We use this to set up a \texttt{tcolorbox} style \texttt{bleach p} which wash out the in-law nodes, when \texttt{\textbackslash gtrifpnode} expands to \texttt{(true)}. This style is formulated locally by \texttt{/gtr/tcbset}:

\begin{tikzpicture}
\genealogytree[template=formal graph, male/.style={},female/.style={box={circular arc}},
tcbset={bleach p/.code={
  \ifpnode{\tcbset{enhanced jigsaw,opacityback=0.2}}{}}
},
\texttt{\textbackslash gtrifpnode}{\texttt{\textbackslash tcbset}{\texttt{\textbackslash enhanced jigsaw,opacityback=0.2}}}{}\}
],
options for subtree={fam_B}{\texttt{\textbackslash box}{\texttt{\textbackslash colback=red!20!white,bleach p}}},
options for subtree={fam_C,fam_D}{\texttt{\textbackslash box}{\texttt{\textbackslash colback=blue!20!white,bleach p}}},
options for node={na5}{\texttt{\textbackslash box}{\texttt{\textbackslash colback=green!20!white}}},
\texttt{\textbackslash input}{\texttt{\textbackslash example.formal.graph}}
\end{tikzpicture}

\begin{tikzpicture}
\genealogytree[template=formal graph, male/.style={},female/.style={box={circular arc}},
tcbset={bleach p/.code={
  \ifpnode{\tcbset{enhanced jigsaw,opacityback=0.2}}{}}
},
\texttt{\textbackslash gtrifpnode}{\texttt{\textbackslash tcbset}{\texttt{\textbackslash enhanced jigsaw,opacityback=0.2}}}{}\}
],
options for subtree={fam_B}{\texttt{\textbackslash box}{\texttt{\textbackslash colback=red!20!white,bleach p}}},
options for subtree={fam_C,fam_D}{\texttt{\textbackslash box}{\texttt{\textbackslash colback=blue!20!white,bleach p}}},
options for node={na5}{\texttt{\textbackslash box}{\texttt{\textbackslash colback=green!20!white}}},
\texttt{\textbackslash input}{\texttt{\textbackslash example.formal.graph}}
\end{tikzpicture}

\begin{tikzpicture}
\genealogytree[template=formal graph, male/.style={},female/.style={box={circular arc}},
tcbset={bleach p/.code={
  \ifpnode{\tcbset{enhanced jigsaw,opacityback=0.2}}{}}
},
\texttt{\textbackslash gtrifpnode}{\texttt{\textbackslash tcbset}{\texttt{\textbackslash enhanced jigsaw,opacityback=0.2}}}{}\}
],
options for subtree={fam_B}{\texttt{\textbackslash box}{\texttt{\textbackslash colback=red!20!white,bleach p}}},
options for subtree={fam_C,fam_D}{\texttt{\textbackslash box}{\texttt{\textbackslash colback=blue!20!white,bleach p}}},
options for node={na5}{\texttt{\textbackslash box}{\texttt{\textbackslash colback=green!20!white}}},
\texttt{\textbackslash input}{\texttt{\textbackslash example.formal.graph}}
\end{tikzpicture}
2.3 Tutorial: A Database Family Diagram (Sand Clock)

This tutorial shows the application of a database concept for representing the node content. Also, the sand clock diagram is shown which units ancestor and descendant graphs.

2.3.1 Creation of a Basic Sand Clock Diagram

The sandclock construct is the starting point for a sand glass type genealogy tree. The proband is the constriction for the sand glass where the ancestors and descendants of the proband meet. Therefore, a sandclock can and should contain child and parent constructs. There has to be exactly one child, because a sandclock has no own g-node but inherits it from the child.

For the following examples, we use \texttt{genealogypicture} to create genealogy trees. This is a handy combination of \texttt{tikzpicture} and \texttt{genealogytree}.

% minimal sandclock diagram
\begin{genealogypicture}[template=formal graph]
\sandclock
{  
child{
  g{\text{ proband }}
}
}
\end{genealogypicture}

Now, we can add parent and child constructs. Here, we use single-member families since the tree will be grown later on.

% basic sandclock diagram (ready to be extended)
\begin{genealogypicture}[template=formal graph]
\sandclock
{  
child{
  g{\text{ proband }}
  p{a}
  child{\% grows in child direction
    g{b}
  }
  child{\% grows in child direction
    g{c}
  }
}
parent{\% grows in parent direction
  g{A}
}
parent{\% grows in parent direction
  g{B}
}
}
\end{genealogypicture}
2.3.2 Node Content in Database Format

In the following, we will construct a family diagram for Carl Friedrich Gauß (1777–1855).

We step back a little bit and consider the minimal sand clock diagram as starting point. The node content, of course, may be any formatted \texttt{\LaTeX} text.

\begin{genealogypicture}
\[\text{processing=database, database format=medium marriage below,}\]
sandclock
{
  child{
  g[id=GauxCarl1777]{
    male,
    name={Johann \textbf{Carl Friedrich} \textit{Gauß}},
    birth={1777-04-30}{Braunschweig (Niedersachsen)},
    death={1855-02-23}{Göttingen (Niedersachsen)},
    profession={Mathematiker, Astronom, Geodät und Physiker},
    image={Carl_Friedrich_Gauss.jpg},
  }
}
\end{genealogypicture}

In this context, the database approach means that the node content should not contain a formatted text but just the data core which is going to be formatted later. This is the same principle as for creating a bibliography with \texttt{biblatex} or \texttt{bibtex}.

So, we tell \texttt{genealogytree} that we want to use such a database concept by setting \texttt{/gtr/processing \textsuperscript{P.138}} to \texttt{database}. Now, the content can be given as a key-value list. See Chapter 7 on page 161 for all feasible keys.

Further, we tell \texttt{genealogytree} how to format this given data by setting \texttt{/gtr/database format \textsuperscript{P.174}} to some predefined value. Everything can be customized later.

The basic information for a person is \texttt{/gtr/database/name \textsuperscript{P.165}}, \texttt{/gtr/database/male \textsuperscript{P.165}} or \texttt{/gtr/database/female \textsuperscript{P.165}}, \texttt{/gtr/database/birth \textsuperscript{P.169}} and \texttt{/gtr/database/death \textsuperscript{P.170}}.

\begin{genealogypicture}
\[\text{processing=database},\]
sandclock
{
  child{
  g{Carl Friedrich \textbf{Gauß},
    born 1777, died 1855}
}
}
\end{genealogypicture}

In the example above, we also added a \texttt{/gtr/database/profession \textsuperscript{P.166}} which appears in the output, and an \texttt{/gtr/database/image \textsuperscript{P.166}} which is not used. Note the markup with \texttt{\textbf{\textit{}}} inside the \texttt{/gtr/database/name \textsuperscript{P.165}} which marks preferred name parts and the surname. There is no name parsing as known from \texttt{bib(la)tex}.

As \texttt{/gtr/id \textsuperscript{P.92}} for Carl Friedrich Gauß, «GauxCarl1777» was chosen. Such id values could be chosen to your liking. As a common guideline, they should be human readable/understandable, because they may be needed to manipulate the graph afterwards and something like «gd0h-xhag-0ugh-opod-89sq-sdqj-8pah» may not be easily associated with Gauß. Also, they
should be automatically producible for the comfortable case, that a genealogy program exports
data in this format.

In this tutorial, this common guideline is sharpened to follow these rules:

- A person id is build as \textit{XxxxYyyyZzzz}, where \textit{Xxxx} are four letters of the surname, \textit{Yyyy} are four letters of the (preferred) first name, and \textit{Zzzz} is the year of birth (maybe, estimated).

- A family id is build as \textit{AaaaBbbbZzzz}, where \textit{Aaaa} are four letters of the husbands surname, \textit{Bbbb} are four letters of the wives surname, and \textit{Zzzz} is the year of marriage (maybe, estimated).

- Only \textit{a,...,z, A,..., Z} letters are used. Accented letters like umlauts are replaced by letters from the masks above. If a name part is shorter than four letters, letters from the masks are used for complement.

- If two identical id values are produced for two or more persons or families following these rules, they are distinguished by adding \textit{-(counter)}.

### 2.3.3 Formatting the Node Content

First, we adapt some graph geometry settings to our liking. /gtr/node size \(\rightarrow P. 85\), /gtr/level size \(\rightarrow P. 84\), and /gtr/level distance \(\rightarrow P. 83\) set size and distance values.

With /gtr/box \(\rightarrow P. 98\), we set \texttt{tcolorbox} options for the appearance of the node box. Note that \texttt{\gtrDBsex} is set to \texttt{male} by the database values inside the node content. There are predefined /tcb/male \(\rightarrow P. 101\) and /tcb/female \(\rightarrow P. 101\) styles, but with /gtr/tcbset \(\rightarrow P. 115\) we change them to colorize also the interior of the box.

\begin{genealogypicture}
[processing=database,
database format=medium marriage below,
node size=2.4cm,
level size=3.5cm,
level distance=6mm,
tcbset={male/.style={colframe=blue,colback=blue!5},
         female/.style={colframe=red,colback=red!5}},
box={fit basedim=7pt,boxsep=2pt,segmentation style=solid,
halign=left,before upper=\parskip\parskip1pt,\gtrDBsex,drop fuzzy shadow},
]
\end{genealogypicture}
As second step, we adapt the format of the given data inside the node output.

With `/gtr/list separators hang`\textsuperscript{P.194}, the event list is formatted with hanging indent. `/gtr/name font`\textsuperscript{P.185} and `/gtr/surn code`\textsuperscript{P.184} are used to format the name of the person. `/gtr/place text`\textsuperscript{P.190} inserts a `\newline` before the place of an event is printed in our example. Finally, `/gtr/date format`\textsuperscript{P.186} is used to change the way dates are printed.

\begin{genealogypicture}
\[\begin{array}{l}
\text{processing=database,}
\text{database format=medium marriage below,}
\text{node size=2.4cm,}
\text{level size=3.5cm,}
\text{level distance=6mm,}
\text{list separators hang,}
\text{name font=\bfseries,}
\text{surn code=\textcolor{red!50!black}{\#1},}
\text{place text=\newline{},}
\text{date format=d/mon/yyyy,}
\text{tcbset={male/.style={colframe=blue,colback=blue!5},}
\text{female/.style={colframe=red,colback=red!5}},}
\text{box={fit basedim=7pt,boxsep=2pt,segmentation style=solid,}
\text{halign=left,before upper=\parskip1pt,}
\text{\gtrDBsex,drop fuzzy shadow,}
\text{}}
\end{array}\]
\end{genealogypicture}

\textbf{Johann Carl Friedrich Gauß}

\begin{itemize}
\item 30/Apr/1777 Braunschweig (Niedersachsen)
\item \textsuperscript{†}23/Feb/1855 Göttingen (Niedersachsen)
\textbf{Mathematiker, Astronom, Geodäte und Physiker.}
\end{itemize}
2.3.4 Adding Images

The predefined `/gtr/database format` options do not consider images. But we can add image code easily to be `/gtr/box` definition which accepts `tcolorbox` settings.

`/tcb/if image defined` decides, if an image is present, and sets `tcolorbox` options accordingly. The file name of this image is `/gtrDBimage` which is set to `Carl_Friedrich_Gauss.jpg` by the database values inside the node content. Here, it is accessed by `/tikz/fill overzoom DBimage`.

Options from *The tcolorbox package* are used to enlarge the box width by 25mm and fill the space with this image:

```latex
\begin{genealogypicture}[
  \processing=database,
  database format=medium marriage below,
  node size=2.4cm,
  level size=3.5cm,
  level distance=6mm,
  list separators hang,
  name font=\bfseries,
  surn code={\textcolor{red!50!black}{#1}},
  place text={\newline}{},
  date format=d/mon/yyyy,
  tcbset={\male/.style={colframe=blue,colback=blue!5},
         \female/.style={colframe=red,colback=red!5}},
  box={fit basedim=7pt,boxsep=2pt,segmentation style=solid,
       halign=left,before upper=\parskip1pt,
       \gtrDBsex,drop fuzzy shadow,
       if image defined={add to width=25mm,right=25mm,
                         underlay={\begin{tcbclipinterior}
                       \path[fill overzoom DBimage]
                        ([xshift=-24mm]interior.south east) rectangle (interior.north east);
         \end{tcbclipinterior}},
       },{},
  },
\sandclock
{ child{
  \id=GauxCarl1777}{
    \male,
    name={Johann \textbf{Carl Friedrich} \textit{Gauß}},
    birth={1777-04-30}{Braunschweig (Niedersachsen)},
    death={1855-02-23}{Göttingen (Niedersachsen)},
    profession={Mathematiker, Astronom, Geodät und Physiker},
    image={Carl_Friedrich_Gauss.jpg},
  }
}
\end{genealogypicture}
```

1[^1]

[^1]: http://commons.wikimedia.org/wiki/File:Carl_Friedrich_Gauss.jpg
2.3.5 Full Example with Frame

The sandclock example is now extended with family and ancestors and descendants of Gauß as shown at the beginning of this tutorial. The full sandclock example is saved as «example.gauss.graph»:

```
sandclock{  
  child[id=GauaOsth1805]{  
    p[id=OsthJoha1780]{  
      female,  
      name=\{Johanna\} Elisabeth Rosina \{Osthoff\},  
      birth={1780-05-08}{Braunschweig (Niedersachsen)},  
      marriage={1805-10-09}{Braunschweig (Niedersachsen)},  
      death={1809-10-11}{G"ottingen (Niedersachsen)},  
      comment={Wei\ss{}gerberstochter},  
    }  
  }  
  g[id=GauxCarl1777]{  
    male,  
    name=\{Johann\} Carl Friedrich \{Gau\ss{}\},  
    birth={1777-04-30}{Braunschweig (Niedersachsen)},  
    death={1855-02-23}{G"ottingen (Niedersachsen)},  
    profession={Mathematiker, Astronom, Geod\"at und Physiker},  
    image={Carl_Friedrich_Gauss.jpg},  
  }  
  c[id=GauxCarl1806]{  
    male,  
    name=\{Carl\} Joseph \{Gau\ss{}\},  
    birth={1806-08-21}{Braunschweig (Niedersachsen)},  
    death={1873-07-04}{Hannover (Niedersachsen)},  
  }  
  c[id=GauxWilh1808]{  
    female,  
    name=\{Wilhelmina\} \{Gau\ss{}\},  
    birth={1808-02-29}{G"ottingen (Niedersachsen)},  
    death={1840-08-12}{T"ubingen (Baden-W"urttemberg)},  
  }  
  c[id=GauxLudw1809]{  
    male,  
    name=\{Ludwig\} \{Gau\ss{}\},  
    birth={1809-09-10}{G"ottingen (Niedersachsen)},  
    death={1810-03-01}{G"ottingen (Niedersachsen)},  
  }  
  union[id=GauaWald1810]{  
    p[id=WaldFrie1788]{  
      female,  
      name=\{Friederica\} Wilhelmine \{Waldeck\},  
      birth={1788-04-15}{G"ottingen (Niedersachsen)},  
      marriage={1810-08-14}{G"ottingen (Niedersachsen)},  
      death={1831-09-12}{G"ottingen (Niedersachsen)},  
      comment={Rechtswissenschaftlerstochter},  
    }  
  }  
  c[id=GauxEuge1811]{  
    male,  
    name=\{Eugen\} Peter Samuel Marius \{Gau\ss{}\},  
    birth={1811-07-29}{G"ottingen (Niedersachsen)},  
    death={1896-07-04}{Columbia (Missouri)},  
    profession={Rechtswissenschaftler, Kaufmann},  
  }  
  c[id=GauxWilh1813]{  
    male,  
    name=\{Wilhelm\} August Carl Matthias \{Gau\ss{}\},  
    birth={1813-10-23}{G"ottingen (Niedersachsen)},  
  }  
}
```
death={(1879-08-23)}{St. Louis (Missouri)},
}
c[id=GauxTher1816]{
  female,
  name={Henriette Wilhelmine Karoline \pref{Therese} \surn{Gau\ss{}}},
  birth={(1816-06-09)}{G\"ottingen (Niedersachsen)},
  death={(1864-02-11)}{Dresden (Sachsen)},
}
}
parent[id=GoosEgge1735]{
  g[id=GauxGebh1743]{
    male,
    name={\pref{Gebhard} Dietrich \surn{Gau\ss{}}},
    birth={(1743-02-13)}{Braunschweig (Niedersachsen)},
    death={(1808-04-14)}{Braunschweig (Niedersachsen)},
    profession={G\"artner, Wasserkunstmeister, Rechnungsf"uhrer},
  }
  parent[id=GoosLbtk1705]{
    g[id=GoosJyrg1715]{
      male,
      name={\pref{J\"urgen} Gooss},
      birth={(1715)}{V"olkenrode (Niedersachen)},
      death={(1774-07-05)}{Braunschweig (Niedersachsen)},
      profession={Lehmmaurer},
    }
    p[id=GoosHinr1655]{
      male,
      name={\pref{Hinrich} Gooss},
      birth={((caAD)1655)}{},
      death={(1726-10-25)}{V\"olkenrode (Niedersachen)},
    }
    p[id=LxttkKath1674]{
      female,
      name={\pref{Katharina} Magdalena \surn{Eggenlings}},
      birth={((caAD)1710)}{Rethen},
      marriage={((caAD)1735)}{V\"olkenrode (Niedersachen)},
      death={(1774-04-03)}{Braunschweig (Niedersachsen)},
    }
  }
  p[id=EggeKath1710]{
    female,
    name={\pref{Katharina} Magdalena \surn{Eggenlings}},
    birth={((caAD)1710)}{Rethen},
    marriage={((caAD)1735)}{V\"olkenrode (Niedersachen)},
    death={(1774-04-03)}{Braunschweig (Niedersachsen)},
  }
}
parent[id=BentKron1740]{
  g[id=BenzDoro1743]{
    female,
    name={\pref{Dorothea} Benz},
    birth={(1743-06-18)}{Velpke (Niedersachsen)},
    marriage={((caAD)1710)}{Rethen},
    death={(1839-04-18)}{G\"ottingen (Niedersachen)},
    comment={Steinhauerstochter},
  }
  parent[id=BentBbbb1740]{
    g[id=BentChri1717]{
      male,
      name={\pref{Christoph} Bentze},
      birth={(1717)}{Velpke (Niedersachsen)},
      death={(1748-09-01)}{Velpke (Niedersachsen)},
      profession={Steinhauer},
    }
  }
}

40
As final polish, \texttt{/gtr/edges}^{P.204} are set to be \texttt{rounded} and the used symbols are recorded by \texttt{/gtr/symbols record reset}^{P.243} and displayed by \texttt{/gtr/SymbolsLegend}^{P.244} inside \texttt{/gtr/after tree}^{P.115}.

Finally, the whole diagram is put into a titled \texttt{tcolorbox} to exhibit the example:

\begin{tcolorbox}[enhanced,sharp corners,boxrule=0.6pt,left=0pt,right=0pt,colback=blue!50!black,interior style image=goldshade.png,halign=center,center title,fonttitle=\bfseries,ttitle={The Family of Carl Friedrich Gau\ss\} \ (1777--1855)} ]\begin{genealogypicture}
\processing=database,
database format=medium marriage below,
node size=2.4cm,
level size=3.5cm,
level distance=6mm,
list separators hang,
name font=\bfseries,
surn code={\textcolor{red!50!black}{#1}},
place text={\newline}{},
date format=d/mon/yyyy,
tcbset={male/.style={colframe=blue,colback=blue!5},
female/.style={colframe=red,colback=red!5}},
box={fit basedim=7pt,boxsep=2pt,segmentation style=solid,halign=flush left,before upper=\parskip1pt,\gtrDBsex,drop fuzzy shadow,if image defined={add to width=25mm,right=25mm,underlay={\begin{tcbclipinterior}path[fill overzoom DBimage]\hspace*{-24mm}interior.south east) rectangle (interior.north east);}\end{tcbclipinterior}},}
\},
edges=rounded,
symbols record reset,
after tree={\node[font=\scriptsize\itshape,text width=1.8cm,below left,fill=white,fill opacity=0.4,text opacity=1]at (current bounding box.north east) {\gtrSymbolsLegend};},
}\input{example.gauss.graph}\end{genealogypicture}
\end{tcolorbox}
The Family of Carl Friedrich Gauß (1777–1855)

**Johanna**

**Elisabeth Rosina Osthoff**

- 8/May/1780
  - Braunschweig (Niedersachsen)
  - 11/Oct/1809
  - Göttingen (Niedersachsen)

**Weißgerberstochter.**

**Johann Carl Friedrich Gauß**

- 30/Apr/1777
  - Braunschweig (Niedersachsen)
  - 23/Feb/1855
  - Göttingen (Niedersachsen)
  - Mathematiker, Astronom, Geodät und Physiker.

**Katharina Lütken**

- 19/Aug/1764
  - Völkenrode (Niedersachsen)
  - 15/Apr/1749
  - Völkenrode (Niedersachsen)
  - ca. 1765
  - Völkenrode (Niedersachsen)

**Gebhard Dietrich Gauß**

- 13/Feb/1743
  - Braunschweig (Niedersachsen)
  - 14/Apr/1808
  - Braunschweig (Niedersachsen)
  - Gärtner, Wasserkunstmäster, Rechnungsführer.

**Katharina Magdalena Egingtings**

- ca. 1710
  - ca. 1735
  - Völkenrode (Niedersachsen)

**Dorothea Benze**

- 25/Apr/1776
  - Völkenrode (Niedersachsen)

**Jürgen Gooss**

- 1715
  - Völkenrode (Niedersachsen)
  - 5/Jul/1774
  - Braunschweig (Niedersachsen)

**Hinrich Gooss**

- ca. 1655
  - 25/Oct/1726
  - Braunschweig (Niedersachsen)

**Christoph Bentze**

- 1717
  - Velpke (Niedersachsen)
  - 1/Sep/1749
  - Velpke (Niedersachsen)

**Dorothea Benze**

- 1717
  - Velpke (Niedersachsen)
  - 1/Sep/1749
  - Velpke (Niedersachsen)

**Johanne Elisabeth Rosina Osthoff**

- ca. 1765
  - 11/Oct/1809
  - Göttingen (Niedersachsen)
  - Wolfserstochter.

**Friederica Wilhelmine Waldeck**

- ca. 1765
  - 12/Sep/1831
  - Göttingen (Niedersachsen)

**Katharina Krone**

- ca. 1710
  - after 1743
  - Völkenrode (Niedersachsen)

**Carl Joseph Gauß**

- 21/Aug/1806
  - Völkenrode (Niedersachsen)

**Wilhelmine Gauß**

- 29/Feb/1808
  - Göttingen (Niedersachsen)

**Ludwig Gauß**

- 10/Sep/1809
  - Göttingen (Niedersachsen)

**Eugen Peter Samuel Marius Gauß**

- 29/Jul/1811
  - Göttingen (Niedersachsen)

**Wilhelm August Carl Matthias Gauß**

- 23/Oct/1813
  - Göttingen (Niedersachsen)

**Henriette Wilhelmine Karoline Therese Gauß**

- 9/Jul/1816
  - Göttingen (Niedersachsen)

**Gebhard Dietrich Gauß**

- 13/Feb/1743
  - Braunschweig (Niedersachsen)

**Johanna Elisabeth Rosina Osthoff**

- ca. 1765
  - 11/Oct/1809
  - Göttingen (Niedersachsen)
  - Wolfserstochter.

**Friederica Wilhelmine Waldeck**

- ca. 1765
  - 12/Sep/1831
  - Göttingen (Niedersachsen)

**Katharina Krone**

- ca. 1710
  - after 1743
  - Völkenrode (Niedersachsen)

**Carl Joseph Gauß**

- 21/Aug/1806
  - Völkenrode (Niedersachsen)

**Wilhelmine Gauß**

- 29/Feb/1808
  - Göttingen (Niedersachsen)

**Ludwig Gauß**

- 10/Sep/1809
  - Göttingen (Niedersachsen)

**Eugen Peter Samuel Marius Gauß**

- 29/Jul/1811
  - Göttingen (Niedersachsen)

**Wilhelm August Carl Matthias Gauß**

- 23/Oct/1813
  - Göttingen (Niedersachsen)

**Henriette Wilhelmine Karoline Therese Gauß**

- 9/Jul/1816
  - Göttingen (Niedersachsen)

=born,
=married,
=died.
2.4 Tutorial: Descendants of the Grandparents (Connecting Trees)

This tutorial will show how to create a «descendants of the grandparents» type of diagram. For this, two genealogy trees have to be connected.

2.4.1 Descendants of the Two Grandparents

Since «descendants of the grandparents» cannot be formulated by the grammar of this package, see Chapter 4 on page 63, a descendants tree for each pair of grandparents is considered.

In this example, the proband is $c_4$. First, we take a look at the descendants of the father’s parents $a_1$ and $a_2$. Note that we arranged the red colored father’s family at the right hand side and that the father node $c_1$ has a combined $/gtr/id$ suffix of $c_1a$, because we added an $/gtr/id$ suffix of $0a$ to every id value of our first tree.

The other settings in this example are less important, but one may observe that the $/tcb/male$ and $/tcb/female$ styles were redefined to show not different colors but different shapes.
Secondly, we take a look at the descendants of the mother’s parents $b_1$ and $b_2$. Note that this time we arranged the red colored mother’s family at the left hand side and that the father node $c_1$ has a different /gtr/id \(^{+P.92}\) of \texttt{c1@b}, because we added an /gtr/id suffix \(^{+P.94}\) of \texttt{@b} to every id value of our second tree.

\begin{tikzpicture}
\genealogytree[template=formal graph,id suffix=@b, tcbset={male/.style={sharp corners},female/.style={circular arc}}, edges={anchoring=center},box={colback=blue!25}]
{
  child{
    g[male]{b_1} p[female]{b_2}
    child[family box={colback=red!25}]{
      p[male,id=c1]{c_1} g[female]{c_2}
      c[female,box={fuzzy halo,colback=yellow}]{c_3} c[female] c[male]{c_4} c[male]{c_5}
    }
    child{
      p[male]{b_3} g[female]{b_4}
      c[female]{b_5} c[male]{b_6} c[male]{b_7}
      union{
        p[male]{b_8} c[female]{b_9}
      }
    }
    child{
      g[male]{b_{10}} p[female]{b_{11}}
      c[female]{b_{12}} c[male]{b_{13}}
    }
  }
}\end{tikzpicture}

2.4.2 Connected Diagram

After the preparations, the \texttt{genealogytree} \(^{+P.55}\) diagrams can easily be put together.

Using /gtr/set position \(^{+P.111}\) with value \texttt{c1@b} at \texttt{c1@a} for the second \texttt{genealogytree} \(^{+P.55}\) puts node $c_1$ from the diagram directly on node $c_1$ of the first \texttt{genealogytree} \(^{+P.55}\). Note that in a more complicated situation more manual intervention may be necessary to avoid unwanted overlapping of other nodes.

In the first \texttt{genealogytree} \(^{+P.55}\), one sees a /gtr/phantom* \(^{+P.127}\) option which makes the first family $c_1,\ldots,c_5$ invisible but still space reserving.

Using /gtr/id suffix \(^{+P.94}\) or /gtr/id prefix \(^{+P.94}\) allows to distinguish nodes with the same id value in different trees. Otherwise, the id values would have to be changed manually.
2.5 Tutorial: Multi-Ancestors

In the following, a multi-ancestor denotes an ancestor who is connected over more than one descendency line to the proband, i.e. where descendants have children with other descendents. This situation is not covered by the auto-layout algorithm. Depending on the complexity, such a graph can be drawn by manipulating one or more genealogy trees.

2.5.1 Triple Ancestor Example

In this example, X and Y are triple ancestors of the proband Z. As first step, a child diagram is set up with all three descendency lines from X and Y to Z, but only the c line is drawn fully. In our example, a₅ and b₅ are parents to b₆, also b₆ and c₅ are parents to Z (not yet displayed).

\begin{genealogypicture}[template=formal graph, timeflow=left,]
child{
g[male]{X} p[female]{Y}
child{
child{ g[male]{a₁}
child{ g[female]{a₂}
child{ g[male]{a₃}
child{ g[female]{a₄}
child{ g[male]{a₅}
}}}}
child{ g[male]{b₁}
child{ g[male]{b₂}
child{ g[male]{b₃}
child{ g[male]{b₄}
child{ g[male]{b₅}
child{ g[female]{b₆}
}}}}
child{ g[male]{c₁}
child{ g[male]{c₂}
child{ g[male]{c₃}
child{ g[female]{c₄}
child{ g[male]{c₅}
child{ g[male]{Z}
}}}}}
}}}
}
\end{genealogypicture}
2.5.2 Adding Edges Manually

Now, we add the missing connections. For this, /gtr/id→P.92 values are added to all involved nodes and families. Then, the connections are drawn using /gtr/add parent→P.222 to add a5 as additional parent for b6, and to add b6 as additional parent for Z.

The diagram has all necessary edges now, but, currently, is not balanced.
2.5.3 Manual Position Adjustments

To balance the graph, the final position of the $b_6$ is adjusted using the `/gtr/tikz`\footnote{P.103} option with some TikZ shift operations. These final shiftings do not influence the auto-layout algorithm, but the edges move with the nodes.

Alternatively, `/gtr/distance`\footnote{P.96}, `/gtr/pivot`\footnote{P.97}, and `/gtr/pivot shift`\footnote{P.106} can be used to influence the auto-layout algorithm. `/gtr/pivot shift`\footnote{P.106} was used for the $Z$ node to move it inside its family. But these manipulations would not move a node from its layer as was done for $b_6$ to display the generation skip.

\begin{genealogypicture}[template=formal graph,timeflow=left,
add parent=a5 to AB\_fam,
add parent=b6 to BC\_fam,
]  
  child{  
    g[male]{X} p[female]{Y}  
    child{  
      g[male]{a_1}  
      child{  
        g[female]{a_2}  
        child{  
          g[male]{a_3}  
          child{  
            g[male]{a_4}  
            child{  
              g[female, id=a5]{a_5}  
            }  
          }  
        }  
      }  
    }  
  }  
  child{  
    g[male]{b_1}  
    child{  
      g[male]{b_2}  
      child{  
        g[male]{b_3}  
        child{  
          g[male]{b_4}  
          child[id=AB\_fam]{  
            g[male]{b_5}  
            child{  
              g[female, id=b6,tikz={xshift=-6.5mm,yshift=5.5mm}]{b_6}  
            }  
          }  
        }  
      }  
    }  
  }  
  child{  
    g[male]{c_1}  
    child{  
      g[female]{c_2}  
      child{  
        g[male]{c_3}  
        child{  
          g[female]{c_4}  
          child[id=BC\_fam,pivot shift=-8.25mm]{  
            g[male]{c_5}  
            child{  
              g[male]{Z}  
            }  
          }  
        }  
      }  
    }  
  }  
\end{genealogypicture}
2.6 Tutorial: Externalization

Creating diagrams requires a considerable amount of compilation time. Especially, if the document contains several diagrams, it is desirable to avoid compiling already finished diagrams on every document run. One solution would be to create a document for every diagram and to include the resulting PDF to the main document. Another way is known as externalization.

2.6.1 Externalization Process

Externalization means that diagrams are edited inside the main document as usual, but they are automatically exported to external files, compiled if necessary, and the resulting PDF files are included to the main document as images. At least two externalization options are available:

- TikZ externalization: Here, the whole original document is compiled in a sophisticated way. The external library of TikZ can automatically externalize all \texttt{tikzpicture} environments, see [5].

- \texttt{tcolorbox} externalization: Here, marked code snippets are compiled in a not so sophisticated but more robust way. The external library of \texttt{tcolorbox} only externalizes marked and named snippets. Besides \texttt{tikzpicture} environments, \texttt{genealogypicture} environments and other constructs can be externalized. These snippets are written to external files, compiled and the resulting PDF files are included to the main document as images, see [4].

The further tutorial describes the externalization using the external library of \texttt{tcolorbox}. This library is already included by the \texttt{genealogytree} package.

2.6.2 Document Setup

To use the externalization, the preamble of the main document has to contain the \texttt{\newcommand*{\tcbEXTERNALIZE}{}} command. Without this command, no externalization operation will be executed. Typically, \texttt{\tcbEXTERNALIZE} is the last entry of the preamble. Everything between \texttt{\tcbEXTERNALIZE} and \texttt{\begin{document}} is thrown away in the external document.

\begin{verbatim}
\documentclass{article}
\usepackage[all]{genealogytree}
\tcbEXTERNALIZE
\begin{document}
\section{My heading}
\% ...
Your main document containing texts, diagrams, figures, etc.
\% ...
\end{document}
\end{verbatim}

To use the externalization options, the compiler has to be called with the \texttt{-shell-escape} permission to authorize potentially dangerous system calls. Be warned that this is a security risk.
2.6.3 Marking Diagrams for Externalization

Before we care about externalization, we set up an example with two genealogy tree diagrams. One uses a \texttt{tikzpicture} and the other one the \texttt{genealogypicture} \texttt{P.57} shortcut.

\begin{verbatim}
This is the first example:\par\smallskip
\begin{tikzpicture}
\genealogytree[template=symbol nodes]{
child{
 gm pf cf
 child{gm pf cf cm}
 child{gm pf cm c- cm}
}
}
\end{tikzpicture}
\bigskip

This is the first example:
\begin{itemize}
\item \begin{tikzpicture}
\genealogytree[template=symbol nodes]{
child{
 gm pf cf
 child{gm pf cf cm}
 child{gm pf cm c- cm}
}
}
\end{tikzpicture}
\end{itemize}
\end{verbatim}

Now follows the second example:\par\smallskip
\begin{genealogypicture}[template=tiny boxes]{
child{
 gm pf c-
 child{gm pf c-c-}
 child{gm pf c-c-c-}
}
}
\end{genealogypicture}

Now follows the second example:
\begin{itemize}
\item \begin{genealogypicture}[template=tiny boxes]{
child{
 gm pf c-
 child{gm pf c-c-}
 child{gm pf c-c-c-}
}
}
\end{genealogypicture}
\end{itemize}

To externalize the diagrams, the document has to be set up as described in the previous subsection. Further, both diagrams have to be marked for externalization:

- Replace \texttt{tikzpicture} by \texttt{extikzpicture} and add a \textit{unique name} as additional parameter.

- Replace \texttt{genealogypicture} \texttt{P.57} by \texttt{exgenealogypicture} \texttt{P.57} and add a \textit{unique name} as additional parameter.

By default, these \textit{unique names} are the names of the external files inside an \texttt{external} sub-directory. Depending on the operation system, the sub-directory \texttt{external} may have to be generated manually.
This is the first externalized example:
\begin{extikzpicture}{first_example}
\genealogytree[template=symbol nodes]{
  \child{
    gm pf cf
    \child{gm pf cf cm}
    \child{gm pf cm c- cm}
  }
}
\end{extikzpicture}

Now follows the second externalized example:
\begin{exgenealogypicture}{second_example}{template=tiny boxes}
\child{
  g-p-c-
  \child{g-p-c-c-}
  \child{g-p-c-c-c-}
}
\end{exgenealogypicture}

This is the first externalized example:
\begin{extikzpicture}{first_example}
\genealogytree[template=symbol nodes]{
  \child{
    gm pf cf
    \child{gm pf cf cm}
    \child{gm pf cm c- cm}
  }
}
\end{extikzpicture}

% The next example is always compiled because of ‘!’ :
\begin{exgenealogypicture}{second_example}{template=tiny boxes}[!]
\child{
  g-p-c-
  \child{g-p-c-c-}
  \child{g-p-c-c-c-}
}
\end{exgenealogypicture}

After the diagrams are generated, they are compiled only again, if the diagram content changes. Changes caused by global settings will not be recognized.

To force recreation, just delete the external files. Another way is to add an exclamation mark as an option.

More details about controlling the externalization process are found in [4].
2.7 Tutorial: Conversion to Pixel Images

This tutorial is somewhat off-topic, because it considers the conversion of vector images to pixel images and is not directly related to genealogy trees. On the other hand, the need for conversion of such diagrams arises often, e.g. for third-party photo quality printing, web content, and import to pixel-focused software.

- This tutorial only considers the conversion from PDF (vector image) to PNG (pixel image). Further conversions to e.g. JPEG can be adapted or easily done from PNG to JPEG with many available tools.
- This presentation is not exhaustive. It only gives a short glimpse of some selected options for the conversion tools.

2.7.1 Command Line Conversion with MuPDF

Here, we assume to already have a vector image file example.pdf which has to be converted to PNG.

First, MuPDF\(^2\) has to be installed on the system, if it not already is. It provides a command line tool mutool for conversion which is not necessarily allocatable by the standard system path settings.

The following example conversion has to be given on a single line using a command shell (command window) or inside a script:

```
Command line example (a single line)
mutool draw -r 600 -o "example.png" "example.pdf"
```

- Replace mutool by an appropriate adaption (with path) for your system.
- The example input file is "example.pdf" and the example output file is "example.png". 
- Adapt -r 600 to get larger or smaller pixel images. It defines the dots per inch resolution.

2.7.2 Command Line Conversion with Ghostscript

Here, we assume to already have a vector image file example.pdf which has to be converted to PNG.

First, Ghostscript\(^3\) has to be installed on the system, if it not already is. It provides a command line tool gs or gswin32c or gswin64c for conversion which is not necessarily allocatable by the standard system path settings.

The following example conversion has to be given on a single line using a command shell (command window) or inside a script:

```
Command line example (a single line)
gswin64c -dSAFER -dBATCH -dNOPAUSE -sDEVICE=png16m -r600 -dTextAlphaBits=4 -dGraphicsAlphaBits=4 -sOutputFile="example.png" "example.pdf"
```

- Replace gswin64c by an appropriate adaption (with path) for your system.
- The example input file is "example.pdf" and the example output file is "example.png".
- Adapt -r 600 to get larger or smaller pixel images. It defines the dots per inch resolution.

---

\(^2\) mupdf.com  
\(^3\) www.ghostscript.com
2.7.3 Command Line Conversion with ImageMagick

Here, we assume to already have a vector image file `example.pdf` which has to be converted to PNG.

First, ImageMagick\(^4\) has to be installed on the system, if it not already is. It provides a command line tool `convert` for conversion which is typically allocatable by the standard system path settings.

The following example conversion has to be given on a single line using a command shell (command window) or inside a script:

```
Command line example
convert -density 600 -alpha Remove -quality 90 "example.pdf" "example.png"
```

- Replace `convert` by an appropriate adaption for your system, if needed. On Windows, there other programs named `convert` which may lead to conflicts.
- The example input file is "example.pdf" and the example output file is "example.png".
- Adapt `-density 600` to get larger or smaller pixel images. It defines the dots per inch resolution.

2.7.4 Conversion with the 'standalone' Package

Conversion to pixel images can be done using the `standalone` package. Also, this package is designed to create standalone graphics.

By default, the package uses ImageMagick for conversion in the background. See the package documentation for specific considerations for Windows.

```
\documentclass
  \border=2mm,
  \convert={ density=600 -alpha Remove, outext=.png }
\{standalone\}
\usepackage\{all\}\{genealogytree\}
\usepackage\{lmodern\}
\begin\{document\}
\begin\{genealogypicture\}\{template=symbol nodes\}
  parent{
    g{male}
    insert{gtrparent4}
  }
\end\{genealogypicture\}
\end\{document\}
```

- If the document above is compiled with the `-shell-escape` permission, the compiled PDF files is converted to PNG automatically.
- Adapt `density=600` to get larger or smaller pixel images. It defines the dots per inch resolution.

\(^4\)http://www.imagemagick.org
2.7.5 Conversion during Externalization

If externalization with \texttt{tcolorbox} is used, see Section 2.6 on page 49, possibly many PDF vector images are created. With the following hack, an automatic conversion with \texttt{mutool} from MuPDF is added to the external compilation.

\begin{verbatim}
\documentclass{article}
\usepackage[all]{genealogytree}
\usepackage{lmodern}
\tcbEXTERNALIZE

\makeatletter
\appto{\tcbexternal@corecompile}{\%
  \ShellEscape{\%
    \"mutool\"
    draw
    -r 600
    -o \"\tcbexternal@job@name.png\"
    \"\tcbexternal@job@name.pdf\"
  }\%}
\makeatother

\begin{document}

\begin{exgenealogypicture}![\!]{export}[template=symbol nodes]
  parent{
    g{male}
    insert{gtrparent4}
  }
\end{exgenealogypicture}

\end{document}
\end{verbatim}

- If the document above is compiled with the \texttt{--shell-escape} permission, all externalized graphics (here: one) are converted to PNG automatically.
- Replace "\texttt{mutool}" by an appropriate adaption (possibly with path) for your system.
- Adapt \texttt{-r 600} to get larger or smaller pixel images. It defines the dots per inch resolution.
- Note that this is a hack of internals of \texttt{tcolorbox}. This hack may become useless in the future. Especially, for a single image, the \texttt{standalone} package should be preferred.
3 Genealogy Tree Macros

3.1 Creating a Genealogy Tree

\genealogytree[(options)]{tree contents}

This is the main genealogy tree drawing macro of the package. The \textit{tree contents} has to obey to the tree grammar rules documented in Chapter 4 on page 63. The \textit{options} control how the drawing is done. These \textit{options} are \texttt{pgf} keys with the key tree path /\texttt{gtr}/ and they are described in the following. The actual drawing is done with help of the TikZ package. Therefore, every \genealogytree has to be placed into a \texttt{tikzpicture} environment. It is possible to put several \genealogytree macros into the same \texttt{tikzpicture} and interconnect them.

```latex
\begin{tikzpicture}
\genealogytree[template=signpost]
{parent{
g[male]{proband}
c[female]{sister}
c[male]{brother}
p[male]{father}
p[female]{mother}
}}
\end{tikzpicture}
```

Detailed information about the genealogy tree grammar is found in Chapter 4 on page 63. The short version is that a genealogy tree can have three types of nodes:

- \texttt{c} nodes are \textit{child} nodes to a family,
- \texttt{p} nodes are \textit{parent} nodes to a family,
- \texttt{g} nodes are usually \textit{child} nodes to one family and \textit{parent} nodes another family or even several families. Here, \texttt{g} can be memorized as \textit{genealogy} node.

A \textit{family} is a set of \textit{parent} and \textit{child} nodes which has to contain exactly one \textit{genealogy} node (\texttt{g} node). All nodes of a family are interconnected with an edge set. In contrast to ordinary tree structures where an edge connects one node to another node, here, an edge connects a node to a family.

A genealogy tree can have following types of families:

- \texttt{parent}: A \textit{parent} family may contain other \textit{parent} families. Trees with this construction grow into \textit{ancestor direction}.
- \texttt{child}: A \textit{child} family may contain other \textit{child} families or \textit{union} families. Trees with this construction grow into \textit{descendant direction}.
- \texttt{union}: A \textit{union} ties a second child-type family to a \texttt{g} node as parent of this family.
- \texttt{sandclock}: A \textit{sandclock} connects ancestors and descendants starting from a single proband.
\begin{tikzpicture}
\genealogytree[template=signpost]
{
parent{
\g[male]{proband}
\c[female]{sister}
\c[male]{brother}
parent{
\g[male]{father}
\p[male]{grandfather}
\p[female]{grandmother}
}
\p[female]{mother}
}
}\end{tikzpicture}

\genealogytreeinput[(options)]{(file name)}
Uses the content of the file denoted by \textit{(file name)} to create a \genealogytree P.55 with the given \textit{(options)}. See Section 15.1 on page 357 for the file of the following example.

\begin{tikzpicture}
\genealogytreeinput[template=signpost]{example.option.graph}
\end{tikzpicture}
\begin{genealogypicture}[(options)]
⟨tree content⟩
\end{genealogypicture}

This is a shortcut combination of one \genealogytree P.55 inside a \tikzpicture. For ⟨options⟩ and ⟨tree contents⟩ see \genealogytree P.55. This environment allows more compact source code, but one cannot combine several trees and adding additional \texttt{TikZ} commands has to be done by \texttt{/gtr/tikzpicture} P.114 or \texttt{/gtr/after tree} P.115.

\begin{genealogypicture}
[template=signpost]
parent{
  g[male]{proband}
  c[female]{sister}
  c[male]{brother}
  p[male]{father}
  p[female]{mother}
}
\end{genealogypicture}

\begin{exgenealogypicture}[(externalization options)]{(name)}[(options)]
⟨tree content⟩
\end{exgenealogypicture}

This is an externalized version of \genealogypicture using the \texttt{external} library of the package \texttt{tcolorbox} [4]. The picture is drawn by automatic compilation of an external file denoted by ⟨name⟩ (usually prefixed by a directory or string). Afterwards, the created pdf image is included into the main document. As long as the ⟨tree contents⟩ and the ⟨options⟩ are not changed, the external file is not compiled again which saves overall compilation time. The process can be controlled by ⟨externalization options⟩, see [4]. For a detailed application example, see Section 2.6 on page 49.
3.2 Using Tree Options

\texttt{\textbackslash gtrset\{⟨options⟩\}}

Sets ⟨options⟩ for every following \texttt{\genealogytree} inside the current \TeX\ group. These ⟨options⟩ are \texttt{pgf} keys with the key tree path /\texttt{gtr/} and they are described in the following.

\begin{verbatim}
\begin{tikzpicture}
\genealogytree
{ parent{
  g[male]{proband}
  c[female]{sister}
  c[male]{brother}
  p[male]{father}
  p[female]{mother}
}
}
\end{tikzpicture}
\end{verbatim}

Another important field of application for \texttt{\gtrset} is to create own styles for later usage.

\begin{verbatim}
\gtrset{mytree/.style={
  template=signpost,
  box={colback=yellow!20},
  edges={swing,no background,
         foreground=yellow!50!black},
}}
\begin{tikzpicture}
\genealogytree[mytree]% own style
{ parent{
  g[male]{proband}
  c[female]{sister}
  c[male]{brother}
  p[male]{father}
  p[female]{mother}
}
}
\end{tikzpicture}
\end{verbatim}

\texttt{\textbackslash gtrkeysgappto\{⟨hook⟩\}\{⟨key list⟩\}}

Auxiliary macro which globally appends a ⟨key list⟩ (options) to a ⟨hook⟩ macro which may already contain a key list.

\texttt{\textbackslash gtrkeysgappto\{⟨hook⟩\}\{⟨key list⟩\}}

Auxiliary macro which appends a ⟨key list⟩ (options) to a ⟨hook⟩ macro which may already contain a key list.
3.3 Accessing Information inside Nodes

Inside the node content, there are several processing informations available which can be used for debugging or steering the output. Also see Section 11.4 on page 265 for displaying these values.

\textbf{\texttt{\gtrnodetype}}

Holds the node type \texttt{g}, \texttt{p}, or \texttt{c}.

\textbf{\texttt{\gtrnodeid}}

Holds the /\texttt{gtr/id}\textsuperscript{P.92} value of the node.

\textbf{\texttt{\gtrnodenumber}}

Holds the internal node number.

\textbf{\texttt{\gtrnodefamily}}

Holds the internal family number this node belongs to.

\textbf{\texttt{\gtrnodelevel}}

Holds the tree level number this node belongs to.

\begin{tikzpicture}
\genealogytree[
  template=signpost,
  level size=2cm,
  content interpreter content={
    type & \gtrnodetype\\
    id & \gtrnodeid\\
    number & \gtrnodenumber\\
    family & \gtrnodefamily\\
    level & \gtrnodelevel\\
  }]

  parent{
    g[male,id=Abc]{}
    c[female]{}
    c[male]{}
  parent{
    g[male,id=Bob]{}
    p[male,id=Jim]{}
    p[female]{}
  }
  p[female]{}
}
\end{tikzpicture}

\textbf{\texttt{\gtrifnodeid{\texttt{true}}}{\texttt{false}}}}

Expands to \texttt{\texttt{true}}, if /\texttt{gtr/id}\textsuperscript{P.92} was set, and to \texttt{false} otherwise.

\textbf{\texttt{\gtrifgnode{\texttt{true}}}{\texttt{false}}}}

Expands to \texttt{\texttt{true}}, if the node type is \texttt{g}, and to \texttt{false} otherwise.

\textbf{\texttt{\gtrifcnode{\texttt{true}}}{\texttt{false}}}}

Expands to \texttt{\texttt{true}}, if the node type is \texttt{c}, and to \texttt{false} otherwise.

\textbf{\texttt{\gtrifpnode{\texttt{true}}}{\texttt{false}}}}

Expands to \texttt{\texttt{true}}, if the node type is \texttt{p}, and to \texttt{false} otherwise.
\texttt{\textbackslash gtrifroot\{\textbf{true}\}\{\textbf{false}\}}

Expands to \textit{(true)}, if the node is the root node of a \textit{parent} tree or of a \textit{child} tree, and to \textit{(false)} otherwise. For a \textit{sandclock} tree, it expands always to \textit{(false)}.

\texttt{\textbackslash gtrifleaf\{\textbf{true}\}\{\textbf{false}\}}

Expands to \textit{(true)}, if the node type is \textit{c} or \textit{p} or if the node is the root node of a \textit{parent} tree or of a \textit{child} tree, and to \textit{(false)} otherwise. Note that \textit{(false)} is set for all \textit{g} nodes with the root node as an exception, even if the node does not have a parent or a child. Also note that a root node is intentionally considered to be a leaf also.

\texttt{\textbackslash gtrifchild\{\textbf{true}\}\{\textbf{false}\}}

Expands to \textit{(true)}, if the node type is \textit{c} or \textit{g} in a \textit{parent} family or is \textit{g} but not root in a \textit{child} family, and to \textit{(false)} otherwise.

\texttt{\textbackslash gtrifparent\{\textbf{true}\}\{\textbf{false}\}}

Expands to \textit{(true)}, if the node type is \textit{p} or is \textit{g} in a \textit{child} family or is \textit{g} but not root in a \textit{parent} family, and to \textit{(false)} otherwise.

\texttt{\textbackslash gtrifleafchild\{\textbf{true}\}\{\textbf{false}\}}

Expands to \textit{(true)}, if \texttt{\textbackslash gtrifleaf} and \texttt{\textbackslash gtrifchild} are both true, and to \textit{(false)} otherwise.

\texttt{\textbackslash gtrifleafparent\{\textbf{true}\}\{\textbf{false}\}}

Expands to \textit{(true)}, if \texttt{\textbackslash gtrifleaf} and \texttt{\textbackslash gtrifparent} are both true, and to \textit{(false)} otherwise.

\begin{tikzpicture}
\genealogytree
\trust thrown away
\end{tikzpicture}
3.4 Auxiliary Tools

\gtrautosizebox{⟨width⟩}{⟨height⟩}{⟨contents⟩}

Puts the ⟨contents⟩ into a horizontal box and resizes this box to fit into a rectangular frame of given ⟨width⟩ and ⟨height⟩. The resizing keeps the aspect ratio intact. The star (*) variant does shrinking only and keeps small boxes unchanged.

If ⟨width⟩ is missing or is not greater than 0pt, then only the ⟨height⟩ is considered for scaling.

If ⟨height⟩ is missing or is not greater than 0pt, then only the ⟨width⟩ is considered for scaling.

If ⟨width⟩ and ⟨height⟩ are both missing or both are not greater than 0pt, the box is kept unscaled.

A \tikzpicture environment with given TikZ ⟨options⟩ is put into \gtrautosizebox or \gtrautosizebox* with given ⟨width⟩ and ⟨height⟩.
4

Graph Grammar

4.1 Graph Structure

In graph theory, a graph is defined by a set of vertices (or nodes) and a set of edges connecting these vertices. A general graph structure would certainly allow to depict genealogy data, but building and displaying such a general graph is not supported by this \LaTeX\ package.

An ordinary tree structure is a specialized (directed) graph which has a root node as starting point. Every node may have one or more descendant nodes. In this relationship, the first node is called parent and the second is called child. Such tree structures are heavily used for many applications. Also, there exist excellent \LaTeX\ packages to display such structures. Such tree structures can also be used for several kinds of genealogy type diagrams, but, by definition, they miss the core element of genealogy: the family consisting of two parents and several childs.

The graph structure used by the genealogytree package is intended to put the family as a set of parent and child nodes in the foreground. Every family is allowed to have more than one parent and more than one child. The interconnection between the parent and child nodes of a family is not considered to be bilateral between pairs of nodes, but to be multilateral between all nodes of the family. From the idea, a node is connected not to another node, but to one or more families. Still, there apply strong restrictions on the set of possible graphs, because graphs have to be reasonable processable and presentable. The restrictions are realized by the following graph grammar. In the following, the resulting graphs are called genealogy trees.

A family is a set of parent and child nodes which has to contain exactly one genealogy node.
(g node). Therefore, a family can have three types of nodes:
  - c nodes are child nodes to a family, see Section 4.6 on page 73,
  - p nodes are parent nodes to a family, see Section 4.7 on page 73,
  - g nodes are usually child nodes to one family and parent nodes another family or even several families, see Section 4.8 on page 73.

A genealogy tree can have following types of families:
  - parent: A parent family may contain other parent families. Trees with this construction grow into ancestor direction, see Section 4.2 on page 65,
  - child: A child family may contain other child families or union families. Trees with this construction grow into descendant direction, see Section 4.3 on page 67,
  - union: A union ties a second child-type family to a g node as parent of this family, see Section 4.4 on page 69,
  - sandclock: A sandclock connects ancestors and descendants starting from a single proband, see Section 4.5 on page 71.

As will be documented on the following pages, the graph input data is strongly hierarchically organized. Each element is allowed to have specific sub-elements. The starting point is the root element which is the top element inside genealogytree. The root of a parsable graph is one of the following:
  - a parent (for ancestor graphs), see Section 4.2 on the facing page,
  - a child (for descendant graphs), see Section 4.3 on page 67,
  - a sandclock (for mixed ancestor/descendant graphs), see Section 4.5 on page 71.
4.2 Subgraph 'parent'

A parent subgraph is a family where the g node acts as a child. This family may have arbitrary child and parent leaves. Also, this family may have arbitrary parent subgraphs.

**Syntax for a 'parent' subgraph**

```
parent[⟨parent options⟩]{
  g[⟨node options⟩]{⟨node content⟩} mandatory; exactly once
  c[⟨node options⟩]{⟨node content⟩} optional; zero or many times
  p[⟨node options⟩]{⟨node content⟩} optional; zero or many times
  parent[⟨parent options⟩]{⟨subtree content⟩} optional; zero or many times
  input{⟨file name⟩} optional; zero or many times
  insert{⟨csname⟩} optional; zero or many times
}
'g', 'c', 'p', 'parent', 'input' may appear in arbitrary order.
```

The optional ⟨parent options⟩ can be settings for the current family or the whole subgraph. See Chapter 5 on page 77 and especially Section 5.6 on page 104 and Section 5.7 on page 107 for feasible options.

```latex
\begin{tikzpicture}
genealogytree[template=signpost,
options for node={pA}{box={colback=red!20!white}}]
{
  parent{
    c[id=pB]{B\hspace{2em} (child)}
    g[id=pA]{A\hspace{2em} (proband)}
    c[id=pC]{C\hspace{2em} (child)}
    c[id=pD]{D\hspace{2em} (child)}
    p[id=pE]{E\hspace{2em} (parent)}
    p[id=pF]{F\hspace{2em} (parent)}
  }
}
\end{tikzpicture}
```
The `\gtrparserdebug` command helps to detect structural errors. Here, we get:

\gtrparserdebug{
  parent{
    c[id=pB]{B\(child\)}
    g[id=pA]{A\(proband\)}
    c[id=pC]{C\(child\)}
    c[id=pD]{D\(child\)}
    p[id=pE]{E\(parent\)}
    p[id=pF]{F\(parent\)}
  }
}

<table>
<thead>
<tr>
<th>Genealogytree Parser Debugger</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Start:</strong> Parent Family 1, Level 1</td>
</tr>
<tr>
<td><strong>Child:</strong> Individual 1, Family 1, Level 0</td>
</tr>
<tr>
<td>Options: id=pB</td>
</tr>
<tr>
<td>Content: B(child)</td>
</tr>
<tr>
<td><strong>Child:</strong> Individual 2, Family 1, Level 0</td>
</tr>
<tr>
<td>Options: id=pA</td>
</tr>
<tr>
<td>Content: A(proband)</td>
</tr>
<tr>
<td><strong>Child:</strong> Individual 3, Family 1, Level 0</td>
</tr>
<tr>
<td>Options: id=pC</td>
</tr>
<tr>
<td>Content: C(child)</td>
</tr>
<tr>
<td><strong>Child:</strong> Individual 4, Family 1, Level 0</td>
</tr>
<tr>
<td>Options: id=pD</td>
</tr>
<tr>
<td>Content: D(child)</td>
</tr>
<tr>
<td><strong>Parent:</strong> Individual 5, Family 1, Level 1</td>
</tr>
<tr>
<td>Options: id=pE</td>
</tr>
<tr>
<td>Content: E(parent)</td>
</tr>
<tr>
<td><strong>Parent:</strong> Individual 6, Family 1, Level 1</td>
</tr>
<tr>
<td>Options: id=pF</td>
</tr>
<tr>
<td>Content: F(parent)</td>
</tr>
<tr>
<td><strong>End:</strong> Parent Family 1, Level 1</td>
</tr>
</tbody>
</table>

End of Genealogytree Parser Debugger
4.3 Subgraph 'child'

A **child** subgraph is a family where the **g** node acts as a parent. This family may have arbitrary child and parent leaves. Also, this family may have arbitrary **child** and **union** subgraphs.

**Syntax for a 'child' subgraph**

```
\begin{tikzpicture}
\genealogytree[template=signpost, options for node={pA}{box={colback=red!20!white}}]
{
  child{
    g[id=pA]{A\(proband\)}
    p[id=pB]{B\(parent\)}
    c[id=pC]{C\(child\)}
    c[id=pD]{D\(child\)}
    c[id=pE]{E\(child\)}
  }
}\end{tikzpicture}
```

\begin{itemize}
  \item \texttt{\begin{tikzpicture}\genealogytree[template=signpost, options for node={pA}{box={colback=red!20!white}}]}
  \item \texttt{\end{tikzpicture}}
\end{itemize}

The optional \texttt{(child options)} can be settings for the current family or the whole subgraph. See Chapter 5 on page 77 and especially Section 5.6 on page 104 and Section 5.7 on page 107 for feasible options.
\gtrparserdebug \ P. 250 can help to detect structural errors. Here, we get:

```
gtrparserdebug{
    child{
        g[id=pA]{A\(proband\)}
        p[id=pB]{B\(parent\)}
        c[id=pC]{C\(child\)}
        c[id=pD]{D\(child\)}
        c[id=pE]{E\(child\)}
    }
}
```

Genealogytree Parser Debugger

Start: Child Family 1, Level 0

Parent: Individual 1, Family 1, Level 0
Options: id=pA
Content: A\(proband\)

Child: Individual 3, Family 1, Level -1
Options: id=pC
Content: C\(child\)

Child: Individual 4, Family 1, Level -1
Options: id=pD
Content: D\(child\)

Child: Individual 5, Family 1, Level -1
Options: id=pE
Content: E\(child\)

End: Child Family 1, Level 0

End of Genealogytree Parser Debugger
4.4 Subgraph 'union'

A **union** subgraph is a family without a g node. The g node (parent) is inherited from an embedding child family. A union family may have arbitrary child and parent leaves. Also, this family may have arbitrary child subgraphs.

Syntax for a 'union' subgraph

\begin{tikzpicture}
\genealogytree[template=signpost, options for node={pA}{box={colback=red!20!white}}]{
  child{
    p[id=pB]{B\ (parent)}
    g[id=pA]{A\ (proband)}
    c[id=pC]{C\ (child)}
    union{
      p[id=pD]{D\ (parent)}
      c[id=pE]{E\ (child)}
    }
  }
}
\end{tikzpicture}

The optional (child options) can be settings for the current family or the whole subgraph. See Chapter 5 on page 77 and especially Section 5.6 on page 104 and Section 5.7 on page 107 for feasible options. As a special case for unions, note that the g node of the embedding child family will not be affected by these options.
can help to detect structural errors. Here, we get:

```
gtreparsedebug
child{
  p[id=pB]{B\(parent\)}
  g[id=pA]{A\(proband\)}
  c[id=pC]{C\(child\)}
union{
  p[id=pD]{D\(parent\)}
  c[id=pE]{E\(child\)}
}
}
```

Genealogytree Parser Debugger

Start: Child Family 1, Level 0

Parent: Individual 1, Family 1, Level 0
Options: id=pB
Content: B\(parent\)

Parent: Individual 2, Family 1, Level 0
Options: id=pA
Content: A\(proband\)

Child: Individual 3, Family 1, Level -1
Options: id=pC
Content: C\(child\)

Start: Union Family 2, Level 0

Parent: Individual 4, Family 2, Level 0
Options: id=pD
Content: D\(parent\)

Child: Individual 5, Family 2, Level -1
Options: id=pE
Content: E\(child\)

End: Union Family 2, Level 0
End: Child Family 1, Level 0
End of Genealogytree Parser Debugger
4.5 Subgraph 'sandclock'

A *sandclock* subgraph is a family without a *g* node. The *g* node (child) is inherited from an embedded *child* family. A *sandclock* family may have arbitrary child and parent leaves. Also, this family must have at least one *child* subgraph and may have arbitrary *parent* subgraphs.

### Syntax for a 'sandclock' subgraph

```latex
\begin{tikzpicture}
genealogytree[template=signpost,
options for node={pA}{box={colback=red!20!white}}]
{ sandclock{
  c[id=pB]{B\(child\)}
  child{
    g[id=pA]{A\(proband\)}
    c[id=pa]{a\(child\)}
    c[id=pb]{b\(child\)}
  }
  p[id=pC]{C\(parent\)}
  parent{
    g[id=pD]{D\(parent\)}
    c[id=pE]{E\(child\)}
    p[id=pF]{F\(parent\)}
  }
}
\end{tikzpicture}
```

'c', 'p', 'child', 'parent', 'input' may appear in arbitrary order.
Genealogytree Parser Debugger

Start: Sandclock Family 1, Level 1

Child: Individual 1, Family 1, Level 0
Options: id=pB
Content: B\(child)"

Start: Child Family 2, Level 0

Parent: Individual 2, Family 2, Level 0
Options: id=pA
Content: A\(proband)"

Child: Individual 3, Family 2, Level -1
Options: id=pa
Content: a\(child)"

Child: Individual 4, Family 2, Level -1
Options: id=pb
Content: b\(child)"

Parent: Individual 5, Family 2, Level 0
Options: id=pX
Content: X\(partner)"

End: Child Family 2, Level 0

Parent: Individual 6, Family 1, Level 1
Options: id=pC
Content: C\(parent)"

Start: Parent Family 3, Level 2

Child: Individual 7, Family 3, Level 1
Options: id=pD
Content: D\(parent)"

Child: Individual 8, Family 3, Level 1
Options: id=pE
Content: E\(child)"

Parent: Individual 9, Family 3, Level 2
Options: id=pF
Content: F\(parent)"

End: Parent Family 3, Level 2

End: Sandclock Family 1, Level 1

End of Genealogytree Parser Debugger
4.6 Node 'c'

The c (child) node is a leaf node which is child to a family.

Syntax for a 'c' node

```
c[(node options)]{(node content)}
```

For the optional (node options), see Chapter 5 on page 77 and especially Section 5.5 on page 95.

The (node content) can be any text to be displayed inside the node. This (node content) can also be processed before displaying, see Chapter 6 on page 137 and especially Chapter 7 on page 161 for database processing. Also, the (node content) can be completely ignored for processing. In this case, one can use c{} or even shorter c(token) for the node.

```
\begin{genealogypicture}[
  template=formal graph,
  content interpreter content=
  \{n_{\gtrnodenumber}\},
]\n  child{ g-p-c-c-
    child{ p-g-
      child{ p-g-c-c-c- }
      c-c-
    }
    c-c-
  }
\end{genealogypicture}
```

4.7 Node 'p'

The p (parent) node is a leaf node which is parent to a family.

Syntax for a 'p' node

```
p[(node options)]{(node content)}
```

For the optional (node options), see Chapter 5 on page 77 and especially Section 5.5 on page 95. For (node content), see Section 4.6 on page 73.

4.8 Node 'g'

The g (genealogy) node is an interconnecting individual which is member of at least two families. For one family it is a child, for another one it is a parent.

Syntax for a 'g' node

```
g[(node options)]{(node content)}
```

For the optional (node options), see Chapter 5 on page 77 and especially Section 5.5 on page 95. For (node content), see Section 4.6 on page 73.
4.9 Data 'input'

Feasible subgraphs may be read from external files using the input command at places where such subgraphs are expected.

**Syntax for data 'input'**

```
input{(file name)}
```

The following example reads a parent subgraph from a file. See Section 15.1 on page 357 for the file contents.

\begin{verbatim}
\begin{tikzpicture}
\genealogytree[template=signpost]
{
parent{
 g{Puppy}
 input{example.option.graph}
 parent
 {
  g{female}{Nanny}
  p{male}{Pa}
  p{female}{Ma}
 }
 }
}
\end{tikzpicture}
\end{verbatim}
4.10 Control Sequence 'insert'

Feasible subgraphs may be inserted from control sequences using the \texttt{insert} command at places where such subgraphs are expected.

\begin{center}
\begin{tikzpicture}
\genealogytree[template=formal graph]
\end{tikzpicture}
\end{center}

\textbf{Syntax for data 'insert'}

\begin{verbatim}
insert\{⟨csname⟩\}
\end{verbatim}

\(⟨csname⟩\) is the name of a control sequence without the leading backslash \texttt{'\textbackslash'}\. This control sequence has to be a parameterless macro whose replacement text is a feasible subgraph.

The following example creates such a macro \texttt{\textbackslash mytest}:

\begin{verbatim}
\newcommand{\mytest}{
  \parent{ g{x_1}
    \parent{ g{x_2} \parent{ g{x_3} p{x_4} } \parent{ g{x_5} p{x_6} p{x_7} }\}
  \} \insert{mytest} 
\}
\begin{tikzpicture}
\genealogytree{template=formal graph}
\{ 
  \parent{ g{a_1}
    \parent{ g{a_3}
      \insert{mytest} \insert{mytest} 
    }
    \insert{mytest}
  } 
\}
\end{tikzpicture}
\end{verbatim}
For the \textit{options} in \texttt{genealogytree $\cdot$ P. 55}, \texttt{genealogypicture $\cdot$ P. 57}, and \texttt{gtrset $\cdot$ P. 58}, keys with \texttt{pgf} syntax can be applied as documented in the following. The key tree path \texttt{/gtr/} is not to be used inside these macros. It is easy to add your own style keys using the syntax for \texttt{pgf} keys, see [5].

Some of the following examples use a standard graph file which is documented in Section 15.1 on page 357.

\section{Option Priorities

This section can be skipped on first reading. Option priorities are more or less natural. This section can be consulted later in case of doubt.

Options for the graph drawing can be set at several spots inside the code using the \texttt{pgf} key-value syntax:

- as parameter of \texttt{gtrset $\cdot$ P. 58} for setting (global) options,
- as optional parameter of \texttt{genealogytree $\cdot$ P. 55} or \texttt{genealogypicture $\cdot$ P. 57},
- as optional parameter of a family identifier like \texttt{parent} or \texttt{child},
- as optional parameter of a node identifier like \texttt{g}, \texttt{p}, or \texttt{c}.

Depending on where the options are given, they are treated with different priority. If an option is given several times with the same priority, the last one wins.

- For options like \texttt{/gtr/pivot $\cdot$ P. 97}, an option setting with higher priority overwrites an option setting with lower priority.
- For options like \texttt{/gtr/box $\cdot$ P. 98}, an option setting with higher appends to an option setting with lower priority. Thus, \texttt{/gtr/box $\cdot$ P. 98} options which are not overwritten, stay active.
5.1.1 Option Priorities for Nodes

Example: Priorities for setting box options to a node with id=A

```
\gtrset{
  ...
  \genealogytree[
    options for node={A}{box={...}}, % priority (1) highest
    options for family={fam_a}{box={...}}, % priority (5)
    options for subtree={fam_a}{box={...}}, % priority (9)
    level 2/.style={
      node={box={...}}, % priority (3)
      family={box={...}}, % priority (7)
      subtree={box={...}}}, % priority (11)
    level/.code={\ifnum#1=2}\relax\gtrset{
      node={box={...}}, % priority (4)
      family={box={...}}, % priority (8)
      subtree={box={...}}\fi},
    box={...}, % priority (13) lowest
  ...
}\%
{...
  ...
  parent[ id=fam_a, % family with id 'fam_a'
    family={box={...}}, % priority (6)
    subtree={box={...}}, % priority (10)
  ]
  ...}
  ...
  p[id=A, % node with id 'A'
    box={...})\{A} % priority (2)
  ...
  ...
}
```

The priorities for options regarding nodes

1. `\gtr/options for node`\textsuperscript{P.95} has the highest priority. The node has to be identified by a given `\gtr/id`\textsuperscript{P.92}. `\gtr/options for node`\textsuperscript{P.95} should be given using `\gtrset`\textsuperscript{P.58} or as option of `\genealogytree`\textsuperscript{P.55}.

2. Optional parameter of a node identifier like `g`, `p`, or `c`.

3. Option `\gtr/node`\textsuperscript{P.96} inside `\gtr/level n`\textsuperscript{P.110}.

4. Option `\gtr/node`\textsuperscript{P.96} inside `\gtr/level`\textsuperscript{P.109}.

5. `\gtr/options for family`\textsuperscript{P.104}; the family has to be identified by a given `\gtr/id`\textsuperscript{P.92}.

6. `\gtr/family`\textsuperscript{P.105} as optional parameter of a family identifier like `parent` or `child`.

7. Option `\gtr/family`\textsuperscript{P.105} inside `\gtr/level n`\textsuperscript{P.110}.

8. Option `\gtr/family`\textsuperscript{P.105} inside `\gtr/level`\textsuperscript{P.109}.

9. `\gtr/options for subtree`\textsuperscript{P.107}; the subtree has to be identified by a given `\gtr/id`\textsuperscript{P.92}.

10. `\gtr/subtree`\textsuperscript{P.108} as optional parameter of a family identifier like `parent` or `child`.

11. Option `\gtr/subtree`\textsuperscript{P.108} inside `\gtr/level n`\textsuperscript{P.110}.

12. Option `\gtr/subtree`\textsuperscript{P.108} inside `\gtr/level`\textsuperscript{P.109}.

13. Setting as parameter of `\genealogytree`\textsuperscript{P.55} or `\gtrset`\textsuperscript{P.58} has the lowest priority.
5.1.2 Option Priorities for Families

Example: Priorities for setting edges options to a family with id=fam_a

\gtrset{
    ...
    \genealogytree{
        options for family={fam_a}{edges={...}}, % priority (1) highest
        options for subtree={fam_a}{edges={...}}, % priority (5)
        level 2/.style={family={edges={...}}}, % priority (3)
        subtree={edges={...}}}, % priority (7)
    \gtrset{family={edges={...}}, % priority (4)
        subtree={edges={...}}}
    \fi,
    edges={...}, % priority (9) lowest
    ...
}{
    parent[ id=fam_a, % family with id 'fam_a'
        family={edges={...}}, % priority (2)
        subtree={edges={...}}, % priority (6)
    ]
    {
        ...
    }
    ...
}

The priorities for options regarding families

1. /gtr/options for family* has the highest priority. The family has to be identified by a given /gtr/id*. /gtr/options for family* should be given using \gtrset* or as option of \genealogytree*.
2. Optional /gtr/family* parameter of a family identifier like parent or child.
3. Option /gtr/family* inside /gtr/level*.
4. Option /gtr/family* inside /gtr/level*.
5. /gtr/options for subtree*; the subtree has to be identified by a given /gtr/id*.
6. Optional /gtr/subtree* parameter of a family identifier like parent or child.
7. Option /gtr/subtree* inside /gtr/level*.
8. Option /gtr/subtree* inside /gtr/level*.
9. Setting as parameter of \genealogytree* or \gtrset* has the lowest priority.
5.2 Graph Growth Setting (Time Flow)

A genealogy tree may grow in one of four directions. This /gtr/timeflow setting is valid for the whole graph, but two graphs with different growth setting may be joined together.

/gtr/timeflow=(direction)  (no default, initially down)

The /gtr/timeflow key controls the growing direction of a given graph. It is always used to place the generations according to this value. If the (direction) is set to down, a child graph will grow down, but a parent graph will grow up. Feasible values are:

- down
- up
- left
- right

```latex
\begin{tikzpicture}
genealogytree [template=signpost,timeflow=down]
{input{example.option.graph}}
\node at ([xshift=4cm]GmDo1956) (past) {Past};
\draw[very thick,->] (past) -- +(0,-2) node[below] {Future};
\end{tikzpicture}
```

See Section 15.1 on page 357 for the included example graph file.
\begin{tikzpicture}
genealogytree[template=signpost,timeflow=up]
{input{example.option.graph}}
\node at ([xshift=4cm]GmDo1956) (past) {Past};
\draw[very thick,->] (past) -- +(0,2) node[above] {Future};
\end{tikzpicture}
\begin{tikzpicture}
genealogytree[template=signpost, timeflow=right, node size=1.2cm, level size=3cm]
{input{example.option.graph}}
\node at ([yshift=-1.5cm]GmDo1956) (past) {Past};
\draw[very thick, ->] (past) -- +(2,0) node[right] {Future};
\end{tikzpicture}

Past \rightarrow Future
5.3 Graph Geometry

The following geometry settings are usually set for the whole graph, but they can be set for every \texttt{/gtr/level} separately. Inside a level, they are fixed.

\texttt{/gtr/level distance=(length)} (no default, initially 5mm)

The given \texttt{(length)} defines the distance between two following generations. This distance can be set in dependency of the \texttt{/gtr/level}.

The \texttt{/gtr/level distance} can be specified for individual level numbers, e.g.

\begin{verbatim}
\gtrset{
    level 0/.style={level distance=5mm},
    level -1/.style={level distance=10mm}
}
\end{verbatim}
/gtr/level size=(length) (no default, initially 3.5cm)

The given (length) defines one dimension of a node.
- If /gtr/timeflow → P.80 is up or down, then /gtr/level size sets the height of a node.
- If /gtr/timeflow → P.80 is left or right, then /gtr/level size sets the width of a node.

The /gtr/level size be set in dependency of the /gtr/level → P.109.

Some actual node implementations may not respect the given /gtr/level size. Note that the placement algorithm ignores deviations and assumes that the restrictions hold.
The given \textit{(length)} defines one dimension of a node.

- If \texttt{/gtr/timeflow \rightarrow P.80} is \textit{up} or \textit{down}, then \texttt{/gtr/level size \rightarrow P.84} sets the width of a node.
- If \texttt{/gtr/timeflow \rightarrow P.80} is \textit{left} or \textit{right}, then \texttt{/gtr/level size \rightarrow P.84} sets the height of a node.

The \texttt{/gtr/node size} can be set in dependency of the \texttt{/gtr/level \rightarrow P.109}. Note that the \texttt{/gtr/node size} may be ignored by nodes boxes which set the width individually or depending from the content width.

If the size should be changed for an individual node, use \texttt{/gtr/box \rightarrow P.98} instead of \texttt{/gtr/node size}:

Some actual node implementations may not respect the given \texttt{/gtr/node size}. The placement algorithm accepts deviations and calculates positions accordingly.
/gtr/node size from=$\langle minlength \rangle$ to $\langle maxlength \rangle$ (no default, initially 2.5cm to 2.5cm)

Sets the /gtr/node size→P.85 in a flexible way ranging from $\langle minlength \rangle$ to $\langle maxlength \rangle$. The actual size of a node is determined by the node content. A node will be enlarged up to $\langle maxlength \rangle$ before the content font size is allowed to shrink. Note that the /gtr/node size from may be ignored by nodes boxes which set the width individually or depending from the content width.

/gtr/child distance in parent graph=$\langle length \rangle$ (no default, initially 1mm)

The given $\langle length \rangle$ defines the minimum distance of two children of a family in a parent graph. The /gtr/child distance in parent graph can be set in dependency of the /gtr/level→P.109. For an individual node, this distance can be overruled by setting /gtr/distance→P.96.

\genealogytree...
child distance in parent graph=5mm
{
  parent{
    p{P_1}
    p{P_2}
    g{C_1}
    c[id=A]{A}
    c[id=B]{B}
  }
}
...

P_1  P_2
  /  /
C_1  A   B
  |
  child distance
/gtr/child distance in child graph=⟨length⟩
(no default, initially 2mm)

The given ⟨length⟩ defines the minimum distance of two children of a family in a child graph. The /gtr/child distance in child graph can be set in dependency of the /gtr/level → P.109. For an individual node, this distance can be overruled by setting /gtr/distance → P.96.

\genealogytree[
...  
  child distance in child graph=5mm
  {  
    child{
      g{P_1}
      p{P_2}
      c{C_1}
      c[id=A]{A}
      c[id=B]{B}
    }
  }
...]

/gtr/child distance=⟨length⟩
(no default, style)

This is an abbreviation for setting /gtr/child distance in parent graph → P.86 and /gtr/child distance in child graph to the same ⟨length⟩.
The given \textit{length} defines the minimum distance of two parents of a family in a \textit{parent graph}. The \textit{/gtr/parent distance in parent graph} can be set in dependency of the \textit{/gtr/level} \textsuperscript{-P.109}. For an individual node, this distance can be overruled by setting \textit{/gtr/distance} \textsuperscript{-P.96}.
/gtr/parent distance in child graph = \langle length \rangle  
(no default, initially 1mm)

The given \langle length \rangle defines the minimum distance of two parents of a family in a child graph. The /gtr/parent distance in parent graph → P.88 can be set in dependency of the /gtr/level → P.109. For an individual node, this distance can be overruled by setting /gtr/distance → P.96.

\genealogytree
{ ...
  parent distance in child graph=5mm
  ...
  child{
    g[id=A]{A}
    p[id=B]{B}
    c[C_1]
    c[C_2]
    c[C_3]
  }
} ...

/gtr/parent distance = \langle length \rangle  
(no default, style)

This is an abbreviation for setting /gtr/parent distance in parent graph → P.88 and /gtr/parent distance in child graph to the same \langle length \rangle.
The given \( \text{length} \) defines the minimum distance of two nodes which are not parents or children of the same family. The /gtr/further distance can be set in dependency of the /gtr/level → P.109. For an individual node, this distance can be overruled by setting /gtr/distance → P.96.

\[ \text{\gtrset{ } \text{further distance}=5\text{mm} } \]

\[ \ldots \]
If `/gtr/further distance` is set in level dependency, it is worth to note that this distance is not used for the nodes on the specified layer but for joining the subtrees on the specified layer. In the following example, the distances set on layer 1 and on layer 2 influence different nodes on layer 3.

\begin{verbatim}
\gtrset{
  level 1/.style={further distance=10mm},
  level 2/.style={further distance=5mm},
}
\end{verbatim}
5.4 Identifiers

Identifiers play an important role for semi-automatic processing of graph data. Every node and every family can be marked by an \texttt{/gtr/id} for later reference. If the graph data is exported or generated by a tool, all nodes and families \textit{should} be marked. This allows to manipulate the graph without editing the generated data.

\texttt{/gtr/id=⟨name⟩} (no default, initially empty)

Every node and every family can be marked by a \langle name⟩ using this option. This \langle name⟩ is used by \texttt{/gtr/options for node}, \texttt{/gtr/options for family}, etc, to set options for the specified node or family.

- The \langle name⟩ should be unique inside the \texttt{tikzpicture} environment.
- A Ti\texttt{kZ node} \langle name⟩ is automatically created for later usage.

\begin{tikzpicture}
genealogytree[template=signpost]
{input{example.option.graph}}
\end{tikzpicture}

For example, let us consider the Smith-Doe graph used many times in this document, see Section 15.1 on page 357. Using the identifiers, Jane Doe should be emphasized strongly. Without specific manipulation, the graph data is depicted as follows:

```latex
\begin{tikzpicture}
genealogytree[template=signpost]
{input{example.option.graph}}
\end{tikzpicture}
```

\begin{center}
\begin{tikzpicture}

\begin{scope}
    \node[fill=red] (A) {Grandpa Smith} [yshift=-1cm];
    \node[fill=blue] (B) [below of=A] {Grandma Smith} [yshift=-1cm];
    \node[fill=red] (C) [below of=B] {Grandpa Doe} [yshift=-1cm];
    \node[fill=blue] (D) [below of=C] {Grandma Doe} [yshift=-1cm];
\end{scope}

\begin{scope}[xshift=2.5cm]
    \node[fill=red] (E) {John Smith} [yshift=-1cm];
    \node[fill=blue] (F) [below of=E] {Jane Doe} [yshift=-1cm];
    \node[fill=red] (G) [below of=F] {Uncle Harry} [yshift=-1cm];
\end{scope}

\begin{scope}[xshift=5cm]
    \node[fill=red] (H) {Arthur} [yshift=-1cm];
    \node[fill=blue] (I) [below of=H] {Berta} [yshift=-1cm];
    \node[fill=red] (J) [below of=I] {Charles} [yshift=-1cm];
\end{scope}

\begin{scope}[yshift=-1cm]
    \node (K) [right of=E] {\textcolor{red}{2008}};
    \node (L) [below of=K] {\textcolor{blue}{2010}};
    \node (M) [below of=L] {\textcolor{red}{2014}};
\end{scope}

\end{tikzpicture}
\end{center}
One could inspect the source code in Section 15.1 on page 357 to see the given identifiers. For a large dataset, this may become inconvenient. A good alternative is to use $/gtr/show id$ to overlay the depicted graph with all given $/gtr/id$ values.

\begin{tikzpicture}
\genealogytree[template=signpost,show id]
{input{example.option.graph}}
\end{tikzpicture}

Now, Jane Doe can be emphasized. Note that the id value Jane1982 is also a TikZ node and can be used such.

\begin{tikzpicture}
\genealogytree[template=signpost,show id,\
    options for node={Jane1982}{box={colback=red!50},pivot},\
    options for node={Harr1987}{distance=3.5cm}]
{input{example.option.graph}}
\draw [decorate,decoration={brace,amplitude=5pt,mirror,raise=2pt},\
    line width=1pt,yshift=0pt] (Jane1982.south east) -- (Jane1982.north east) \\
    node [align=center,right=9pt,midway,fill=yellow] {Most important person};
\end{tikzpicture}
The given \text{(text)} is prefixed to every \gtr{id} P.92. This option is intended to be used as part of an option list for a \genealogytree P.55 or genealogypicture P.57. If not used there, it will be set to empty by \genealogytree P.55.

\begin{tikzpicture}
\genealogytree
\[template=signpost, id prefix=xx:, show id\]
{input{example.option.graph}}
\end{tikzpicture}

\gtr{id} suffix=(text) (no default, initially empty) N 2015-06-09

The given \text{(text)} is suffixed to every \gtr{id} P.92. This option is intended to be used as part of an option list for a \genealogytree P.55 or genealogypicture P.57. If not used there, it will be set to empty by \genealogytree P.55.

\begin{tikzpicture}
\genealogytree
\[template=signpost, id suffix=@yy, show id\]
{input{example.option.graph}}
\end{tikzpicture}
5.5 Node Options

The given ⟨options⟩ are set for all nodes with /gtr/id → P. 92 values from the given ⟨id list⟩. If an /gtr/id → P. 92 value is not existing, the setting is silently ignored. The intended spot for using /gtr/options for node is before \genealogytree → P. 55 or inside its option list. Also see Section 5.1.1 on page 78.

\begin{tikzpicture}
\genealogytree
[template=signpost,
 options for node={Arth2008,John1980}{%
     box={interior style={top color=red!30,bottom color=red}}
 }]
{input{example.option.graph}}
\end{tikzpicture}

\begin{tikzpicture}
\genealogytree
[template=signpost,
 noptions for node={Arth2008,John1980}{%
     box={interior style={top color=red!30,bottom color=red}}
 }]
\end{tikzpicture}

Identical to using /gtr/options for node.
The given \( \langle \text{options} \rangle \) are set for all nodes within the current scope. This scope is primarily intended to be a \gtr/level \( \rightarrow \text{P. 109} \) or \gtr/level \( n \rightarrow \text{P. 110} \) definition. For other spots, where \gtr/node is not needed, it may be ignored or directly replaced by its content. Also see Section 5.1.1 on page 78.

\begin{tikzpicture}
\genealogytree{template=signpost,
level 2/.style={
node={box={interior style={top color=red!30,bottom color=red}}} }
}{input{example.option.graph}}
\end{tikzpicture}

\begin{tikzpicture}
\genealogytree{template=signpost,options for node={GpDo1955}{distance=2cm}}
}{input{example.option.graph}}
\end{tikzpicture}

\gtr/distance={\langle \text{length} \rangle} \hspace{1em} \text{(no default, initially \text{-1sp})}

A non-negative \( \langle \text{length} \rangle \) replaces the default minimum distance to the preceding sibling. The default settings are given by \gtr/child distance in parent graph \( \rightarrow \text{P. 86} \) etc.

\begin{tikzpicture}
\genealogytree{template=signpost,level 2/.style={
node={box={interior style={top color=red!30,bottom color=red}}} }
}{input{example.option.graph}}
\end{tikzpicture}
Using this option, a node can gain a pivot role in the graph construction. Feasible values are
- **none**: no special treatment.
- **child**: pivot role as a child of a family. The node will be placed centered according to its parents or its pivot parent.
- **parent**: pivot role as a parent of a family. The node will be placed centered according to its children or its pivot child.
- **both**: pivot role as a child and as a parent.

A sequence of /gtr/pivot settings for ancestors or descendants can be used to emphasize a certain lineage. In the following example, the nodes marked in red form such a lineage. The green node is a pivot as a child.

```
\begin{tikzpicture}
genealogytree[template=signpost,  
options for node={Bert2010,John1980,GpSm1949}{pivot,  
box={interior style={top color=red!30,bottom color=red}}},  
options for node={Jane1982}{pivot=child,box={colback=green!50}} ]  
{input{example.option.graph}}
\end{tikzpicture}
```
/gtr/box=\{\langle options\rangle\} (no default)

Passes the given \langle options\rangle to the underlying /gtr/node processor → P.138. Depending on the selected processor, the \langle options\rangle are usually tcolorbox options which describe how a node box is drawn. If a processor is not based on the tcolorbox package, the \langle options\rangle can be different, e.g. TikZ options.

\begin{tikzpicture}
genealogytree[template=formal graph, 
  options for node={node_B}{box={colback=green!50}}, 
]{
  child{
    g{A} p{id=node_B}{B} 
    c{C} c{D} c{E}
  }
}
\end{tikzpicture}

/gtr/box clear (no value)

/gtr/box settings are additive. To clear all box settings, use this option.

/gtr/node box=\{\langle options\rangle\} (no default)

This is an abbreviation for placing /gtr/box inside /gtr/node → P.96.

\begin{tikzpicture}
genealogytree[template=formal graph, 
  level -1/.style={node box={colback=red!50}}, 
]{
  child{
    g{A} p{B} 
    c{C} c{D} c{E}
  }
}
\end{tikzpicture}
/gtr/family box=\langle options \rangle

This is an abbreviation for placing /gtr\box \rightarrow P.98 inside /gtr/family \rightarrow P.105.

\begin{tikzpicture}
genealogytree[template=formal graph]
\%
child{
  g{A} p{B}
  child[family box={colback=red!50}]{
    g{C} p{a_1} c{a_2}
    child{ g{a_3} p{a_4} c{a_5} c{a_6} }
  }
  c{D} c{E}
}
\end{tikzpicture}

\begin{tikzpicture}
genealogytree[template=formal graph]
\%
child{
  g{A} p{B}
  child[subtree box={colback=red!50}]{
    g{C} p{a_1} c{a_2}
    child{ g{a_3} p{a_4} c{a_5} c{a_6} }
  }
  c{D} c{E}
}
\end{tikzpicture}
/gtr/turn\{option\} (default right, initially off)

This is a special \texttt{/gtr/box}\textsuperscript{P.98} style to rotate the content a node. Typically, all nodes of a \texttt{/gtr/level n}\textsuperscript{P.110} may be rotated together.

Feasible \textit{option} values are:

- \texttt{off}: no rotation.
- \texttt{right}: rotation by 90 degrees.
- \texttt{upsidedown}: rotation by 180 degrees.
- \texttt{left}: rotation by 270 degrees.

\begin{tikzpicture}
\genealogytree\{template=formal graph\}{% child{ g[turn]{A} p{B} c[turn=left]{C} c{D} c{E} }
\}
\end{tikzpicture}

\begin{tikzpicture}
\genealogytree\{template=signpost, level 2/.style={level size=3cm,node size from=1cm to 2cm, node={turn,box={no shadow,drop fuzzy shadow southwest}}}}\{input{example.option.graph}}
\end{tikzpicture}
\begin{tikzpicture}
\genealogytree[template=signpost, level 0/.style={level size=2cm,node size from=1cm to 2cm, node=\{turn=left,box=\{no shadow,drop fuzzy shadow northeast\}\}}, input=\{example.option.graph\}]
\end{tikzpicture}

The following three options are \texttt{tcolorbox} options which are declared by the \texttt{genealogytree} package. They can be redefined for customization.

/\texttt{tcb/male} \hfill \text{(style, no value)}
\hspace{1cm} A \texttt{tcolorbox} option defined as
\begin{verbatim}
/\texttt{tcbset}{male/.style={colframe=blue}}
\end{verbatim}

/\texttt{tcb/female} \hfill \text{(style, no value)}
\hspace{1cm} A \texttt{tcolorbox} option defined as
\begin{verbatim}
/\texttt{tcbset}{female/.style={colframe=red}}
\end{verbatim}

/\texttt{tcb/neuter} \hfill \text{(style, no value)}
\hspace{1cm} A \texttt{tcolorbox} option defined as
\begin{verbatim}
/\texttt{tcbset}{neuter/.style={}}
\end{verbatim}

The following three options are \texttt{genealogytree} options which are shortcuts for setting the three options above inside a \texttt{/gtr/box} \textsuperscript{P.98}.

/\texttt{gtr/male} \hfill \text{(style, no value, initially unset)}
\hspace{1cm} This is an abbreviation for placing \texttt{tcb/male} inside \texttt{/gtr/box} \textsuperscript{P.98}.

/\texttt{gtr/female} \hfill \text{(style, no value, initially unset)}
\hspace{1cm} This is an abbreviation for placing \texttt{tcb/female} inside \texttt{/gtr/box} \textsuperscript{P.98}.

/\texttt{gtr/neuter} \hfill \text{(style, no value, initially unset)}
\hspace{1cm} This is an abbreviation for placing \texttt{tcb/neuter} inside \texttt{/gtr/box} \textsuperscript{P.98}.

Also see \texttt{/gtr/database/sex} \textsuperscript{P.165}.

101
\begin{tikzpicture}
genealogytree[template=formal graph,]
\begin{scope}
child{
g[male]{A}
p[female]{B}
c[female]{C} c[male]{D} c[neuter]{E}
}\end{scope}
\end{tikzpicture}

\tcbset{
male/.style={sharp corners,colframe=blue,colback=blue!10,\
watermark text=\gtrSymbolsSetDraw(blue!30)\gtrsymMale},
female/.style={arc=4pt,colframe=red,colback=red!10,\
watermark text=\gtrSymbolsSetDraw(red!30)\gtrsymFemale},
neuter/.style={arc=2pt,colframe=black!80!white,colback=black!5,\
watermark text=\gtrSymbolsSetDraw(black!20)\gtrsymNeuter},
}\begin{tikzpicture}
genealogytree[template=formal graph,]
\begin{scope}
child{
g[male]{A}
p[female]{B}
c[female]{C} c[male]{D} c[neuter]{E}
}\end{scope}
\end{tikzpicture}
Applies extra TikZ \textit{options} for drawing a node. These \textit{options} are \textit{not} used for the node box content, even if \texttt{/gtr/processing} \textsuperscript{\textsuperscript{-P.138}} is set to \texttt{tikznode}. They are used in the drawing process to reserve a node space for the later placement of the box content. The most interesting usage is to move the node from its computed position. Note that the auto-layout algorithm is \textit{not} aware of such movements, but edge drawing will follow the new positioning.

\begin{tikzpicture}
\genealogytree[template=formal graph,]
{%
child{
  g{A} p{B}
  c[tikz={xshift=-20mm,yshift=-5mm}]{C}
  c{D} c{E}
}
}
\end{tikzpicture}

For node movements which influence the auto-layout algorithm, see \texttt{/gtr/distance} \textsuperscript{\textsuperscript{-P.96}}, \texttt{/gtr/pivot} \textsuperscript{\textsuperscript{-P.97}}, and \texttt{/gtr/pivot shift} \textsuperscript{\textsuperscript{-P.106}}. For node content option settings like coloring, see \texttt{/gtr/box} \textsuperscript{\textsuperscript{-P.98}}.
5.6 Family Options

/\texttt{gtr/options for family}\{\{id list\}\{\{options\}\}\}

The given \{\texttt{options}\} are set for all families with \texttt{/gtr/id} \^\textsuperscript{P.92} values from the given \{\texttt{id list}\}. If an \texttt{/gtr/id} \^\textsuperscript{P.92} value is not existing, the setting is silently ignored. The intended spot for using \texttt{/gtr/options for family} is before \texttt{\genealogytree} \^\textsuperscript{P.55} or inside its option list. Also see Section 5.1.1 on page 78 and Section 5.1.2 on page 79.

\begin{tikzpicture}
\genealogytree[template=signpost, options for family={Doe}{box={
  coltext=green!25!black,fontupper=\bfseries,width=3cm, interior style={top color=green!50!white,bottom color=green!75!black}}
}}
\end{tikzpicture}

\texttt{\gtrsetoptionsforfamily\{\{id list\}\{\{options\}\}\}}
\texttt{\getree_set_options_for_family:nn\{\{id list\}\{\{options\}\}\}}

Identical to using \texttt{/gtr/options for family}.
The given \texttt{options} are set for all nodes and edges within the current scope. This scope is intended to be a \texttt{/gtr/level} \textsuperscript{\textit{P.109}} or \texttt{/gtr/level n} \textsuperscript{\textit{P.110}} definition or an option of a family identifier like \texttt{parent} or \texttt{child}. Also see Section 5.1.1 on page 78 and Section 5.1.2 on page 79.

\begin{tikzpicture}
\genealogytree[template=signpost, 
level 2/.style={
    family={
        edges={swing,foreground=red,background=red!20},
        box={interior style={top color=red!30,bottom color=red}}}
    }]
{input{example.option.graph}}
\end{tikzpicture}
For a family, there is a parent pivot point (typically centered between the parents) and a child pivot point (typically centered between the children). Normally, the auto-layout algorithms brings both points in congruence. Using a `/gtr/pivot shift`, there is a shift of the given `<length>` between these two points. Note that this works for child, parent, and sandclock, but not for union.

\begin{tikzpicture}
\genealogytree[template=signpost]
{
parent[pivot shift=-1.5cm]{
  g{Child}
  p[male]{Father}
  p[female]{Mother}
}
}\genealogytree[tree offset=4.5cm]{
  parent{
    g{Child}
    p[male]{Father}
    p[female]{Mother}
  }
}\genealogytree[tree offset=9cm]{
  parent[pivot shift=1.5cm]{
    g{Child}
    p[male]{Father}
    p[female]{Mother}
  }
}\end{tikzpicture}
5.7 Subtree Options

The given \langle options \rangle are set for all subtrees with /gtr/id→P.92 values from the given \langle id list \rangle. Subtrees are identified by the /gtr/id→P.92 of the root family of the subtree. If an /gtr/id→P.92 value is not existing, the setting is silently ignored. The intended spot for using /gtr/options for subtree is before \genealogytree→P.55 or inside its option list. Also see Section 5.1.1 on page 78 and Section 5.1.2 on page 79.
The given \texttt{options} are set for all families and their nodes and edges within the current scope. This scope is intended to be a \texttt{/gtr/level} +P.109 or \texttt{/gtr/level n} +P.110 definition or an option of a family identifier like \texttt{parent} or \texttt{child}. Also see Section 5.1.1 on page 78 and Section 5.1.2 on page 79.
5.8 Level Options

With \textcommand{/gtr/level} and \textcommand{/gtr/level n} options can be set for individual levels of the graph. Inside the key list of these styles, the following options can be used:

- All geometry options, see Section 5.3 on page 83.
- \textcommand{/gtr/node} to set options for nodes.
- \textcommand{/gtr/family} to set options for families.
- \textcommand{/gtr/subtree} to set options for subtrees.
- Also see \textcommand{/gtr/ignore} and \textcommand{/gtr/ignore level}.

Also see Section 5.1.1 on page 78 and Section 5.1.2 on page 79.

\textcommand{/gtr/level=<number>} (style, initially empty)

An initially empty style which is applied at each level with the level \texttt{<number>} as parameter. This style can be redefined.
At each level with the level number \( n \) this style is applied after /gtr/level \(^{\text{P.109}}\). This style can be (re-)defined.

\begin{tikzpicture}
genealogytree[template=signpost,  
level 2/.style={node box={colback=black!30}},  
level 1/.style={node box={colback=red!30}},  
level 0/.style={node box={colback=yellow!30}},  
]
{input{example.option.graph}}
\end{tikzpicture}
5.9 Tree Positioning Options

/gtr/proband level\[\langle number\rangle\] (no default, initially 0)
Sets the level number of the proband to \langle number\rangle. All level numbers inside the given tree will be adapted accordingly. This is useful in connection with /gtr/level \[\text{P.109}\] dependent settings, especially when two trees are connected.

/gtr/tree offset\[\langle length\rangle\] (no default, initially 0pt)
Sets the offset value of the root family to \langle length\rangle. Depending on the given /gtr/timeflow \[\text{P.80}\], this means a shift in horizontal or vertical direction in reference of the tikzpicture coordinate system.

/gtr/after parser\[\langle code\rangle\] (no default, initially empty)
Adds \langle code\rangle to a list of code which is executed after the tree content is parsed and before the parsed data is drawn. This is used internally by other options and may not be needed by a normal user.

The following options allow to shift the whole tree such that a specific node is placed at a specific position.

- /gtr/set position: place a node centered at a position.
- /gtr/adjust position \[\text{P.112}\]: place a node relative to a position (respecting the node dimensions).
- /gtr/adjust node \[\text{P.113}\]: place a node relative to another node (respecting both node dimensions).

/gtr/set position\[\langle node\rangle\] at \langle position\rangle (style, no default)
Adjusts the current graph such that a \langle node\rangle of the graph is placed at the given \langle position\rangle. If the \langle position\rangle is given by coordinates, one has to use curly brackets to enclose \langle position\rangle, e.g. \{2,3\}. The \langle node\rangle is identified by a /gtr/id \[\text{P.92}\].

\begin{tikzpicture}
\node[draw,fill=red!30,minimum size=3cm] (X) at (0,0) {};
\draw[white] (X.south west)--(X.north east) (X.north west)--(X.south east);
\genealogytree[template=signpost,
set position=Harr1987 at X,
options for node={Harr1987}{box={colback=yellow!50}}]
{input{example.option.graph}}
\draw[red!70] (Harr1987) circle (1.5cm);
\end{tikzpicture}
Adjusts the current graph such that a \textit{node} of the graph is placed in the given \textit{direction} relative to the given \textit{position} with a given \textit{distance} in this direction and an optional \textit{shift} orthogonal to the direction. The \textit{node} is identified by a \texttt{/gtr/id} \footnote{P. 92}.

Feasible values for the \textit{direction} are

- right
- left
- above
- below

\begin{tikzpicture}
\draw[red] (-0.3,-0.3)--++(0.6,0.6) (-0.3,0.3)--++(0.6,-0.6);
\node[right=3mm] at (0,0) {Reference Position};
\end{tikzpicture}

\begin{genealogytree}[template=signpost,]
\adjust position=Harr1987 left of {0,0} distance 1cm,
options for node={Harr1987}{box={colback=yellow!50}}]
\input{example.option.graph}
\end{genealogytree}
Adjusts the current graph such that a \textit{(node)} of the graph is placed in the given \textit{(direction)} relative to the given \textit{(reference node)} (a TikZ node) with a given \textit{(distance)} in this direction and an optional \textit{(shift)} orthogonal to the direction. The \textit{(node)} is identified by a /gtr/id \textsuperscript{P. 92}.

Feasible values for the \textit{(direction)} are
\begin{itemize}
  \item \textbf{right} (right of \textit{(reference node)}.\textit{east})
  \item \textbf{left} (left of \textit{(reference node)}.\textit{west})
  \item \textbf{above} (above of \textit{(reference node)}.\textit{north})
  \item \textbf{below} (below of \textit{(reference node)}.\textit{south})
\end{itemize}

\begin{tikzpicture}
\node[fill=yellow!50,draw=red] (R) {Reference Node};
genealogytree[template=signpost,
adjust node=Harr1987 left of R distance 1cm,
options for node={Harr1987}{box={colback=yellow!50}}]
{input{example.option.graph}}
\end{tikzpicture}
5.10 TikZ and Tcolorbox Options

Also see /gtr/tikz$^{\text{P.103}}$.

\gtr/tikzpicture={\langle tikz options \rangle} \hspace{1cm} \text{(no default, initially empty)}

Used to insert \langle tikz options \rangle to the \texttt{tikzpicture} environment inside \texttt{genealogypicture} $^{\text{P.57}}$. This option is ignored by \texttt{genealogytree} $^{\text{P.55}}$!

\begin{genealogypicture}[template=formal graph,\texttt{tikzpicture}={\execute at end picture={\path[draw=red,double,double distance=1pt,very thick,rounded corners] ([xshift=-5mm,yshift=-5mm]current bounding box.south west) rectangle ([xshift=5mm,yshift=5mm]current bounding box.north east);}}] child{ \g{\texttt{box}={colback=red!50}} \{A\} \p\{B\} \c\{C\} \c\{D\} \c\{E\} } \end{genealogypicture}

\begin{genealogypicture}[template=formal graph,\texttt{tikzset}={\texttt{myfill/.style}={\texttt{top color=yellow,bottom color=red}}}] child{ \g{\texttt{box}={\texttt{interior style=myfill}}} \{A\} \p\{B\} \c\{C\} \c\{D\} \c\{E\} } \end{genealogypicture}

\gtr/tikzset={\langle tikz options \rangle} \hspace{1cm} \text{(no default, initially empty)}

Used to insert \langle tikz options \rangle before the tree is drawn by \texttt{genealogytree} $^{\text{P.55}}$ or \texttt{genealogypicture} $^{\text{P.57}}$. In contrast to \texttt{gtr/tikzpicture}, one can use \texttt{gtr/tikzset} also for \texttt{genealogytree} $^{\text{P.55}}$, but some settings may need to be given in the argument of \texttt{tikzpicture} (see \textit{The TikZ and PGF Packages} [5]). Note that \texttt{genealogytree} $^{\text{P.55}}$ does not limit the scope of these settings.

\begin{genealogypicture}[template=formal graph,\texttt{tikzset}={\texttt{myfill/.style}={\texttt{top color=yellow,bottom color=red}}} ] child{ \g{\texttt{box}={\texttt{interior style=myfill}}} \{A\} \p\{B\} \c\{C\} \c\{D\} \c\{E\} } \end{genealogypicture}
/gtr/after tree={⟨tikz code⟩} (no default, initially empty)

Used to insert ⟨tikz code⟩ after the tree is drawn by \genealogytree → P.55 or genealogypicture → P.57. This is also used internally by other options.

\begin{genealogypicture}
[template=formal graph,
after tree={\draw[very thick,blue,-Latex] (node_A) to[out=180,in=120] (node_C);
}]
child{
  \g[box={colback=red!50},id=node_A]{A}
  p\{B\}
  c\{C\} c\{D\} c\{E\}
}
\end{genealogypicture}

/gtr/tcbset={⟨tcolorbox options⟩} (no default, initially empty)

Used to insert ⟨tcolorbox options⟩ before the tree is drawn by \genealogytree → P.55 or genealogypicture → P.57.

Note that \genealogytree → P.55 does not limit the scope of these settings.

\begin{genealogypicture}
[template=formal graph,
tcbset={
  male/.style={colframe=blue,colback=blue!5},
  female/.style={colframe=red,colback=red!5}
}]
child{
  \g[male]\{A\}
  p[female]\{B\}
  c[male]\{C\} c[female]\{D\} c[male]\{E\}
}
\end{genealogypicture}
This is an extension to the \texttt{fit} library of Ti\textit{k}Z. This option must be given to a \texttt{node path} command. The \texttt{⟨id⟩} has to be an \texttt{/gtr/id} value of a family. All nodes of this family are given to the \texttt{fit} option of a Ti\textit{k}Z node which is sized to frame all family members.

Like \texttt{/tikz/fit to subtree=⟨id⟩}, this is an extension to the \texttt{fit} library of Ti\textit{k}Z. All nodes of the subtree identified by \texttt{⟨id⟩} are given to the \texttt{fit} option of a Ti\textit{k}Z node which is sized to frame the whole subtree.

\begin{tikzpicture}
genealogytree[template=tiny boxes]
{
child[id=R,family box={colback=blue}]{
g-p-
    child{
        g-p-c-c-c-c-c-
        child{ p-g-c- }
    }
    c-c-
    child{ g-p-c- }
    child[id=X,subtree box={colback=red}]{
p-g-
        child{ g-p-c-c-c-c-c-c- }
        union{
            p-c-c-
            child{ g-p-c-c- }
        }
    }
}
\node[draw=blue,fill=blue!20,fill opacity=0.25,inner sep=0.5mm, pin={pin edge=blue}left:family with id R}, fit to family=R] {};
\node[draw=red,fill=red!20,fill opacity=0.25,inner sep=0.5mm, pin={pin edge=red}right:subtree with id X}, fit to subtree=X] {};
\end{tikzpicture}
5.11 Ignoring Input

The following options allow to ignore some parts of the input data. Note that debugging using the methods from Chapter 11 on page 249 will usually ignore the ignore settings. Also, if some counters are incremented by node or family options, these increments may not be undone by ignoring the particular node or family.

\begin{tikzpicture}
genealogytree[template=signpost,timeflow=left,level size=3cm]
{
  parent[id=DoeJones]{
    g[id=Deir2012,female]{Deirdre \gtrsymBorn\,2012}
    parent[id=Jones]{
      g[id=Mary1988,female]{Aunt Mary \gtrsymBorn\,1988}
      p[id=JimJ1944,male]{Jim Jones \gtrsymBorn\,1944}
      % the following node is going to be ignored
      p[ignore,id=Jenn1949,female]{Jenny Jones \gtrsymBorn\,1949}
    }
  }
}
\end{tikzpicture}

\begin{tikzpicture}
\genealogytree[template=signpost,timeflow=left,level size=3cm]
{
  parent[id=DoeJones]{
    g[id=Deir2012,female]{Deirdre \gtrsymBorn\,2012}
    parent[id=Jones]{
      g[id=Mary1988,female]{Aunt Mary \gtrsymBorn\,1988}
      p[id=JimJ1944,male]{Jim Jones \gtrsymBorn\,1944}
      % the following node is going to be ignored
      p[ignore,id=Jenn1949,female]{Jenny Jones \gtrsymBorn\,1949}
    }
  }
}
\end{tikzpicture}
/gtr/ignore node={⟨id list⟩} (style, no default)

All nodes with /gtr/id → P.92 values from the given ⟨id list⟩ are ignored. If an /gtr/id → P.92 value is not existing, the setting is silently ignored.

\begin{tikzpicture}
genealogytree[template=signpost,ignore node={Bert2010,Char2014,Harr1987}]
{input{example.option.graph}}
\end{tikzpicture}

/gtr/ignore node{⟨id list⟩}

Identical to using /gtr/ignore node.

/gtr/ignore subtree={⟨id list⟩} (style, no default)

All subtrees with /gtr/id → P.92 values from the given ⟨id list⟩ are ignored. If an /gtr/id → P.92 value is not existing, the setting is silently ignored.

\begin{tikzpicture}
genealogytree[template=signpost,ignore subtree={Doe}]
{input{example.option.graph}}
\end{tikzpicture}

/gtr/ignore subtree{⟨id list⟩}

Identical to using /gtr/ignore subtree.
The level with the given \langle number \rangle is ignored. This also removes unconnected nodes and families. **Note that /gtr/ignore level should never be used, if /gtr/proband level P.111 was set!**

This style sets /gtr/level n \rightarrow P.110 options to remove all unwanted nodes and families. Depending on the algebraic sign of \langle number \rangle the implementation differs. Zero has no effect.

\begin{tikzpicture}
\genealogytree[template=signpost,ignore level=2]
{input{example.option.graph}}
\end{tikzpicture}

\begin{tikzpicture}
\genealogytree[template=signpost,ignore level=2]
{input{example.option.graph}}
\end{tikzpicture}
Ignores all child nodes \texttt{c} in a \texttt{parent} graph. Thereby, all siblings in an ancestor tree are removed.
This is a global option for the whole graph and cannot be applied locally.

\begin{tikzpicture}
\genealogytree[template=signpost,ignore parent childs]
{input{example.option.graph}}
\end{tikzpicture}
5.12 Inserting Input

The following options allow to insert parsable data into the input. This is a powerful feature with the risk to corrupt the structure of the resulting graph. Note that grammar checks are not so strictly applied at the insertion points and occurring errors may be difficult to detect.

\gtrparserdebug \textsuperscript{P.250} ignores inserting options, while \gtrprocessordebug \textsuperscript{P.252} uses these options.

Recursive insertion is possible, i.e. inserting into already inserted data, but should be handled with care. Especially, using /gtr/insert after node and /gtr/insert after family \textsuperscript{P.122} should never be used to insert data after the root element of an inserted node or family.

/gtr/insert after node=\{⟨id⟩\}{⟨input data⟩} \hspace{1cm} \text{(style, no default)}

Inserts ⟨input data⟩ into the graph data right after the node with the given ⟨id⟩ was processed. If no node with ⟨id⟩ exists, this setting is silently ignored. If more than one insert command was given for a specific node, following insert commands for this node are ignored. Note that grammar checks are not so strictly applied at the insertion point, i.e. one has to be more careful to obey the rules to avoid mess.
/gtr/insert after family = \{(id)\}\{(input data)\} (style, no default)

Inserts \{(input data)\} into the graph data right after the family with the given \{(id)\} was processed. If no family with \{(id)\} exists, this setting is silently ignored. There should be not more than one /gtr/insert after family command for a specific family; using it twice may give unpredictable results. Note that grammar checks are not so strictly applied at the insertion point, i.e. one has to be more careful to obey the rules to avoid mess. Especially, never use /gtr/insert after family for the root family!
/gtr/insert at begin family={⟨id⟩}{⟨input data⟩} (style, no default)

Inserts ⟨input data⟩ into the graph data of the family with the given ⟨id⟩, before the content of the family is processed. If no family with ⟨id⟩ exists, this setting is silently ignored. There should be not more than one /gtr/insert at begin family command for a specific family; using it twice may give unpredictable results. Note that grammar checks are not so strictly applied at the insertion point, i.e. one has to be more careful to obey the rules to avoid mess.

\begin{tikzpicture}
\genealogytree[template=formal graph, 
content interpreter content={\gtrifnodeid{\gtrnodeid}{n_{\gtrnodenumber}}}, 
insert at begin family={fam_a}{
\child[subtree box={colback=red!50}]{g-p-c-c-} 
\p{box={colback=yellow!50}}-
 },
\child[id=root]{ g-p- 
\child[subtree box={colback=blue!30},id=fam_a]{ g-c-c-c- } 
\c-c- 
}
}
\end{tikzpicture}
/gtr/insert at end family={⟨id⟩}{⟨input data⟩}

Inserts ⟨input data⟩ into the graph data of the family with the given ⟨id⟩, after the content of the family is processed. If no family with ⟨id⟩ exists, this setting is silently ignored. There should be not more than one /gtr/insert at end family command for a specific family; using it twice may give unpredictable results. Note that grammar checks are not so strictly applied at the insertion point, i.e. one has to be more careful to obey the rules to avoid mess.

\begin{tikzpicture}
\genealogytree[template=formal graph,
content interpreter content={\gtrifnodeid{\gtrinodeid}{n_{\langle\gtrnodenumber\rangle}}},
insert at end family={fam_a}{
  child[subtree box={colback=red!50}]{g-p-c-c-}
  p{box={colback=yellow!50}}-
},
}
{child[id=root]{ g-p-
  child[subtree box={colback=blue!30},id=fam_a]{ g-c-c-c- }
  c-c-
}
}
\end{tikzpicture}
5.13 Phantom Nodes and Subtrees

A /gtr/subtree[^108] style which makes the whole current subtree invisible. This style can also be applied for single nodes. If a \{length\} value is used, the /gtr/node size[^85] for all nodes of the subtree is replaced by \{length\} (width for vertical time flow and height for horizontal time flow).

\begin{tikzpicture}
\genealogytree[template=formal graph, content interpreter content={\gtrifnodeid{\gtrnodeid}{n_{{\gtrnodenumber}}}},]
{
    child{ g-p-
        child{ g-p-
            c[id=A]-
            % invisible phantom
            c[phantom=2cm]-
            c-
        }
        % phantom; borders made visible
        child[phantom,subtree box={show bounding box}]{
            g-p-c-c-c-
        }
        c[id=B]-
    }
}
\end{tikzpicture}
\begin{tikzpicture}
\genealogytree[template=formal graph,
content interpreter content={\gtrifnodeid\gtrnodeid}{n_{\gtrnodenumber}}],
]
{
child{ g-p-
  child{ g-p-
    child{ g-p-c-c-c-}
    c-c-
    child{ g-p-c-
      child{ g-p-c-c-c-}
    }
  }
  child[phantom=3cm]{g[\id=P1]-c[\id=P2]-
    child{ g-p-c-
      child{ g-p-c-c-c-}
      c-c-
    }
  }
}
child{ g-p-c-
  child{ g-p-c-c-c-c-}
  c-c-
}
}
\path[draw,top color=yellow!50,bottom color=blue!50]
  (P2.south west) rectangle node {Phantom Area} (P1.north east);
\end{tikzpicture}
Identical to /gtr/phantom\textsuperscript{P.125}, but the phantom subtree is connected by an edge with its embedding family.

\begin{tikzpicture}
genealogytree [template=formal graph, content interpreter content={\gtrifnodeid{\gtrnodeid}{n_{\{\gtrnodenumber\}}}},]
{
  child{ g-p-
    child{ g-p-
      c[id=A]-
      % invisible phantom
      c[phantom*=2cm]-
      c-
    }
    % phantom; borders made visible
    child[phantom*,subtree box={show bounding box}]{
      g-p-c-c-c-
    }
    c[id=B]-
  }
}
\end{tikzpicture}
5.14 Childless Families

For childless families in a child or sandclock graph, there is in some situations the danger of overlapping edges, if the perpendicular edge drawing is used, also see Section 14.4 on page 354. In the following, a (partial) solution is described which automatically inserts phantom nodes (childs) to childless families using some options. For the impatient: try /gtr/insert phantom for childless families” [P.130] and read the details later.

\begin{tikzpicture}

\begin{genealogytree}[template=tiny boxes, childless/.style={family={box={colback=red!20}, edges={foreground={red,line width=0.2mm}}}}]
{
\node (g-p) {g-p-}
\node (g-p-id=A) [childless] {g-p- union{p-c-}}
\node (g-p-id=B) {g-p-}
\node (g-p-id=C) {g-p-}
\node (g-p-id=D) {g-p-}
\node (g-p-id=E) {g-p-}
\node (g-p-id=F) {p-g-}
\node (g-c-c) {g-c-c-}
\end{genealogytree}
\end{tikzpicture}

In the example above, childless families are marked in red color. Not all cases are critical. For the six childless families from left to right and top to bottom:

- The upper left childless family (A) also contains a union construct with a child. The resulting patchwork family is therefore not considered childless and there is no edge overlapping problem besides the usual one described in Section 14.2.2 on page 350. /gtr/insert for childless families”[P.129] ignores this patchwork family.

- The upper right childless family (B) has overlapping edges with the edges of the sibling. Without coloring, the impression of four parents is given. Inserting a phantom child node is a way to avoid the problem.

- The same is true for the next family (middle left, C) where the nodes falsely appear like parents of the sibling family.

- The middle right childless family (D) is a fake family. Here, child{g-} was used instead of c-. Currently, this makes no difference, but /gtr/insert for childless families”[P.129] will insert a phantom child which changes the layout.

- The lower left childless family (E) is the same kind of fake family. Inserting a phantom node will increase the graph by one (useless) level.
The lower right childless family (F) will also get a phantom node increasing the graph by one level. Since this family is on the lowest layer, edge overlapping is not possible and a phantom child is not needed. Nevertheless, /gtr/insert for childless families will add a child, if not hindered by setting /gtr/insert for childless families level limit → P.131 to a suitable number.

\begin{tikzpicture}
\genealogytree[template=tiny boxes, insert for childless families={c[box={colback=black!90}]{}}], childless/.style={family={box={colback=red!20}, edges={foreground={red,line width=0.2mm}}}},
\end{tikzpicture}

Note that the patchwork family (upper left) is not considered childless, does not need inserted input and does not get one.

Also note that the two single-person families without child do not need inserted input, but get one anyhow. Avoid this by using c- instead of child{ g- }.

For the lowest level, the layout algorithm does not need inserted input at all, but get it. This can be avoided manually by setting /gtr/insert for childless families level limit → P.131 appropriately.

\begin{tikzpicture}
\genealogytree[template=tiny boxes, insert for childless families={⟨input data⟩} (no default, initially empty)]
\end{tikzpicture}
This is a shortcut for using \texttt{/gtr/insert for childless families} → P.129\={c[phantom]}{}

An invisible \texttt{/gtr/phantom} \texttt{\^P.125} node is inserted for every childless family in a \texttt{child} (\texttt{sandclock}) graph to avoid overlapping edges with sibling families. Details see above.

\begin{tikzpicture}
genealogytree[template=tiny boxes, insert phantom for childless families, childless/.style={family={box={colback=red!20}, edges={foreground={red,line width=0.2mm}}}}]

\begin{align*}
\text{child}[g-p-] \\
\text{child}[childless,id=A]{g-p-union{p-c-}} \\
\text{child}[g-p-] \\
\text{c-} \\
\text{child}[childless,id=C]{g-p-} \\
\text{child}[g-c-c-] \\
\text{child}[childless,id=E]{g-} \\
\text{child}[childless,id=F]{p-g-} \\
\text{c-} \\
\text{c-} \\
\text{child}[childless,id=D]{g-} \\
\text{c-} \\
\text{child}[childless,id=B]{g-p-} \\
\end{align*}
\end{tikzpicture}

Note the superfluous vertical space below the graph. Here, the space for the phantom nodes of the lowest level row is reserved. This can be avoided manually by setting \texttt{/gtr/insert for childless families level limit} \texttt{\^P.131} appropriately.
/gtr/insert for childless families level limit = (level) (no default, initially \(-2147483647\))

/gtr/insert for childless families \(\rightarrow\) P.129 and /gtr/insert phantom for childless families \(\rightarrow\) P.130 are only respected, if the current /gtr/level \(\rightarrow\) P.109 is larger than /gtr/insert for childless families level limit.

Typically, /gtr/insert for childless families level limit is set to the lowest level number of the graph to avoid an inserted empty level. See /gtr/level \(\rightarrow\) P.109 for an example to display the level numbers.

\begin{tikzpicture}
genealogytree[template=tiny boxes, insert phantom for childless families, insert for childless families level limit=-3, childless/.style={family={box={colback=red!20}, edges={foreground={red,line width=0.2mm}}}} ]
{
  child{ g-p-
    child[childless,id=A]{ g-p- union{ p-c- } }
    child{ g-p-
      c-
      child[childless,id=C]{ g-p- }
      child{ g-c-c-
        child[childless,id=E]{ g- }
        child[childless,id=F]{ p-g- }
        c- }
      c-}
    child[childless,id=B]{ g-p- }
  }
}
\end{tikzpicture}
5.15 Autofill Parent Graphs (Ancestors)

Missing nodes (ancestors) in a parent graph can be automatically added using the following options. All options in this section complement parents in a family to two parents. Note that the algorithm does not check and will fail, if a family contains three or more parent nodes.

The options are global for the graph and cannot be applied for subtrees. See \texttt{\gtrparent1} \textsuperscript{P.307}, \texttt{\gtrparent2} \textsuperscript{P.307}, etc. for local complementation.

\texttt{/gtr/autofill parents unspecific=} \texttt{\{level\}} \hspace{1cm} \text{(default 5, initially unset)} \hspace{1cm} \texttt{N 2020-06-05}

\texttt{/gtr/autofill parents unspecific=} \texttt{\{level\}} * \hspace{1cm} \text{(default 5, initially unset)} \hspace{1cm} \texttt{N 2020-06-05}

Complements all ancestors of parent families up to the given \texttt{(level)}.

All added nodes are customized by \texttt{/gtr/complemented} \textsuperscript{P.135}.

- For a parent family with zero parents, two parents are added.
- For a parent family with a single parent, one parent is added.
- For a parent family with two parents, nothing is added.

\texttt{/gtr/autofill parents unspecific} also sets \texttt{/gtr/ignore level} \textsuperscript{P.119} to \texttt{(level)}+1.

\begin{tikzpicture}
genealogytree[template=formal graph, autofill parents unspecific=3, complemented/.style={box={colframe=gray!50,colback=gray!10}}, complemented family/.style={edges={% foreground={gray!50,line width=1pt},background={white,line width=1.75pt}}}, ]
{ parent{
  g{A}
  parent{
    g{B}
    p{C}
    parent{
      g{D}
      p{E}
    }
  }
  p{F}
}
}
\end{tikzpicture}
This options needs \texttt{/gtr/processing}\textsuperscript{P.138}=\texttt{database}. See Chapter 7 on page 161 for the database processing concept.

Complements all ancestors of parent families up to the given $\langle\texttt{level}\rangle$.

All added nodes are customized by \texttt{/gtr/complemented}\textsuperscript{P.135}.

- For a parent family with zero parents, a \texttt{/gtr/database/male}\textsuperscript{P.165} and a \texttt{/gtr/database/female}\textsuperscript{P.165} parent node are added.
- For a parent family with a single \texttt{/gtr/database/male}\textsuperscript{P.165} parent, a \texttt{/gtr/database/female}\textsuperscript{P.165} parent node is added.
- For a parent family with a single \texttt{/gtr/database/female}\textsuperscript{P.165} parent, a \texttt{/gtr/database/male}\textsuperscript{P.165} parent node is prefixed.
- For a parent family with two parents which are \texttt{/gtr/database/male}\textsuperscript{P.165} and \texttt{/gtr/database/female}\textsuperscript{P.165}, nothing is changed.
- For a parent family with two parents which are \texttt{/gtr/database/female}\textsuperscript{P.165} and \texttt{/gtr/database/male}\textsuperscript{P.165}, the sorting order is reversed.

\texttt{/gtr/autofill parents male female} also sets \texttt{/gtr/ignore level}\textsuperscript{P.119} to $\langle\texttt{level}\rangle + 1$. 

\begin{tikzpicture}
\genealogytree
\{template=database sideways, database format=name,empty name text={}, level size=3cm,node size from=6mm to 50mm, box={valign=center}, autofill parents male female=3, \}
\parent{ 
  g{male,name=\texttt{Frederik} \texttt{surn}{Smith}} 
  parent{ 
    g{male,name=\texttt{Ernest} \texttt{surn}{Smith}} 
    parent{ 
      g{male,name=\texttt{Dominik} \texttt{surn}{Schmidt}} 
      p{female,name=\texttt{Katharina} \texttt{surn}{Schmidt}} 
    } 
    p{female,name=\texttt{Maria} \texttt{surn}{Huber}} 
  } 
}
\end{tikzpicture}
This option needs `/gtr/processing`\(^{P.138}\)=database. See Chapter 7 on page 161 for the database processing concept.

Complements all ancestors of parent families up to the given (level).

All added nodes are customized by `/gtr/complemented`\(^{P.135}\).

- For a parent family with zero parents, a `/gtr/database/female`\(^{P.165}\) and a `/gtr/database/male`\(^{P.165}\) parent node are added.
- For a parent family with a single `/gtr/database/male`\(^{P.165}\) parent, a `/gtr/database/female`\(^{P.165}\) parent node is prefixed.
- For a parent family with a single `/gtr/database/female`\(^{P.165}\) parent, a `/gtr/database/male`\(^{P.165}\) parent node is added.
- For a parent family with two parents which are `/gtr/database/male`\(^{P.165}\) and `/gtr/database/female`\(^{P.165}\), the sorting order is reversed.
- For a parent family with two parents which are `/gtr/database/female`\(^{P.165}\) and `/gtr/database/male`\(^{P.165}\), nothing is changed.

/gtr/autofill parents female male also sets `/gtr/ignore level`\(^{P.119}\) to (level)+1.

Switches the autofill algorithm off, i.e. this sets the normal case.
/gtr/complemented

An initially empty style which is applied for every complemented parent node. This style can be redefined, see example for /gtr/autofill parents unspecific → P.132.

/gtr/complemented family

An initially empty style which is applied for every complemented parent family. This style can be redefined, e.g. to set a dimmed edge drawing for complemented families. Note that this style does not apply to a family, where just one parent was autofilled.

\gtrset{
  complemented family/.style={%
    edges={rounded=3pt,
      foreground={gray!50, line width=0.2mm},
      background={white, line width=0.6mm}
    },
  }
}

/gtr/complemented phantom

A style which sets /gtr/complemented/.style={phantom}. This sets all complemented nodes to be /gtr/phantom → P.125 nodes (invisible).
5.16 Special and Auxiliary Options

/gtr/reset (no value)
Resets all options to their default values.

/gtr/code=(code) (no default)
The given \texttt{\langle\ code\ \rangle} is executed immediately. This option is useful to place some arbitrary code into an option list.

\begin{tikzpicture}
genealogytree[template=formal graph,
code={\newcommand{\mycom}{(a_{\gtrnodenumber})}},
]{
child{
g{A~\mycom}
p{B}
c{C} c{D~\mycom} c{E}
}
}
\end{tikzpicture}

/gtr/keysfrom=(macro) (no default)
The given \texttt{\langle\ macro\ \rangle} (without parameters) is supposed to contain an option list. The keys from the list are applied.

\begin{tikzpicture}
\genealogytree[template=formal graph, keysfrom=\mylist ]{
\child{
g{A}
p{B}
c{C} c{D} c{E}
}
}
\end{tikzpicture}
Every node in a \genealogytree graph is drawn inside a rectangular box. These boxes are arranged by the auto-layout algorithm to build the entire graph.

The interior of a node box is created by an element called /gtr/node processor. Several customizable node processors are predefined by the package to choose from. Further, an own node processor can be added easily.

The node data may be used as-is or changed in some way before the node processor displays it. This is done by an element called /gtr/content interpreter. Again, several content interpreters are predefined by the package to choose from and own interpreters can be added.

The combination of node interpreter and node processor is called node data processing in the following.

Two classes of node processings can be distinguished:

- Non-interpreting node data processings take their content text as-is and just format it with colors, fonts, frames, etc; see Section 6.2 on page 139.
- Interpreting node data processings use some /gtr/content interpreter to possibly change the content.
  - The most prominent processing is database node processing where the node content is interpreted as organized data. Some representation of the data will form the visual output; see Chapter 7 on page 161.
  - Further interpreters are documented in Section 6.4 on page 155.
6.1 Setting a Node Data Processing and Processor

In this context, there is a small difference between node data processing and a node data processors. The processing is the combination of an node data interpreter and a node data processors. If the interpreter does not change the node data, the difference vanishes.

\[ \text{/gtr/node processor} = \langle \text{macro} \rangle \] (no default)

Sets a \langle \text{macro} \rangle for processing the content of a node. This \langle \text{macro} \rangle has to be defined without parameters. It should display the node content which is stored in \gtrBoxContent → P. 154.

\begin{verbatim}
\newcommand{\myprocessor}{% 
  \tikz\node[outer sep=0pt]{\gtrBoxContent};% 
}\gtrset{node processor=\myprocessor}
\end{verbatim}

This option is useful for authors who wish to implement some very specific node processing (drawing) which is not covered by the standard mechanisms. See \gtr/processing for the standard processors. Since the standard processors are highly customizable, there may be no need to create a specific processor for most use cases.

A predefined \gtr/node processor is set by using \gtr/processing which also sets a \gtr/content interpreter → P. 155.

\[ \text{/gtr/processing} = \langle \text{processing} \rangle \] (no default, initially fit)

Defines the base procedure for processing the content of a node. Feasible values for \langle processing \rangle are

- \textit{fit}: The content is set \textit{as-is} inside a \texttt{tcboxfit} macro from the \texttt{tcolorbox} package, see Section 6.2.1 on page 139.
- \texttt{tcolorbox}: The content is set \textit{as-is} inside a \texttt{tcolorbox} environment from the \texttt{tcolorbox} package, see Section 6.2.2 on page 143.
- \texttt{tcbox}: The content is set \textit{as-is} inside a \texttt{tcbox} macro from the \texttt{tcolorbox} package, see Section 6.2.3 on page 146.
- \texttt{tcbox*}: As a variant to \texttt{tcbox}, the content is also set \textit{as-is} inside a \texttt{tcbox} macro from the \texttt{tcolorbox} package, see Section 6.2.4 on page 149.
- \texttt{tikznode}: The content is set \textit{as-is} inside a \texttt{node} macro from the \texttt{tikz} package, see Section 6.2.5 on page 152.
- \texttt{database}: The content is interpreted as database key-value pairs. The processed content is set inside a \texttt{tcboxfit} macro from the \texttt{tcolorbox} package, see Chapter 7 on page 161.

Values given to \gtr/box → P. 98 will be interpreted according to the defined \langle processing \rangle. For \texttt{tcolorbox}, the values have to be \texttt{tcolorbox} settings; for \texttt{tikznode}, the values have to be \texttt{tikz} settings.
6.2 Predefined Non-Interpreting Processings

6.2.1 fit

The preset processing is based on `\tcboxfit` of the `tcolorbox` package [4]. Options given to `/gtr/box` [P. 98] have to be `tcolorbox` options which are used by `\tcboxfit`.

The `/gtr/no content interpreter` [P. 157] is used. The main characteristics of the applied node data processor are:

- Full observance of `/gtr/level size` [P. 84], `/gtr/node size` [P. 85], and `/gtr/node size from` [P. 86]. These options can be used without restriction.
- The node content is set inside a `minipage`. The text size of the content and the margins are shrunk automatically, if needed. The used font should be freely scalable for this.
- Due to the fit algorithm, this node processing will consume more compilation time than other ones.
- To observe node and level settings as far as possible, the dimensions can be set by `/tcb/gtrNodeDimensions` or `/tcb/gtrNodeDimensionsLandscape`. `/tcb/gtrNodeDimensions` is initially set.

This processor is also used for database processing, see Chapter 7 on page 161.
\begin{genealogypicture}[processing=fit]
  child{ g{root} p{X} } \\
  child{ p{Y} g{A} c{B} c{C} } \\
  c{D} \\
  child{ g{E} p{Z} c{F} c{G} c{H} } \\
\end{genealogypicture}
\begin{genealogypicture}
  \begin{itemize}
    \item This is some longer text.
    \item Y
    \item A
    \item B
    \item C
    \item D
    \item E
    \item F
    \item G
    \item H
  \end{itemize}
\end{genealogypicture}
\begin{genealogypicture}
  level size=1.5cm, level distance=5mm, node size=2cm,
  box={halign=center, valign=center, size=small, arc=2mm, colback=green!20}
%
  child{ g{root}
    c[turn=left]{Abc}
    c[turn=upsidedown]{Bcd}
    c[turn=right]{Cde}
  }
\end{genealogypicture}
6.2.2 \texttt{tcolorbox}

This processing is based on the \texttt{tcolorbox} environment of the \texttt{tcolorbox} package \cite{tcolorbox}. Options given to \texttt{/gtr/box} \cite{gtrbox} have to be \texttt{tcolorbox} options.

The \texttt{/gtr/no content interpreter} \cite{gtnocontent} is used. The main characteristics of the applied node data processor are:

- For \texttt{/gtr/timeflow} \cite{gtrtimeflow} settings \texttt{up} and \texttt{down}, full observance of \texttt{/gtr/node size} \cite{gtrnodesize}, but no observance of \texttt{/gtr/node size from} \cite{gtrnodesizefrom}. The \texttt{/gtr/level size} \cite{gtrlevels} is observed, but content which is too large may overflow.

- For \texttt{/gtr/timeflow} \cite{gtrtimeflow} settings \texttt{left} and \texttt{right}, full observance of \texttt{/gtr/level size} \cite{gtrlevels}. \texttt{/gtr/node size} \cite{gtrnodesize} and \texttt{/gtr/node size from} \cite{gtrnodesizefrom} are both observed, but content which is too large may overflow.

- Using the option \texttt{natural height}, the height of a node box can be freely adapted to its content. This may be especially useful for \texttt{/gtr/timeflow} \cite{gtrtimeflow} settings \texttt{left} and \texttt{right}, but with some limited use for \texttt{/gtr/timeflow} \cite{gtrtimeflow} settings \texttt{up} and \texttt{down}.

- Extremely customizable using options.

- To observe node and level settings as far as possible, the dimensions can be set by \texttt{/tcb/gtrNodeDimensions} or \texttt{/tcb/gtrNodeDimensionsLandscape}. \texttt{/tcb/gtrNodeDimensions} is initially set.
\begin{genealogypicture}
child{ g{root} p{X}
  child{ p{Y} g{A} c{B} c{C} }
c{D}
  child{ g{E} p{Z} c{F} c{G} c{H} }
}
\end{genealogypicture}

\begin{genealogypicture}[processing=tcolorbox,
level size=1.2cm, node size=2.5cm,
box={halign=center, valign=center, size=fbox, arc=2mm, colback=red!20}]
child{ g{root} p{This is some longer text.}
  child{ p{Y} g{A} c{B} c{C} }
c{D}
  child{ g{E} p{Z} c{F} c{G} c{H} }
}
\end{genealogypicture}
\begin{genealogypicture}
\begin{tcolorbox}
\begin{tikzpicture}
\node (root) {This is some longer text.}
child{ g{root} p{This is some longer text.}
child{ p{Y} g{A} c{B} c{C} }
child{ c{D} }
child{ g{E} p{Z} c{F} c{G} c{H} }
}
\end{tikzpicture}
\end{tcolorbox}
\end{genealogypicture}

\begin{genealogypicture}
\begin{tcolorbox}
\begin{tikzpicture}
\node (root) {This is some longer text.}
child{ g{root}
child{ c[turn=left]{Abc}
child{ c[turn=upsidedown]{Bcd}
child{ c[turn=right]{Cde}
}
}
}
\end{tikzpicture}
\end{tcolorbox}
\end{genealogypicture}
6.2.3 tcbox

This processing is based on \texttt{tcbox} of the \texttt{tcolorbox} package [4]. Options given to /gtr/box have to be \texttt{tcolorbox} options which are used by \texttt{tcbox}.

The /gtr/no content interpreter is used. The main characteristics of the applied node data processor are:

- For /gtr/timeflow settings up and down, no observance of /gtr/node size and /gtr/node size from, but full observance of /gtr/level size, if /tcb/gtrNodeDimensions is set.

- For /gtr/timeflow settings left and right, no observance of /gtr/level size. /gtr/node size and /gtr/node size from are both observed, if /tcb/gtrNodeDimensions is set.

- If not specified otherwise by options, the content is set as a single line and the box is sized according to its content.

- To observe node and level settings as far as possible, the dimensions can be set by /tcb/gtrNodeDimensions or /tcb/gtrNodeDimensionsLandscape. /tcb/gtrNodeDimensions is initially not set, but /gtr/turn will switch dimensions settings on.

\begin{genealogypicture}[processing=tcbox]
child{ g{root} p{X}
child{ p{Y} g{A} c{B} c{C} }
c{D}
child{ g{E} p{Z} c{F} c{G} c{H} }
}
\end{genealogypicture}
\begin{genealogypicture}[processing=tcbox,  
level size=1cm,  
box={valign=center,size=title,arc=2mm,colback=red!20} ]  
child{ g{root} p{This is some longer text.}  
child{ p(Y) g(A) c(B) c(C) }  
c(D)  
child{ g(E) p(Z) c(F) c(G) c(H) }  
}  
\end{genealogypicture}
This processing is based on \texttt{tcbox} of the \texttt{tcolorbox} package \cite{tcolorbox}. Options given to \texttt{/gtr/box} \textsuperscript{P.98} have to be \texttt{tcolorbox} options which are used by \texttt{tcbox}. This is a variant of Section 6.2.3 on page 146.

The \texttt{/gtr/no content interpreter} \textsuperscript{P.157} is used. The main characteristics of the applied node data processor are:

- For \texttt{/gtr/timeflow} \textsuperscript{P.80} settings \texttt{up} and \texttt{down}, observance of \texttt{/gtr/node size} \textsuperscript{P.85} (but width may grow beyond) and full observance of \texttt{/gtr/level size} \textsuperscript{P.84}, if \texttt{/tcb/gtrNodeDimensions} is set.

- For \texttt{/gtr/timeflow} \textsuperscript{P.80} settings \texttt{left} and \texttt{right}, some observance of \texttt{/gtr/level size} \textsuperscript{P.84} (but width may grow beyond). \texttt{/gtr/node size} \textsuperscript{P.85} and \texttt{/gtr/node size from} \textsuperscript{P.86} are both observed, if \texttt{/tcb/gtrNodeDimensions} is set.

- If not specified otherwise by options, the content is set horizontally and vertically centered as a single line.

- To observe node and level settings as far as possible, the dimensions can be set by \texttt{/tcb/gtrNodeDimensions} or \texttt{/tcb/gtrNodeDimensionsLandscape}. \texttt{/tcb/gtrNodeDimensions} is initially set.
This is some longer text.
This processing is based on \node of the \tikz package [5]. Options given to /gtr/box\textsuperscript{P.98} have to be \tikz options which are used by \node.

The /gtr/no content interpreter\textsuperscript{P.157} is used. The main characteristics of the applied node data processor are:

- No observance of /gtr/level size\textsuperscript{P.84}, /gtr/node size\textsuperscript{P.85}, and /gtr/node size from\textsuperscript{P.86}.

- Not as customizable as other processors, but full \tikz options.

- This node processing will consume the smallest compilation time.

- To observe node and level settings as far as possible, the dimensions can be set by /tikz/gtrNodeDimensions or /tikz/gtrNodeDimensionsLandscape. /tikz/gtrNodeDimensions is initially not set, but /gtr/turn\textsuperscript{P.100} will switch dimensions settings on.
\begin{genealogypicture}[processing=tikznode, 
level size=5mm,box={fill=yellow!10}, ]
child{ g(root) p(X) 
child{ p(Y) g(A) c(B) c(C) } 
c(D) 
child{ g(E) p(Z) c(F) c(G) c(H) } 
}
\end{genealogypicture}

\begin{genealogypicture}[processing=tikznode, 
timeflow=right, 
level size=1cm,level distance=10mm ]
child{ g(root) p(X) 
child{ p(Y) g(A) c(B) c(C) } 
c(D) 
child{ g(E) p(Z) c(F) c(G) c(H) } 
}
\end{genealogypicture}

\begin{genealogypicture}[processing=tikznode, 
level size=1.5cm,level distance=5mm,node size=2cm, 
box={fill=green!20,gtrNodeDimensions}] 
% 
child{ g(root) 
c[turn=left]{Abc} 
c[turn=upsidedown]{Bcd} 
c[turn=right]{Cde} 
}
\end{genealogypicture}
6.3 Creating a Customized Non-Interpreting Processor

For most applications, one of the predefined non-interpreting processings with their processors will suffice, see Section 6.2 on page 139.

But using `/gtr/node processor`\footnote{P.138}, also a new node processor can be defined.

\gtrset{node processor=myprocessor}

Here, `myprocessor` is an own macro which has to be defined without parameters. Inside the macro definition, the following can be used.

\begin{description}
\item[\gtrBoxContent] Contains the (already interpreted or not interpreted) node content.
\item[\gtrNodeMinWidth] Contains the current target minimum node width as defined by the various tree settings.
\item[\gtrNodeMaxWidth] Contains the current target maximum node width as defined by the various tree settings.
\item[\gtrNodeMinHeight] Contains the current target minimum node height as defined by the various tree settings.
\item[\gtrNodeMaxHeight] Contains the current target maximum node height as defined by the various tree settings.
\item[\gtrNodeBoxOptions] Contains the option settings for the current node. These are the assembled `/gtr/box`\footnote{P.98} settings as comma separated key-value list for the current node.
\end{description}

For demonstration, a simple processor based on the `minipage` environment is constructed:

\begin{verbatim}
\newcommand{\myprocessor}{%
  \begin{minipage}[c]{\gtrNodeMinWidth}\gtrNodeMinHeight\%
    \begin{center}\gtrBoxContent\end{center}\end{minipage}%
  }%
\begin{genealogypicture}{node processor=\myprocessor, %
  level size=8mm,level distance=10mm,node size=2cm}
  \child{ g{root} %
    c{Abc} %
    c{Bcd} %
    c{Cde} %
  }
\end{genealogypicture}
\end{verbatim}

\begin{figure}
\centering
\begin{tikzpicture}
  \node (root) {root};
  \node (Abc) [below left of=root] {Abc};
  \node (Bcd) [below right of=root] {Bcd};
  \node (Cde) [below right of=Bcd] {Cde};
  \draw (root) -- (Abc) -- (Bcd) -- (Cde);
\end{tikzpicture}
\end{figure}
6.4 Content Interpreters

The predefined non-interpreting processings from Section 6.2 on page 139 can easily adapted to become interpreting, if `/gtr/content interpreter` or `/gtr/content interpreter code` is set. The interpreter changes the node content somehow (see Chapter 7 on page 161 for the main example) and gives the changed content to the chosen `/gtr/node processor`.

\[ /gtr/content interpreter=(macro) \quad \text{(no default)} \]

Sets \((macro)\) for interpreting the content of a node. This \((macro)\) has to take one mandatory parameter (the original box content). It has to define a new parameterless macro \(\texttt{gtrBoxContent} \rightarrow \text{P.154} \) which should store the content which is given to the current `/gtr/node processor` for further compilation.

\[
\texttt{\gtrset}\{\text{content interpreter=}\texttt{myinterpreter}\}
\]

The most important interpreter is realized by database processing, see Chapter 7 on page 161. This option may be used to implement an own kind of database processing which differs from the package implementation.

Another use case is to replace the node content completely by some automated content like numbering the nodes.

\[
\texttt{\newcommand}\{\text{myinterpreter}\}[1]\{\texttt{def} \texttt{gtrBoxContent}#1^{(\texttt{gtrnodenumber})}\}\}
\]

\[
\begin{tikzpicture}
\genealogytree[
template=formal graph, 
content interpreter=\texttt{myinterpreter } ]
{
  child{
    g{A} p{B}
    child{ p{C} g{D} c{E} c{F} }
    c{G}
  }
}
\end{tikzpicture}
\]
/gtr/content interpreter code=\{(code)\} \hfill (no default)

Sets \( (code) \) for interpreting the content of a node. This \( (code) \) can use a parameter \#1 (the original box content) and has to define a new parameterless macro \gtrBoxContent \( \rightarrow \) P. 154 which should store the content which is given to the current /gtr/node processor \( \rightarrow \) P. 138 for further compilation.

\begin{tikzpicture}
\genealogytree[
  \scriptsize
  template=formal graph,
  content interpreter code={\def\gtrBoxContent{#1^{(\gtrnodenumber)}}}
]
\child{
  \node (A) {A}
  \node (B) [below] {B}
  \child{
    \node (C) [below] {C}
    \node (D) [below] {D}
    \node (E) [below] {E}
    \node (F) [below] {F}
    \node (G) [below] {G}
  }
}
\end{tikzpicture}

\begin{tikzpicture}
\genealogytree[
  \scriptsize
  template=formal graph,
  content interpreter content={#1^{(\gtrnodenumber)}}
]
\child{
  \node (A) {A}
  \node (B) [below] {B}
  \child{
    \node (C) [below] {C}
    \node (D) [below] {D}
    \node (E) [below] {E}
    \node (F) [below] {F}
    \node (G) [below] {G}
  }
}
\end{tikzpicture}

/gtr/content interpreter content=\{(code)\} \hfill (no default)

Sets \( (code) \) for interpreting the content of a node. This \( (code) \) is the definition for \gtrBoxContent \( \rightarrow \) P. 154. The \( (code) \) can use a parameter \#1 (the original box content).

\begin{tikzpicture}
\genealogytree[
  \scriptsize
  template=formal graph,
  content interpreter content={#1^{(\gtrnodenumber)}}
]
\child{
  \node (A) {A}
  \node (B) [below] {B}
  \child{
    \node (C) [below] {C}
    \node (D) [below] {D}
    \node (E) [below] {E}
    \node (F) [below] {F}
    \node (G) [below] {G}
  }
}
\end{tikzpicture}
\begin{tikzpicture}
\genealogytree[
  template=formal graph,
  content interpreter content={N_{\gtrnodenumber}} ]
{
  child{
    g- p-
    child{ p- g- c- c- c- union{ p- c- } }
    c-
    child{ g- p- c- child{ g- p- c- c- c- } }
  }
}
\end{tikzpicture}

\begin{tikzpicture}
\genealogytree[
  template=formal graph,
  deletion content interpreter 
]
{
  child{
    g{A} p{B}
    child{ p{C} g{D} c{E} c{F} }
    c{G}
  }
}
\end{tikzpicture}

/gtr/no content interpreter (no value, initially set)
Virtually removes any content interpreter. The node content is given directly to the current /gtr/processing \textsuperscript{P.138} for further compilation. Actually, this defines \gtrBoxContent \textsuperscript{P.154} to contain the original box content.

/gtr/deletion content interpreter (no value, initially set)
Deletes any box content. This leads to empty boxes.
This is the content interpreter for database processing, see Chapter 7 on page 161.

% ... make database processing dependend on tcbox*
processing=tcbox*,
database content interpreter,
% ...

The box content is not only used as-is but is also set as /gtr/id \textsuperscript{P.92} of the node. This implies that no macro code is used inside the nodes.

\begin{tikzpicture}
genealogytree
[template=formal graph,
id content interpreter,
options for node={A,G}{box={colback=blue!50,colframe=blue}} ]
{ child{
    g{A} p{B}
    child{ p{C} g{D} c{E} c{F} }
    c{G}
}
}
\draw[-Latex,blue!75!black,thick]
(A) edge[out=180,in=180] (E)
edge[out=90,in=90] (B)
(G) edge[out=270,in=0] (F) ;
\end{tikzpicture}
Sets \( \langle \text{code} \rangle \) for interpreting the content of a node. This \( \langle \text{code} \rangle \) is the definition for \gtrBoxContent. Also, the \( \langle \text{id} \rangle \) for the node is set. The \( \langle \text{code} \rangle \) and \( \langle \text{id} \rangle \) can use a parameter \#1 (the original box content). Note that \( \langle \text{id} \rangle \) will be fully expanded.

\begin{tikzpicture}
\genealogytree
  \template=formal graph,
  \content interpreter id and content={n\gtrnodenumber\{N_{\gtrnodenumber}\}},
  \options for node={n1,n7}{box={colback=blue!50,colframe=blue}}
\end{tikzpicture}

\begin{tikzpicture}
\genealogytree
  \template=formal graph,
  \content interpreter id and content={n\gtrnodenumber\{N_{\gtrnodenumber}\}},
  \options for node={n1,n7}{box={colback=blue!50,colframe=blue}}
\end{tikzpicture}
Database processing is a specialized node data processing, see Chapter 6 on page 137. The node content is interpreted as organized data and some representation of the data will form the visual output.

To switch to database processing, use

```
/gtr/processing ^P.138 = database
```

The box content is interpreted as key-value database list. The actual box construction is based on `\tcboxfit` of the `tcolorbox` package [4]. Options given to `/gtr/box ^P.98` have to be `tcolorbox` options which are used by `\tcboxfit`.

The `/gtr/database content interpreter ^P.158` is used in combination with the node data processor described in Section 6.2.1 on page 139.

For a quick example-based overview, see the full samples in Section 7.5 on page 174 which use the data given in Section 7.2 on page 163.
7.1 Database Concept

The general idea of this database approach is to separate the data content of a node from the formatting. While this is also a common \TeX/\LaTeX idea, the following concept goes somewhat further.

The content producer could be a human person directly, but more presumably a machine like a genealogy program. The node content is written as a comma separated key-value list. This list is processed and its content formatted by a database processor. For a quick survey with an example, see Section 7.2 on page 163.

The content is exported by a program or hand written as key-value list. The format of this list is described in Section 7.3 on page 165. This list is processed by an enclosing \LaTeX document which is created and manipulated by a human. This enclosing document specifies how the content is displayed. This relieves the exporting program from caring about formatting issues and gives full visual control to a human author. The author is relieved from putting down data by hand which presumably is already data-processed with a genealogy program.

Also, the following methods allow to use the same database for different diagrams with possibly different goals and designs.
7.2 Example Settings

This example data is used in the following (also documented in Section 15.2 on page 358).

File «example.database.graph» for the following examples

```json
child[id=SmitBowd1742]{
  g[id=SmitChar1722]{
    male,
    name = {\pref{Charles} \surn{Smith}},
    birth = {{caAD}1722}{London},
    baptism = {1722-04-13}{London},
    death+ = {1764-10-12}{}{killed},
    profession = {Copper smith, soldier},
    comment = {Invented the square wheel},
  }
  p[id=BowdJane1724]{
    female,
    name = {\pref{Jane} \surn{Bowden}},
    floruit- = {1724/1802},
    marriage = {1742-03-02}{London},
  }
  c[id=BowdAbra1740]{
    male,
    name = {\pref{Abraham} \surn{Bowden}},
    birth+ = {1740-01-04}{London}{out of wedlock},
    death = {1740-02-23}{London}
  }
  c[id=SmitEliz1744]{
    female,
    name = {\pref{Elizabeth} \nick{Liz} \surn{Smith}},
    birth = {1744-02-02}{London},
    floruit = {1780/1805}{New York},
    death = {1812-04-12}{Boston},
    comment = {Had a store in Boston},
  }
  c[id=SmitMich1758]{
    male,
    name = {\pref{Michael} \surn{Smith}},
    birth+ = {1758-03-01}{}{died},
  }
}
```

Note especially the /gtr/id values. They are essential as handle to access a singular node from an importing document without changing the database.
Charles Smith
★ ca. 1722
London
♀ April 13, 1722
London
♀ October 12, 1764
London
Copper smith, soldier. Invented the square wheel.

Jane Bowden
♀ March 2, 1742
London
♀ 1724 to 1802

Abraham Bowden
♂ January 4, 1740
London
♀ February 23, 1740
London

Elizabeth “Liz” Smith
♂ February 2, 1744
London
♀ 1780 to 1805
New York
♀ April 12, 1812
Boston
Had a store in Boston.

Michael Smith
♀ March 1, 1758
7.3 Data Keys

/gtr/database/name={(full name)}  (no default, initially empty)

This key holds the (full name) of a person presumably with markup. For customization, the markup should be done with \texttt{\textbackslash pref \textasciitilde P.\,184}, \texttt{\textbackslash surn \textasciitilde P.\,184}, \texttt{\textbackslash nick \textasciitilde P.\,184} instead of common \LaTeX font settings.

\%
\name{} = \{\pref{Elizabeth} \nick{Liz} \surn{Smith}\},
%

- \texttt{\textbackslash pref \textasciitilde P.\,184} marks a preferred given name.
- \texttt{\textbackslash nick \textasciitilde P.\,184} marks a nickname.
- \texttt{\textbackslash surn \textasciitilde P.\,184} marks a surname.

The saved data is accessible by \texttt{\gtrDBname}.

/gtr/database/shortname={(short name)}  (no default, initially empty)

This key holds an optional (short name) of a person presumably with markup. For customization, the markup should be done with \texttt{\textbackslash pref \textasciitilde P.\,184}, \texttt{\textbackslash surn \textasciitilde P.\,184}, \texttt{\textbackslash nick \textasciitilde P.\,184} instead of common \LaTeX font settings.

\%
\shortname{} = \{\nick{Liz} \surn{Smith}\},
%

The saved data is accessible by \texttt{\gtrDBshortname}.

/gtr/database/sex={(sex)}  (no default, initially \texttt{neuter})

This key holds the (sex) of a person. Feasible (but unchecked) values are \texttt{male} and \texttt{female}. \texttt{neuter} is an additional feasible default value, if the sex is unknown, e.g. for a stillborn child without further data. The saved data is accessible by \texttt{\gtrDBsex}. It is recommended to use the shortcuts below.

\%
\sex{} = female,
%

/gtr/database/female  (style, no value)

Shortcut for \texttt{\gtr/database/sex=\texttt{female}}.

/gtr/database/male  (style, no value)

Shortcut for \texttt{\gtr/database/sex=\texttt{male}}.

/gtr/database/neuter  (style, no value)

Shortcut for \texttt{\gtr/database/sex=\texttt{neuter}}.

/gtr/database/comment={(text)}  (no default, initially empty)

This key holds some comment (text) about a person, e.g. occupation or a very concise life description. The saved data is accessible by \texttt{\gtrDBcomment}.

\%
\comment{} = \{Had a store in Boston\},
%
/gtr/database/profession=(text)  (no default, initially empty)
This key holds some (text) about the profession a person. The saved data is accessible by \gtrDBprofession.

\%
profession = {Copper smith, soldier},
\%

/gtr/database/image=(file name)  (no default, initially empty)
This key holds an image (file name) of a person’s portrait. The saved data is accessible by \gtrDBimage.

\%
image = Marry_Smith_1720.jpg,
\%

/gtr/database/imageopt=(⟨options⟩)  (no default, initially empty)
This key holds some ⟨options⟩ to be applied for including an image with /gtr/database/image. These ⟨options⟩ should be valid for \includegraphics. The saved data is accessible by \gtrDBimageopt.

\%
imageopt={viewport=30pt 50pt 150pt 180pt,clip},
\%

/gtr/database/viewport=(a b c d)  (style, no default)
This style is a shortcut for calling /gtr/database/imageopt with an \includegraphics viewport of ⟨a b c d⟩ plus clip option. The saved data is accessible by \gtrDBimageopt.

\%
viewport=30pt 50pt 150pt 180pt,
\%

/gtr/database/uuid=(text)  (no default, initially empty)
This key holds an universally unique identifier (UUID) (text) of a person. In contrast to /gtr/id → P. 92, the UUID should be globally constant. It may be used for interlinking beyond the scope of a genealogy tree diagram. The saved data is accessible by \gtrDBuuid.

\%
uuid = 1021aa0c-2508-488c-9760-f9f84b4df1dd,
\%
This key holds the Kekulé number of a person. The saved data is accessible by `\gtrDBkekule`.

```
... kekule = 1024,
...```

This key holds the (relation) of a person to a proband. This proband is not necessarily the root node of the tree. Feasible (but unchecked) values are
- **ancestor**: direct ancestor of the proband.
- **descendant**: direct descendant of the proband.
- **sibling**: sibling of the proband which is not an ancestor or descendant, i.e. uncle, aunt, nephew, niece of arbitrary degree.
- **unrelated**: nothing of the above.
Adoptions and relations by marriage are not embraced by this key. The saved data is accessible by `\gtrDBrelation`. It is recommended to use the shortcuts below.

```
... relation = ancestor,
...```

Shortcut for `/gtr/database/relation=ancestor`.

```
... relation = descendant,
...```

Shortcut for `/gtr/database/relation=descendant`.

```
... relation = sibling,
...```

Shortcut for `/gtr/database/relation=sibling`.

```
... relation = unrelated,
...```

Shortcut for `/gtr/database/relation=unrelated`.

This key holds a relationship (text) describing the person. `/gtr/database/relationship` may or may not be consistent with `/gtr/database/relation`. The saved data is accessible by `\gtrDBrelationship`.

```
... relationship = Grandfather,
...```

This key holds a (number or text) describing the age (at death) of a person. The saved data is accessible by `\gtrDBage`.

```
... age = 87,
...```

```
... age = {ca.\,70},
...```
The node data may contain more key-value pairs than needed for the current processing. This option controls how the package should react when detecting unknown keys. Feasible values are

- **ignore**: ignore unknown keys,
- **warn**: warn about unknown keys,
- **error**: stop processing at unknown keys,
- **save**: store the value of an unknown key. If a key dummy is detected, its value is stored under /gtr/database/save/dummy.
The following data keys hold events. Every event consists of

- a date, see Section 7.4 on page 172,
- optionally a place
- and sometimes a modifier.

The three main events are

- Birth,
- Marriage,
- Death.

The other events may or may not be considered for data formatting.

The saved data for the events is accessible by \gtPrintEvent→P.191, \gtPrintDate→P.186, and \gtPrintPlace→P.190. The existence of data can be checked by \gtifdatedefined→P.186 and \gtifplacedefined→P.190.

\gtr/database/birth={⟨date⟩}{⟨place⟩} (no default)
This key holds a birth event with given ⟨date⟩ and ⟨place⟩.

```plaintext
%...
birth = {1744-02-02}{London},
%...
```

\gtr/database/birth+={⟨date⟩}{⟨place⟩}{⟨modifier⟩} (no default)
This key holds a birth event with given ⟨date⟩, ⟨place⟩, and a ⟨modifier⟩ to describe the event further. Feasible values for the ⟨modifier⟩ are

- empty (normal),
- out of wedlock,
- stillborn,
- died.

```plaintext
%...
birth+ = {1740-01-04}{London}{out of wedlock},
%...
```

\gtr/database/birth-=(date) (no default)
This key holds a birth event with given ⟨date⟩.

```plaintext
%...
birth- = {1744-02-02},
%...
```

\gtr/database/baptism={⟨date⟩}{⟨place⟩} (no default)
This key holds a baptism event with given ⟨date⟩ and ⟨place⟩.

\gtr/database/baptism+={⟨date⟩}{⟨place⟩}{⟨modifier⟩} (no default)
Identical to /gtr/database/baptism since there is no valid ⟨modifier⟩.

\gtr/database/baptism-=(date) (no default)
This key holds a baptism event with given ⟨date⟩.
<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/gtr/database/engagement={⟨date⟩}{⟨place⟩}</td>
<td>This key holds an engagement event with given ⟨date⟩ and ⟨place⟩. (no default)</td>
</tr>
<tr>
<td>/gtr/database/engagement+=⟨date⟩{⟨place⟩}{⟨modifier⟩}</td>
<td>Identical to /gtr/database/engagement since there is no valid ⟨modifier⟩. (no default)</td>
</tr>
<tr>
<td>/gtr/database/engagement-=⟨date⟩</td>
<td>This key holds an engagement event with given ⟨date⟩. (no default)</td>
</tr>
<tr>
<td>/gtr/database/marriage={⟨date⟩}{⟨place⟩}</td>
<td>This key holds a marriage event with given ⟨date⟩ and ⟨place⟩. (no default)</td>
</tr>
</tbody>
</table>
| /gtr/database/marriage+=⟨date⟩{⟨place⟩}{⟨modifier⟩} | This key holds a marriage event with given ⟨date⟩, ⟨place⟩, and a ⟨modifier⟩ to describe the event further. Feasible values for the ⟨modifier⟩ are  
  - empty (normal),  
  - other. (no default) |
| /gtr/database/marriage-=⟨date⟩ | This key holds a marriage event with given ⟨date⟩. (no default) |
| /gtr/database/divorce={⟨date⟩}{⟨place⟩} | This key holds a divorce event with given ⟨date⟩ and ⟨place⟩. (no default) |
| /gtr/database/divorce+=⟨date⟩{⟨place⟩}{⟨modifier⟩} | Identical to /gtr/database/divorce since there is no valid ⟨modifier⟩. (no default) |
| /gtr/database/divorce-=⟨date⟩ | This key holds a divorce event with given ⟨date⟩. (no default) |
| /gtr/database/floruit={⟨date⟩}{⟨place⟩} | This key holds a floruit event with given ⟨date⟩ and ⟨place⟩. (no default) |
| /gtr/database/floruit+=⟨date⟩{⟨place⟩}{⟨modifier⟩} | Identical to /gtr/database/floruit since there is no valid ⟨modifier⟩. (no default) |
| /gtr/database/floruit-=⟨date⟩ | This key holds a floruit event with given ⟨date⟩. (no default) |
| /gtr/database/death={⟨date⟩}{⟨place⟩} | This key holds a death event with given ⟨date⟩ and ⟨place⟩. (no default) |
| /gtr/database/death+=⟨date⟩{⟨place⟩}{⟨modifier⟩} | This key holds a death event with given ⟨date⟩, ⟨place⟩, and a ⟨modifier⟩ to describe the event further. Feasible values for the ⟨modifier⟩ are  
  - empty (normal),  
  - killed. (no default) |
| /gtr/database/death-=⟨date⟩ | This key holds a death event with given ⟨date⟩. (no default) |
This key holds a burial event with given \textit{date} and \textit{place}.

This key holds a burial event with given \textit{date}, \textit{place}, and a \textit{modifier} to describe the event further. Feasible values for the \textit{modifier} are

- \textit{empty} (normal),
- \textit{cremated}.

This key holds a burial event with given \textit{date}.

Every event contains a date. If a date is unknown for an event like e.g. a birth, a date range can be used, see next section. Otherwise, e.g. '?' could be used for an unknown date, or more flexibly, something like an own macro \texttt{\nodate} which could be set differently for a specific application, e.g. \texttt{\newcommand{\nodate}{\unskip}}...
### 7.4 Input Format for Dates

A *date* can be given as a *single date* or as a *date range*. A *single date* is specified in the format

\[(c)yyyy-mm-dd\]

with calendar c, year yyyy, month mm, and day dd. The calendar c flag is optional and can be

- **AD**: Anno Domini; this is the default setting, if the calendar flag is omitted. Use this (or nothing) for every ‘normal’ date.
- **BC**: Before Christ; obviously used for dates before Christ.
- **GR**: Gregorian calendar; use this in situations, where the difference between Gregorian and Julian calendar should be emphasized.
- **JU**: Julian calendar; use this in situations, where the difference between Gregorian and Julian calendar should be emphasized.
- **caAD**: circa, but AD; use this for insecure date settings.
- **caBC**: circa, but BC; use this for insecure date settings.
- **ca**: circa; do not use this directly. The language settings for this will be used automatically, if caAD is given and /gtr/calendar print → all but AD is set.
- **other**: other flags may be used without error. The flag is just noted.

The date format can be shortened to \[(c)yyyy-mm\] and \[(c)yyyy\]. Since the calendar flag is optional, yyyy-mm-dd, yyyy-mm, and yyyy are also possible.

A *date range* is specified in the format

\[(c)yyyy-mm-dd/(c)yyyy-mm-dd\]

Every partial date may be shortened as described above.

Also, /\[(c)yyyy-mm-dd\] and /\[(c)yyyy-mm-dd/\] are valid to denote open ranges.

### Date Examples

<table>
<thead>
<tr>
<th>Specification</th>
<th>d.M.yyyy</th>
<th>month d yyyy</th>
<th>dd/month/yyyy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1875-12-07</td>
<td>7.XII.1875</td>
<td>December 7, 1875</td>
<td>07/Dec/1875</td>
</tr>
<tr>
<td>(GR) 1599-08</td>
<td>VIII.1599greg.</td>
<td>August, 1599greg.</td>
<td>Aug/1599greg.</td>
</tr>
<tr>
<td>1475</td>
<td>1475</td>
<td>1475</td>
<td>1475</td>
</tr>
<tr>
<td>/1690-03</td>
<td>before III.1690</td>
<td>before March, 1690</td>
<td>before Mar/1690</td>
</tr>
<tr>
<td>1775-07-15/</td>
<td>after 15.VII.1775</td>
<td>after July 15, 1775</td>
<td>after 15/Jul/1775</td>
</tr>
<tr>
<td>1888-05/1889-06-07</td>
<td>V.1888 to 7.VI.1889</td>
<td>May, 1888 to June 7, 1889</td>
<td>May/1888 to 07/Jun/1889</td>
</tr>
<tr>
<td>(caAD) 1955-02</td>
<td>ca. II.1955</td>
<td>ca. February, 1955</td>
<td>ca. Feb/1955</td>
</tr>
</tbody>
</table>
\texttt{\textbackslash gtrParseDate\{name\}\{\langle date\}\}}

Dates are parsed as part of events automatically, see Section 7.3 on page 165. But with \texttt{\textbackslash gtrParseDate}, a \texttt{\langle date\\}} can be parsed directly. The parsed data is stored using the given \texttt{\langle name\\}} as \texttt{\textbackslash gtrDB\{name\}cal}, \texttt{\textbackslash gtrDB\{name\}day}, \texttt{\textbackslash gtrDB\{name\}month}, \texttt{\textbackslash gtrDB\{name\}year}, \texttt{\textbackslash gtrDB\{name\}endcal}, \texttt{\textbackslash gtrDB\{name\}endday}, \texttt{\textbackslash gtrDB\{name\}endmonth}, \texttt{\textbackslash gtrDB\{name\}endyear}.

\begin{quote}
\texttt{\textbackslash gtrParseDate\{xy\}\{1875-12-07\}}

The parsed date is
\texttt{\textbackslash gtrDB\{xycal\} (AD), \textbackslash gtrDB\{xyday\} (07), \textbackslash gtrDB\{xymonth\} (12), \textbackslash gtrDB\{xyyear\} (1875).}

Formatted date: \texttt{\textbackslash gtrset\{date format=d/M/yyyy\}\textbackslash gtrPrintDate\{xy\}}
\end{quote}

\texttt{\textbackslash gtrParseDate\{xy\}}

The parsed date is \texttt{\textbackslash gtrDB\{xycal\} (AD), \textbackslash gtrDB\{xyday\} (07), \textbackslash gtrDB\{xymonth\} (12), \textbackslash gtrDB\{xyyear\} (1875).}

Formatted date: 7/XII/1875
7.5 Formatting the Node Data

While the macros and options of the next sections describe how to format a single piece of data, the `/gtr/database format` integrates a collection of these pieces to format the total content of a node.

```
/gtr/database format=⟨format⟩ (style, no default, initially medium)
```

Selects a predefined ⟨format⟩ for selecting and arranging data values. The standard ⟨format⟩ designs use `gtrprintlist`\(^{P.193}\) to list events. New ⟨format⟩ designs can be added by `\gtrDeclareDatabaseFormat`\(^{P.182}\). The following sections describe how to customize certain parts of the standard ⟨format⟩ designs, e.g. `/gtr/date format`\(^{P.186}\) for changing the date style.

Feasible (standard) ⟨format⟩ values are

- **full**: name, birth, baptism, engagement, marriage, divorce, floruit, death, burial, and informations as profession and comment.
- **full marriage above**: identical to **full**, but engagement, marriage, divorce is put above and separated by a `\tcbline`.
- **full marriage below**: identical to **full**, but engagement, marriage, divorce is put below and separated by a `\tcbline`.
- **full no marriage**: identical to **full**, but without engagement, marriage, and divorce.
- **medium**: name, birth (or baptism), marriage (or engagement or divorce), death (or burial), and informations as profession and comment. Floruit is displayed, if there is no birth, baptism, death, and burial.
- **medium marriage above**: identical to **medium**, but marriage (or engagement or divorce) is put above and separated by a `\tcbline`.
- **medium marriage below**: identical to **medium**, but marriage (or engagement or divorce) is put below and separated by a `\tcbline`.
- **medium no marriage**: identical to **medium**, but without engagement, marriage, and divorce.
- **short**: name, birth (or baptism), marriage (or engagement or divorce), and death (or burial). Floruit is displayed, if there is no birth, baptism, death, and burial.
- **short marriage above**: identical to **short**, but marriage (or engagement or divorce) is put above and separated by a `\tcbline`.
- **short marriage below**: identical to **short**, but marriage (or engagement or divorce) is put below and separated by a `\tcbline`.
- **short no marriage**: identical to **short**, but without engagement, marriage, and divorce.
- **name**: name only.
- **symbol**: symbol only.
- **empty**: nothing.
- **marriage**: only marriage (or engagement or divorce). This format is intended to be used not for nodes, but for edge labels, see `/gtr/label database options`\(^{P.219}\).
Charles Smith
★ ca. 1722
London
⇒ April 13, 1722
London
× October 12, 1764
Copper smith,
soldier. Invented
the square wheel.

March 2, 1742
London
1724 to 1802

Abraham Bowden
(★) January 4,
1740
London
† February 23,
1740
London

Elizabeth "Liz" Smith
★ February 2, 1744
London
♀ 1780 to 1805
New York
† April 12, 1812
Boston
Had a store in
Boston.

Michael Smith
†† March 1, 1758

March 1, 1758

Charles Smith
★ ca. 1722
London
⇒ April 13, 1722
London
× October 12, 1764
Copper smith,
soldier. Invented
the square wheel.

March 2, 1742
London
1724 to 1802

Abraham Bowden
(★) January 4,
1740
London
† February 23,
1740
London

Elizabeth "Liz" Smith
★ February 2, 1744
London
♀ 1780 to 1805
New York
† April 12, 1812
Boston
Had a store in
Boston.

Michael Smith
†† March 1, 1758

Michael Smith
March 1, 1758

Charles Smith
★ ca. 1722
London
♫ April 13, 1722
London
♫ October 12, 1764
Copper smith, soldier. Invented the square wheel.

Jane Bowden
★ 1724 to 1802

Abraham Bowden
★ January 4, 1740
London
† February 23, 1740
London

Elizabeth "Liz" Smith
★ February 2, 1744
London
♫ 1780 to 1805
New York
♫ April 12, 1812
Boston
Had a store in Boston.

Michael Smith
★† March 1, 1758

Had a store in Boston.
Charles Smith
★ ca. 1722
London
★ October 12, 1764
Copper smith, soldier. Invented the square wheel.

Jane Bowden
★★ March 2, 1742
London
★ 1724 to 1802

Abraham Bowden
(★) January 4, 1740
London
† February 23, 1740

Elizabeth “Liz” Smith
★ February 2, 1744
London
† April 12, 1812
Boston
Had a store in Boston.

Michael Smith
★★ March 1, 1758

177
Charles Smith
★ ca. 1722
London
X October 12, 1764
Copper smith, soldier. Invented the square wheel.

Jane Bowden
★ 1724 to 1802

Abraham Bowden
(★) January 4, 1740
London
† February 23, 1740
London

Elizabeth “Liz” Smith
★ February 2, 1744
London
† April 12, 1812
Boston
Had a store in Boston.

Michael Smith
★★ March 1, 1758

Charles Smith
★ ca. 1722
London
X October 12, 1764
Copper smith, soldier. Invented the square wheel.

Jane Bowden
★ 1724 to 1802

Abraham Bowden
(★) January 4, 1740
London
† February 23, 1740
London

Elizabeth “Liz” Smith
★ February 2, 1744
London
† April 12, 1812
Boston
Had a store in Boston.

Michael Smith
★★ March 1, 1758
Charles Smith
 ★ ca. 1722
 London
 X October 12, 1764

 Jane Bowden
 O March 2, 1742
 London
 X 1724 to 1802

 Abraham Bowden
 (★) January 4, 1740
 London
 † February 23, 1740
 London

 Elizabeth “Liz” Smith
 ★ February 2, 1744
 London
 † April 12, 1812
 Boston

 Michael Smith
 ★† March 1, 1758
Charles Smith
★ ca. 1722
London
★ October 12, 1764

Jane Bowden
★ 1724 to 1802
London
★ March 2, 1742

Abraham Bowden
(★) January 4, 1740
London
† February 23, 1740

Elizabeth “Liz” Smith
★ February 2, 1744
London
† April 12, 1812
Boston

Michael Smith
★★ March 1, 1758

Charles Smith
★ ca. 1722
London
★ October 12, 1764

Jane Bowden
★ 1724 to 1802
London
★ March 2, 1742

Abraham Bowden
(★) January 4, 1740
London
† February 23, 1740

Elizabeth “Liz” Smith
★ February 2, 1744
London
† April 12, 1812
Boston

Michael Smith
★★ March 1, 1758
\begin{genealogypicture}
\begin{itemize}
\item[] database format=name
\item[] processing=database, database format=name, node size=3cm, level size=1cm, box={fit basedim=9pt, boxsep=2pt, halign=flush center, valign=center, gtrDBsex }
\item[] input{example.database.graph}
\end{itemize}
\end{genealogypicture}

\begin{genealogypicture}
\begin{itemize}
\item[] database format=symbol
\item[] processing=database, database format=symbol, node size=1cm, level size=1cm, box={fit basedim=16pt, boxsep=2pt, halign=flush center, valign=center, gtrDBsex }
\item[] input{example.database.graph}
\end{itemize}
\end{genealogypicture}

\begin{genealogypicture}
\begin{itemize}
\item[] database format=empty
\item[] processing=database, database format=empty, node size=1cm, level size=1cm, box={fit basedim=16pt, boxsep=2pt, halign=flush center, valign=center, gtrDBsex }
\item[] input{example.database.graph}
\end{itemize}
\end{genealogypicture}
\gtrDeclareDatabaseFormat{(format)}{(option code)}{(content code)}

Declares a new <format> to be used as value for /gtr/database format → P. 174. The <option code> is used after the data is read and before the box is set. The <content code> is used to fill the box content. It is recommended to start a new <format> name with the letter ‘x’ to avoid collisions with future standard values.

\gtrDeclareDatabaseFormat{xkekule}\%
  \ifdefvoid{\gtrDBkekule}\{}
    \gtrset{box={title=\gtrDBkekule}}\%
  \}\%
\gtrPrintName\%
\begin{gtreventlist}\%
  \gtrifdatedefined{birth}\{
    \gtrlistseparator\gtrPrintEvent{birth}\%
  \}\%
  \gtrifdatedefined{death}\{
    \gtrlistseparator\gtrPrintEvent{death}\%
  \}\%
  \gtrifdatedefined{burial}\{
    \gtrlistseparator\gtrPrintEvent{burial}\%
  \}\%
\end{gtreventlist}\%
\begin{gtrinfolist}\%
  \gtrifprofessiondefined\{
    \gtrlistseparator\gtrPrintProfession\%
  \}\%
  \gtrifcommentdefined\{
    \gtrlistseparator\gtrPrintComment\%
  \}\%
\end{gtrinfolist}\%
\gtrifdatedefined{marriage}\{
  \begin{gtreventlist}\%
    \gtrlistseparator\gtrPrintEvent{marriage}\%
  \end{gtreventlist}\%
\}\%
\begin{genealogypicture}\%
  processing=database,database format=xkekule,\%
  node size=3cm, level size=3.2cm,\%
  list separators hang, place text={\newline}\{},\%
  box={fit basedim=9pt, boxsep=2pt, segmentation style=solid,\%
    center title, fonttitle={\bfseries \normalsize},\%
    halign=flush left, before upper={\parskip=1pt, \gtrDBsex}\ }\%
child{
  g[id=SmitChar1722]\{
    male,\%
    kekule = 2,\%
    name = {\pref{Charles} \surn{Smith}},\%
    birth = {1722}{London},\%
    baptism = {1722-04-13}{London},\%
    death+ = {1764-10-12}{killed},\%
    profession = {Copper smith, soldier},\%
  }\%
  p[id=BowdJane1724]\{
    female,\%
    kekule = 3,\%
    name = {\pref{Jane} \surn{Bowden}},\%
    birth = {1724},\%
    marriage = {1742-03-02}{London},\%
    death = {1802-07-07}{New York},\%
  }\%
  c[id=BowdAbra1740]\{
    male,\%
    name = {\pref{Abraham} \surn{Bowden}},\%
    birth+ = {1740-01-04}{London}{out of wedlock},\%
    death = {1740-02-23}{London},\%
  }\%
  c[id=SmitEliz1744]\{
    female,\%
    kekule = 1,\%
    name = {\pref{Elizabeth} \nick{Liz} \surn{Smith}},\%
    birth = {1744-02-02}{London},\%
  }\%
}
Internally used to print the formatted data inside a node or an edge label. This command will not be needed by most users since it is applied automatically. The format of the output is defined by \texttt{\textbackslash gtrDeclareDatabaseFormat\texttt{P.182}}.

\texttt{\textbackslash gtrset\{database/.cd, name=\texttt{\{pref\{Elizabeth\} \textbackslash nick\{Liz\} \textbackslash surname\{Smith\}\}, birth = \texttt{\{1744-02-02\}\{London\}, death = \texttt{\{1812-04-12\}\{Boston\}, comment = \texttt{\{Had a store in Boston\}},
}}
\%
\texttt{\textbackslash gtrPrintDatabase}

\textit{Elizabeth “Liz” Smith}
\texttt{\{star\} February 2, 1744 in London
\texttt{\{arrow\} April 12, 1812 in Boston
\texttt{Had a store in Boston.}}
7.6 Formatting Names

\gtrPrintName
Used to insert the formatted name. The output format of the name is controlled by \gtr/name and other following options.

\gtrset{database/.cd, name={{\pref{Elizabeth} \nick{Liz} \surn{Smith}}} %...
\gtrPrintName
Elizabeth “Liz” Smith

\pref{{\langle\textit{given name}\rangle}}
Marks a preferred \langle\textit{given name}\rangle. May be redefined directly or using \gtr/pref code.

\surn{{\langle\textit{surname}\rangle}}
Marks a \langle\textit{surname}\rangle. May be redefined directly or using \gtr/surn code.

\nick{{\langle\textit{nickname}\rangle}}
Marks a \langle\textit{nickname}\rangle. May be redefined directly or using \gtr/nick code.

\gtr/pre macro code\{\langle\textit{code}\rangle\} (no default)
Redefines \pref using \langle\textit{code}\rangle.

\gtrset{database/.cd, name={{\pref{Elizabeth} \nick{Liz} \surn{Smith}}} %...
\gtrset{pref code={\textcolor{blue}{\bfseries #1}}} %...
\gtrPrintName
Elizabeth “Liz” Smith

\gtr/surn macro code\{\langle\textit{code}\rangle\} (no default)
Redefines \surn using \langle\textit{code}\rangle.

\gtrset{database/.cd, name={{\pref{Elizabeth} \nick{Liz} \surn{Smith}}} %...
\gtrset{surn code={\textcolor{blue}{\bfseries #1}}} %...
\gtrPrintName
Elizabeth “Liz” Smith

\gtr/nick macro code\{\langle\textit{code}\rangle\} (no default)
Redefines \nick using \langle\textit{code}\rangle.

\gtrset{database/.cd, name={{\pref{Elizabeth} \nick{Liz} \surn{Smith}}} %...
\gtrset{nick code={\textcolor{blue}{\bfseries #1}}} %...
\gtrPrintName
Elizabeth Liz Smith

\gtr/name=full|short (no default, initially full)
Controls, if \gtrPrintName should preferably use the full version (/gtr/database/name \textsuperscript{P.165}) or the short version (/gtr/database/shortname \textsuperscript{P.165}) of a name. If the preferred version is not available, the other version is used.
\gtrset{database/.cd,name={\pref{Elizabeth} \nick{Liz} \surn{Smith}}} \\
\gtrset{name font={\fontfamily{ptm}\selectfont\color{green!50!black}}} \\
\gtrset{database/.cd,name={\pref{Elizabeth} \nick{Liz} \surn{Smith}},female} \\
\gtrset{name code={\gtrPrintSex~\gtrDBname}}

Elizabeth “Liz” Smith

\gtrPrintName

Elizabeth “Liz” Smith

\gtrPrintName

??

N.N.

\gtrPrintName

♀ Elizabeth “Liz” Smith
### 7.7 Formatting Dates

#### \texttt{\gtrPrintDate}\langle\textit{name}\rangle\}

Used to insert a formatted date referred by \langle\textit{name}\rangle. This \langle\textit{name}\rangle is an event name like \textit{birth}, see Section 7.3 on page 165, or any other name used by \texttt{\gtrParseDate}. The output format of the date is controlled by \texttt{/gtr/date format} and other following options.

```
\gtrset{database/.cd,birth={1354-02-09}{Rome}}
%...
The birth was \gtrPrintDate{birth}.
The death was \gtrPrintDate{death}.
```

The birth was February 9, 1354. The death was ??.

#### \texttt{\gtrifdatedefined}\langle\textit{name}\rangle\{\langle\textit{true}\rangle\}{\langle\textit{false}\rangle}\}

Expands to \langle\textit{true}\rangle, if a date with the given \langle\textit{name}\rangle is defined, and to \langle\textit{false}\rangle otherwise.

```
\gtrset{database/.cd,birth={1354-02-09}{Rome}}
%...
\gtrifdatedefined{birth}{The birth was \gtrPrintDate{birth}.
\gtrifdatedefined{death}{The death was \gtrPrintDate{death}.}{}}
```

The birth was February 9, 1354.

#### \texttt{/gtr/date format}=\langle\textit{format}\rangle\}

(no default, initially \texttt{typical})

This option controls how day, month, and year of a date are formatted when using \texttt{\gtrPrintDate}. This setting is \textit{not} dependent, but month names are. One exception to this rule is

- \texttt{typical} \hspace{1cm} \textit{A typical format for the language, here: February 9, 1354}

Further feasible \langle\textit{format}\rangle values are

- \texttt{dd.mm.yyyy} \hspace{1cm} 09.02.1354
- \texttt{d.m.yyyy} \hspace{1cm} 9.2.1354
- \texttt{d.M.yyyy} \hspace{1cm} 9.II.1354
- \texttt{d.month yyyy} \hspace{1cm} 9. February 1354
- \texttt{d.month yyyy} \hspace{1cm} 9. February 1354
- \texttt{d mon.yyyy} \hspace{1cm} 9. Feb. 1354
- \texttt{dd/mm/yyyy} \hspace{1cm} 09/02/1354
- \texttt{d/month/yyyy} \hspace{1cm} 9/February/1354
- \texttt{dd-mm-yyyy} \hspace{1cm} 09-02-1354
- \texttt{d/mm/yyyy} \hspace{1cm} 09/02/1354
- \texttt{d mm yyyy} \hspace{1cm} 9 Feb 1354
- \texttt{d M yyyy} \hspace{1cm} 9 II 1354
- \texttt{dd mon yyyy} \hspace{1cm} 09 Feb 1354
- \texttt{d mon yyyy} \hspace{1cm} 9 Feb 1354
- \texttt{dd-mm-yyyy} \hspace{1cm} 09-02-1354
- \texttt{d mon yyyy} \hspace{1cm} 9 Feb 1354
- \texttt{dd-mm-yyyy} \hspace{1cm} 09-02-1354
\gtrset{database/.cd,birth={1354-02-09}{Rome}}
\gtrset{date format=month d yyyy}

The birth was \gtrPrintDate{birth}.

The birth was February 9, 1354.
/gtr/date code=(\langle code\rangle)  \hspace{1em} \text{(no default)}

Defines \langle code\rangle to be executed by \gtrPrintDate\textsuperscript{\textit{P.186}}. Use this, if /gtr/date format\textsuperscript{\textit{P.186}} is not flexible enough.

\begin{verbatim}
\gtrset{database/.cd,birth={1354-02-09}{Rome}}
\gtrset{date code={%\ifcsdef{#1month}{%\ifcsdef{#1day}{\csuse{#1day}}{}%\csuse{#1month}}%\csuse{#1year}%%}}
  \%...
  \csuse{#1year}\
  \gtrPrintDate{birth}.
\end{verbatim}

The birth was 09(02)1354.

/gtr/calendar text for=\langle calendar\rangle is \{\langle prefix\rangle}\{\langle postfix\rangle\}  \hspace{1em} \text{(no default)}

Defines a \langle prefix\rangle and a \langle postfix\rangle text for a \langle calendar\rangle. This setting is /gtr/language\textsuperscript{\textit{P.247}} dependent for known calendars. This option also allows to set up new \langle calendar\rangle entries.

\begin{verbatim}
\gtrset{database/.cd,birth={(AUC)2107-02-09}{Rome}}
\gtrset{calendar text for=AUC is {}{ a.u.c.}}
\gtrPrintDate{birth}.
\end{verbatim}

The birth was February 9, 2107 a.u.c..

/gtr/calendar print=\langle option\rangle  \hspace{1em} \text{(no default, initially all but AD)}

Defines, if the calendar setting is used for formatting. Feasible \langle option\rangle values are
- \textbf{all}: all calendar settings, including AD.
- \textbf{none}: no calendar settings.
- \textbf{all but AD}: all calendar settings, but excluding AD.

\begin{verbatim}
\gtrset{database/.cd,birth={\langle BC\rangle63-09-23}{Rome},death={\langle AD\rangle14-08-19}{Nola}}
\gtrPrintDate{birth} and died \gtrPrintDate{death}.
\gtrset{calendar print=none}
Augustus was born \gtrPrintDate{birth} and died \gtrPrintDate{death}.
\gtrset{calendar print=all}
Augustus was born \gtrPrintDate{birth} and died \gtrPrintDate{death}.
\gtrset{calendar print=none}
\end{verbatim}

Augustus was born September 23, 63 BC and died August 19, 14.
Augustus was born September 23, 63 and died August 19, 14.
Augustus was born September 23, 63 BC and died AD August 19, 14.
If the date is a *date range* with a start date and an end date, the \(\langle pre\rangle\), \(\langle mid\rangle\), and \(\langle app\rangle\) texts are placed appropriately. This setting is `/gtr/language` dependent.

```latex
\gtrset{database/.cd,birth={1354-02-09/1355-07-20}{Rome}}
\gtrset{date range full={between }{ and }{}}
%...
The birth was \gtrPrintDate{birth}.
```

The birth was between February 9, 1354 and July 20, 1355.

If the date is a *date range* an end date, but without start date, the \(\langle pre\rangle\) and \(\langle app\rangle\) texts are placed around the end date. This setting is `/gtr/language` dependent.

```latex
\gtrset{database/.cd,birth={/1355-07-20}{Rome}}
\gtrset{date range before={\textless\,}{}}{}
%...
The birth was \gtrPrintDate{birth}.
```

The birth was < July 20, 1355.

If the date is a *date range* a start date, but without end date, the \(\langle pre\rangle\) and \(\langle app\rangle\) texts are placed around the start date. This setting is `/gtr/language` dependent.

```latex
\gtrset{database/.cd,birth={1354-02-09/}{Rome}}
\gtrset{date range after={\textgreater\,}{}}{}
%...
The birth was \gtrPrintDate{birth}.
```

The birth was > February 9, 1354.

Sets the same separator text for `/gtr/date range full`, `/gtr/date range before`, `/gtr/date range after`. Use this for shortened range printing.

```latex
\gtrset{database/.cd,birth={1354-02-09/}{Rome}}
\gtrset{date range separator={--}}{}
%...
The birth was \gtrPrintDate{birth}.
```

The birth was February 9, 1354--.
7.8 Formatting Places

\gtrPrintPlace{⟨name⟩}

Used to insert a formatted place referred by ⟨name⟩. This ⟨name⟩ is an event name like birth, see Section 7.3 on page 165. The output format of the place is controlled by /gtr/place text.

\gtrset{database/.cd,birth={1354-02-09}{Rome}}
%...
The birth was \gtrPrintDate{birth} \gtrPrintPlace{birth}.

The birth was February 9, 1354 in Rome.

\gtrifplacedefined{⟨name⟩}{⟨true⟩}{⟨false⟩}

Expands to ⟨true⟩, if a place with the given ⟨name⟩ is defined, and to ⟨false⟩ otherwise.

\gtrset{database/.cd,birth={1354-02-09}{Rome}}
%...
The birth was \gtrPrintDate{birth} %
\gtrifplacedefined{birth}{\gtrPrintPlace{birth}}{}.

The birth was February 9, 1354 in Rome.

/gtr/place text={⟨pre⟩}{⟨app⟩} \quad (no default, initially {in }{})

The ⟨pre⟩ and ⟨app⟩ texts are placed around the place text. This setting is /gtr/language \textsuperscript{P.247} dependent.

\gtrset{database/.cd,birth={1354-02-09}{Rome}}
\gtrset{place text={}{}}
%...
The birth was \gtrPrintDate{birth} %
\gtrifplacedefined{birth}{\gtrPrintPlace{birth}}{}.

The birth was February 9, 1354 (Rome).
7.9 Formatting Events

\gtrPrintEvent\{\langle name \rangle\}

Used to insert a formatted event referred by \langle name \rangle. This \langle name \rangle is an event name like birth, see Section 7.3 on page 165. The output format of the event is controlled by /gtr/event text \textsuperscript{P.192}, \gtrPrintEventPrefix, \gtrPrintDate \textsuperscript{P.186}, and \gtrPrintPlace \textsuperscript{P.190}.

\%\%
\gtrset\{database/.cd,birth={1354-02-09}{Rome}\}
\%
\gtrPrintEvent\{birth\}

\% February 9, 1354 in Rome

\gtrIfEventDefined\{\langle name \rangle\}\{\langle true \rangle\}\{\langle false \rangle\}

Expands to \langle true \rangle, if an event with the given \langle name \rangle is defined, and to \langle false \rangle otherwise. This is an alias for \gtrIfDateDefined \textsuperscript{P.186}.

\%\%
\gtrset\{database/.cd,birth={1354-02-09}{Rome}\}
\%
\gtrIfEventDefined\{birth\}\{\gtrPrintEvent\{birth\}\}\{\}

\% February 9, 1354 in Rome

\gtrPrintEventPrefix\{\langle name \rangle\}

Used to insert an event prefix like a symbol. The prefix depends upon the \langle name \rangle of the event and upon an optional modifier. The output format of the prefix is controlled by the following options with the /gtr/event prefix path.

Birth: \gtrPrintEventPrefix\{birth\}
\par\gtrset\{/gtr/event prefix/birth=(b)\}  
Birth: \gtrPrintEventPrefix\{birth\}

Birth: \star
Birth: (b)

/gtr/event prefix/birth=\langle text \rangle \hspace{1cm} \text{(no default, initially \gtrsymBorn \textsuperscript{P.239})}
Prefix \langle text \rangle for a normal birth.

/gtr/event prefix/birth/out of wedlock=\langle text \rangle \hspace{1cm} \text{(no default, initially \gtrsymBornoutofwedlock \textsuperscript{P.239})}
Prefix \langle text \rangle for a birth out of wedlock.

/gtr/event prefix/birth/stillborn=\langle text \rangle \hspace{1cm} \text{(no default, initially \gtrsymStillborn \textsuperscript{P.239})}
Prefix \langle text \rangle for a birth of a stillborn child.

/gtr/event prefix/birth/died=\langle text \rangle \hspace{1cm} \text{(no default, initially \gtrsymDiedonbirthday \textsuperscript{P.239})}
Prefix \langle text \rangle for a birth if a child who died on birthday.

/gtr/event prefix/baptism=\langle text \rangle \hspace{1cm} \text{(no default, initially \gtrsymBaptized \textsuperscript{P.239})}
Prefix \langle text \rangle for a baptism.

/gtr/event prefix/engagement=\langle text \rangle \hspace{1cm} \text{(no default, initially \gtrsymEngaged \textsuperscript{P.240})}
Prefix \langle text \rangle for an engagement.

/gtr/event prefix/marriage=\langle text \rangle \hspace{1cm} \text{(no default, initially \gtrsymMarried \textsuperscript{P.240})}
Prefix \( \langle \text{text} \rangle \) for a normal marriage.

\[
\text{/gtr/event prefix/marriage/other=\langle\text{text}\rangle} \quad \text{(no default, initially \text{\gtrsymPartnership} \rightarrow \text{P. 240})}
\]

Prefix \( \langle \text{text} \rangle \) for another partnership.

\[
\text{/gtr/event prefix/divorce=\langle\text{text}\rangle} \quad \text{(no default, initially \text{\gtrsymDivorced} \rightarrow \text{P. 240})}
\]

Prefix \( \langle \text{text} \rangle \) for a divorce.

\[
\text{/gtr/event prefix/floruit=\langle\text{text}\rangle} \quad \text{(no default, initially \text{\gtrsymFloruit} \rightarrow \text{P. 241})}
\]

Prefix \( \langle \text{text} \rangle \) for a floruit event.

\[
\text{/gtr/event prefix/death=\langle\text{text}\rangle} \quad \text{(no default, initially \text{\gtrsymDied} \rightarrow \text{P. 240})}
\]

Prefix \( \langle \text{text} \rangle \) for a normal death.

\[
\text{/gtr/event prefix/death/killed=\langle\text{text}\rangle} \quad \text{(no default, initially \text{\gtrsymKilled} \rightarrow \text{P. 240})}
\]

Prefix \( \langle \text{text} \rangle \) for a death in war.

\[
\text{/gtr/event prefix/burial=\langle\text{text}\rangle} \quad \text{(no default, initially \text{\gtrsymBuried} \rightarrow \text{P. 241})}
\]

Prefix \( \langle \text{text} \rangle \) for a normal burial.

\[
\text{/gtr/event prefix/burial/cremated=\langle\text{text}\rangle} \quad \text{(no default, initially \text{\gtrsymFuneralurn} \rightarrow \text{P. 241})}
\]

Prefix \( \langle \text{text} \rangle \) for a cremation.

\[
\text{/gtr/event format=\langle\{\text{format}\}\rangle} \quad \text{(no default, initially \text{\d.M.yyyy})}
\]

This option controls how events are formatted when using \text{\gtrPrintEvent} \rightarrow \text{P. 191}.

Feasible \( \langle \text{format} \rangle \) values are

\[
\begin{align*}
\text{prefix date place} &\quad \star \text{February 9, 1354 in Rome} \\
\text{prefix date} &\quad \star \text{February 9, 1354} \\
\text{date} &\quad \text{February 9, 1354}
\end{align*}
\]

\[
\text{/gtr/event text=\langle\{\text{pre}\}\{\{\text{sep date}\}\{\{\text{sep place}\}\{\{\text{app}\}\}}\rangle} \quad \text{(no default, initially {\{}{-}{\}{}{}}})
\]

The four text pieces are placed inside \text{\gtrPrintEvent} \rightarrow \text{P. 191} as follows:

\[
\langle\text{pre}\rangle\langle\text{sep date}\rangle\langle\text{sep place}\rangle\langle\text{app}\rangle
\]

This setting is \text{not} \text{/gtr/language} \rightarrow \text{P. 247} dependent.

\[
\begin{verbatim}
\gtrset{database/.cd,birth={1354-02-09}{Rome}}
\gtrset{event text={\{}{-}{\}{}{}}}
\gtrPrintEvent{birth}

\text{\textcolor{red}{\star}: February 9, 1354 in Rome}
\end{verbatim}
\]

\[
\text{/gtr/event code=\langle\{\text{code}\}\rangle} \quad \text{(no default)}
\]

Defines \( \langle \text{code} \rangle \) to be executed by \text{\gtrPrintEvent} \rightarrow \text{P. 191}.

Use this, if \text{/gtr/event format} and \text{/gtr/event text} are not flexible enough.

\[
\begin{verbatim}
\gtrset{database/.cd,birth={1354-02-09}{Rome}}
\gtrset{event code=%}
\gtrPrintEventPrefix{#1} \quad \gtrifplacedefined{#1}{((\gtrPrintPlace{#1}){}{})%}
\gtrPrintDate{#1} %
\}
%...
\gtrPrintEvent{birth}

\text{\textcolor{red}{\star}: (in Rome) February 9, 1354}
\end{verbatim}
\]

192
7.10 Formatting Lists of Events

\begin{gtrprintlist}{\langle \text{first}\rangle}\{\langle \text{middle}\rangle\}\{\langle \text{last}\rangle\}\{\langle \text{empty}\rangle\}

\langle \text{environment content} \rangle
\end{gtrprintlist}

This environment is intended for automatically generated content. Inside this environment, a macro \texttt{gtrlistseparator} is defined.

- \texttt{gtrlistseparator} expands to \texttt{\langle \text{first}\rangle}, when it is called the first time.
- \texttt{gtrlistseparator} expands to \texttt{\langle \text{middle}\rangle}, when it is called later.
- \texttt{\langle \text{last}\rangle} is used at the end of the environment, if \texttt{gtrlistseparator} was called at least once.
- \texttt{\langle \text{empty}\rangle} is used at the end of the environment, if \texttt{gtrlistseparator} was never called.

\begin{gtrprintlist}
% \{\unskip\}\{\unskip,\ }\{\unskip.\}\{\unskip\}
gtrlistseparator One
\gtrlistseparator Two
\gtrlistseparator Three
\gtrlistseparator Four
\end{gtrprintlist}

One, Two, Three, Four.

\begin{gtrprintlist}
% \{\begin{itemize}\item\end{itemize}\item\}
\gtrlistseparator One
\gtrlistseparator Two
\gtrlistseparator Three
\gtrlistseparator Four
\end{gtrprintlist}

- One
- Two
- Three
- Four

\begin{gtreventlist}
\langle \text{environment content} \rangle
\end{gtreventlist}

This is a \texttt{gtrprintlist} environment with parameters specified by /\texttt{gtr/list separators} \textsuperscript{P.194}. This environment is used internally by most /\texttt{gtr/database format} \textsuperscript{P.174} settings to print event lists.

\gtrset{list separators=\{\unskip\}\{\unskip.\}\{\unskip\}}

\begin{gtreventlist}
\gtrlistseparator One
\gtrlistseparator Two
\gtrlistseparator Three
\gtrlistseparator Four
\end{gtreventlist}

One, Two, Three, Four.
Defines `gtreventlist`\textsuperscript{P.193} as a `gtrprintlist`\textsuperscript{P.193} with the given parameters. This is used to list events.

\begin{genealogypicture}
\begin{itemize}
\item processing=database,database format=full,
\item node size=4cm, level size=2cm,
\item name code={\gtrDBname},
\item list separators={ }{, }{.}{},
\item info separators={ }{. }{.}{},
\item box={fit basedim=9pt,boxsep=2pt,segmentation style=solid, halign=left,before upper=\parskip1pt,\gtrDBsex}
\end{itemize}
\end{genealogypicture}

\begin{genealogypicture}
\begin{itemize}
\item processing=database,database format=full,
\item node size=3cm, level size=3.2cm,
\item list separators hang=2mm,place text={\newline}{},
\item box={fit basedim=9pt,boxsep=2pt,segmentation style=solid, halign=left,before upper=\parskip1pt,\gtrDBsex}
\end{itemize}
\end{genealogypicture}
7.11 Formatting Comments

\texttt{\textbackslash gtrPrintComment}

Used to insert the formatted comment. May be redefined directly or using \texttt{/gtr/comment code}.

\begin{verbatim}
\gtrset{database/.cd,comment={Had a store in Boston}}
%...
\gtrPrintComment

Had a store in Boston
\end{verbatim}

\texttt{\textbackslash gtrifcommentdefined\{(true)\}{(false)\}}

Expands to \texttt{(true)}, if a comment is defined, and to \texttt{(false)} otherwise.

\begin{verbatim}
\gtrset{database/.cd,comment={Had a store in Boston}}
%...
\gtrifcommentdefined{\gtrPrintComment}{}

Had a store in Boston
\end{verbatim}

\texttt{\textbackslash gtr/comment code=\{(code)\}} (no default, initially \texttt{\itshape\textbackslash gtrDBcomment})

Redefines \texttt{\textbackslash gtrPrintComment} using \texttt{(code)}.

\begin{verbatim}
\gtrset{database/.cd,comment={Had a store in Boston}}
\gtrset{comment code=\{(\textbackslash gtrDBcomment)\}}
%...
\gtrPrintComment

(Had a store in Boston)
\end{verbatim}
7.12 Formatting Professions

\texttt{\textbackslash gtrPrintProfession}

Used to insert the formatted profession. May be redefined directly or using
\texttt{\textbackslash gtr/profession code}.

\texttt{\textbackslash gtrset\{database/.cd,profession={Copper smith, soldier}\}}

\texttt{\textbackslash gtrPrintProfession}

\textbackslash gtrifprofessiondefined\{(\texttt{true})\}{\{(\texttt{false})\}}

Expands to \texttt{(true)}, if a profession is defined, and to \texttt{(false)} otherwise.

\texttt{\textbackslash gtrset\{database/.cd,profession={Copper smith, soldier}\}}

\texttt{\textbackslash gtrifprofessiondefined\{\textbackslash gtrPrintProfession\}\{\}}

\texttt{Copper smith, soldier}

\texttt{\textbackslash gtr/profession code=\{(\texttt{code})\}} \quad \text{(no default, initially \texttt{\{itshape\textbackslash gtrDBprofession\}})}

Redefines \texttt{\textbackslash gtrPrintProfession} using \texttt{(code)}.

\texttt{\textbackslash gtrset\{database/.cd,profession={Copper smith, soldier}\}}

\texttt{\textbackslash gtrset\{profession code=\{(\textbackslash gtrDBprofession)\}\}}

\texttt{\textbackslash gtrPrintProfession}

(Copper smith, soldier)
7.13 Formatting Lists of Information

This is a \texttt{gtrinfolist} environment with parameters specified by \texttt{/gtr/info separators}. This environment is used internally by most \texttt{/gtr/database format} settings to print information lists consisting of \texttt{/gtr/database/profession} and \texttt{/gtr/database/comment} entries.

\texttt{/gtr/info separators} =\{\langle \text{first} \rangle \} \{\langle \text{middle} \rangle \} \{\langle \text{last} \rangle \} \{\langle \text{empty} \rangle \}\ (\text{no default, initially }\{\text{par}\}. )\{. \}\{. \}\{. \}

Defines \texttt{gtrinfolist} as a \texttt{gtrprintlist} with the given parameters. This is used to list informations.
7.14 Formatting Sex

\texttt{\textbackslash gtrPrintSex}

Used to insert a symbolic sign for the sex.

\begin{verbatim}
\gtrset{database/.cd, sex=female}
%...
\gtrPrintSex
\end{verbatim}

\texttt{\textbackslash gtriffemale\{true\}\{false\}}

Expands to \texttt{true}, if \texttt{\gtrDBsex} holds \texttt{female}, and to \texttt{false} otherwise.

\texttt{\textbackslash gtrifmale\{true\}\{false\}}

Expands to \texttt{true}, if \texttt{\gtrDBsex} holds \texttt{male}, and to \texttt{false} otherwise.

Note that the content of the data key \texttt{/gtr/database/sex} \(\textsuperscript{P.165}\) is accessible by \texttt{\gtrDBsex}. Since \texttt{/tcb/female} \(\textsuperscript{P.101}\), \texttt{/tcb/male} \(\textsuperscript{P.101}\), \texttt{/tcb/neuter} \(\textsuperscript{P.101}\), and \texttt{/gtr/female} \(\textsuperscript{P.101}\), \texttt{/gtr/male} \(\textsuperscript{P.101}\), \texttt{/gtr/neuter} \(\textsuperscript{P.101}\) are defined, \texttt{\gtrDBsex} can be used directly as a formatting option, see Section 7.2 on page 163 and the examples in Section 7.5 on page 174.
### 7.15 Formatting Images

\texttt{\gtrifimagedefined\{(true)\}{(false)}}

Expands to \texttt{(true)}, if an image is defined, and to \texttt{(false)} otherwise.

\begin{verbatim}
\gtrset{database/.cd,image=C. F. Gauss.jpg}
\%
\gtrifimagedefined{\includegraphics[width=3cm]{\gtrDBimage}}{no image}
\end{verbatim}

\begin{itemize}
\item \texttt{\gtrincludeDBimage\{(options)\}}
\end{itemize}

Includes a database image with the given \texttt{(options)}. These options are added to any options given by \texttt{/gtr/database/imageopt} \textsuperscript{P. 166}.

\begin{verbatim}
\gtrset{database/.cd,image=C. F. Gauss.jpg}
\%
\gtrifimagedefined{\gtrincludeDBimage\{width=3cm\}}{no image}
\end{verbatim}

\begin{itemize}
\item \texttt{\tcb/\textit{if image defined}={\texttt{(true)}}{\texttt{(false)}}}
\end{itemize}

Sets \texttt{(true)} \texttt{tcolorbox} options, if an image is defined, and sets \texttt{(false)} \texttt{tcolorbox} options otherwise. This key is intended to be used inside \texttt{/gtr/box} \textsuperscript{P. 98} constructs.

\begin{verbatim}
\gtrset{options for node={mynode}{
  box={if image defined={watermark graphics=\gtrDBimage}{}}}
}
\%
\end{verbatim}
Add a prefix \langle text \rangle to every image file name.

\gtrset{image prefix=picturedir/}
\gtrset{database/.cd,image=mytest.jpg}
\%...
Picture file: \texttt{\gtrDBimage}

Picture file: picturedir/mytest.jpg

/ \texttt{tikz/fill zoom DBimage} (no value, initially unset)
Fills the current path with the current database image. The image is zoomed such that it fits inside the path area. Typically, some parts of the path area will stay unfilled.

\gtrset{database/.cd,image=Carl_Friedrich_Gauss.jpg,viewport=30pt 50pt 150pt 180pt}
\%...
\gtrifimagedefined{%
\tikz\path[draw=red,fill zoom DBimage] (0,0) rectangle (2,3);
\tikz\path[draw=red,fill zoom DBimage] (0,0) rectangle (3,3);
\tikz\path[draw=red,fill zoom DBimage] (0,0) rectangle (3,2);
}{no image}

/ \texttt{tikz/fill overzoom DBimage} (no value, initially unset)
Fills the current path with the current database image. The image is zoomed such that the path area fills the image.

\gtrset{database/.cd,image=Carl_Friedrich_Gauss.jpg,viewport=30pt 50pt 150pt 180pt}
\%...
\gtrifimagedefined{%
\tikz\path[draw=red,fill overzoom DBimage] (0,0) rectangle (2,3);
\tikz\path[draw=red,fill overzoom DBimage] (0,0) rectangle (3,3);
\tikz\path[draw=red,fill overzoom DBimage] (0,0) rectangle (3,2);
}{no image}
7.16 Formatting Ages

\texttt{\textbackslash gliPrintAge}

Used to insert the formatted age. May be redefined directly or using \texttt{/gli/age code}.

\begin{itemize}
\item \texttt{\textbackslash gliSet\{database/.cd, age=\{87\}\}}
\item \texttt{\textbackslash gliPrintAge}
\end{itemize}

\begin{itemize}
\item \texttt{[87]}
\end{itemize}

\texttt{\textbackslash gliIfAgeDefined\{(true)\}\{(false)\}}

Expands to \texttt{(true)}, if an age is defined, and to \texttt{(false)} otherwise.

\begin{itemize}
\item \texttt{\textbackslash gliSet\{database/.cd, age=\{87\}\}}
\item \texttt{\textbackslash gliIfAgeDefined\{\textbackslash gliPrintAge\}\{}\texttt{\}}
\end{itemize}

\begin{itemize}
\item \texttt{[87]}
\end{itemize}

\texttt{\textbackslash gliAgeCode\{(code)\}}

(no default, initially \texttt{\textbackslash gliDBAge})

Redefines \texttt{\textbackslash gliPrintAge} using \texttt{(code)}.

\begin{itemize}
\item \texttt{\textbackslash gliSet\{database/.cd, age=\{87\}\}}
\item \texttt{\textbackslash gliSet\{age code=\{died aged~\textbackslash gliDBAge\}\}}
\item \texttt{\textbackslash gliPrintAge}
\end{itemize}

\begin{itemize}
\item died aged 87
\end{itemize}
Edges are drawn between all nodes of a family. For the auto-layout algorithm, the edges are opaque. Space is reserved for the edges according to the various distance settings for nodes, but the edge dimensions themselves are not considered during layout. The following settings and options influence the visual appearance of the edges.

Edges are drawn in two steps: a /gtr/edge/background \(^{P.214}\) followed by a /gtr/edge/foreground \(^{P.213}\). After all edges are drawn, the nodes are drawn (possibly over the edges).
8.1 Edge Settings

\texttt{/gtr/edges=\{\langle edge options \rangle\}}

(style, no default, initially perpendicular)

Defines the \langle edge options \rangle for drawing the edges between the nodes of a family. Normally, an edge is drawn with a \texttt{/gtr/edge/background} graph and a \texttt{/gtr/edge/foreground} graph to allow visual separation of superposed edges. This setting may be given globally, as option of \texttt{\genealogytree} or locally wrapped by \texttt{/gtr/family}. Also see Section 5.1.2 on page 79.

\begin{tikzpicture}
\genealogytree[template=signpost,edges=rounded]
{
  parent[id=SmithDoe]{
    g[id=Arth2008,male]{Arthur\textgtrsymBorn\,2008}
    c[id=Bert2010,female]{Berta\textgtrsymBorn\,2010}
    c[id=Char2014,male]{Charles\textgtrsymBorn\,2014}
    parent[id=Smith,family={edges={foreground={blue!50,line width=2mm}}}]{
      g[id=John1980,male]{John Smith\textgtrsymBorn\,1980}
      p[id=GpSm1949,male]{Grandpa Smith\textgtrsymBorn\,1949}
      p[id=GmSm1952,female]{Grandma Smith\textgtrsymBorn\,1952}
    }
    p[id=Jane1982,female]{Jane Doe\textgtrsymBorn\,1982}
  }
}
\end{tikzpicture}
This is a shortcut for embedding /gtr/edges → P. 204 into /gtr/family → P. 105.
This is a shortcut for embedding /gtr/edges \rightarrow P. 204 into /gtr/options for family \rightarrow P. 104.

\begin{tikzpicture}
\genealogytree[template=signpost,
edges for family={SmithDoe}{
foreground={red,line width=2pt},background={yellow,line width=3pt}},
]
{input{example.option.graph}}
\end{tikzpicture}

This is a shortcut for embedding /gtr/edges \rightarrow P. 204 into /gtr/options for subtree \rightarrow P. 108.

/gtr/subtree edges=\{(edge options)\} (style, no default)

This is a shortcut for embedding /gtr/edges \rightarrow P. 204 into /gtr/options for subtree \rightarrow P. 108.
\gtredgeset\{\textit{options}\}

Sets \textit{\textit{options}} for the /\texttt{gtr/edge} key subtree. Mainly, this macro is intended to easily set up styles for edges.

\begin{tikzpicture}
\gtredgeset\{myedges/.style={rounded=6pt,
foreground={blue!50!black},background={blue!20!white}}
\%
\genealogytree[template=signpost,edges=myedges]
{input{example.option.graph}}
\end{tikzpicture}

\textbf{/tikz\genealogytree edges scope} (style, initially empty)

This style is used to scope the drawing of the edges. It may be redefined e.g. to draw edges on a certain layer.

\begin{tikzpicture}
\genealogytree[template=signpost,edges=myedges]
{input{example.option.graph}}
\end{tikzpicture}

\texttt{\textbackslashusetikzlibrary{backgrounds}}

\texttt{\textbackslash tikzset\{\texttt{genealogytree edges scope/.style={on background layer}}\}}

Note that edges are drawn before nodes. Typically, the setting to draw on the background layer is not necessary. If two genealogy trees are merged, this additional setting may be useful.
8.2 Edge Types

/\texttt{gtr/edge/perpendicular} (no value, initially set)

The edges are drawn in a perpendicular style.

\begin{tikzpicture}
\genealogytree[template=signpost,edges={perpendicular}]
{input{example.option.graph}}
\end{tikzpicture}

/\texttt{gtr/edge/rounded=\langle length \rangle} (default 6pt)

The edges are drawn in a perpendicular but rounded style. The \langle length \rangle describes the size of the rounding.

\begin{tikzpicture}
\genealogytree[template=signpost,edges={rounded=6pt}]
{input{example.option.graph}}
\end{tikzpicture}
The edges are drawn in a swinging style. The \textit{length} describes the control parameter of the underlying curved path.

\begin{tikzpicture}
\genealogytree[template=signpost,edges={swing=12pt}]
\{input{example.option.graph}}
\end{tikzpicture}
The edges are drawn meshed between parents and children. The \textit{options} are TikZ to \textit{path} options.

- For a family without children or without parents, no edge will be drawn.
- For a family with at least two parents and at least two children, a mesh is drawn. The intended use case is for families with just one parent or just one child, i.e. for ordinary trees.

\begin{tikzpicture}
genealogytree[template=signpost,edges=mesh]
{input{example.option.graph}}
\end{tikzpicture}
The edges are drawn in a custom style. This option takes four mandatory parameters \(\langle\text{down}\rangle\), \(\langle\text{up}\rangle\), \(\langle\text{left}\rangle\), and \(\langle\text{right}\rangle\), each of which is a macro. The \(\langle\text{down}\rangle\) macro is used to draw edges for \texttt{/gtr/timeflow \{P.80=down\}}, etc.

Every macro has to take four mandatory parameters:

1. An \texttt{etoolbox \langle listmacro \rangle} which contains the list of anchor positions for the parents.
2. An \texttt{etoolbox \langle listmacro \rangle} which contains the list of anchor positions for the children.
3. A Ti\textit{kZ} node name which denotes the family core (center).
4. A Ti\textit{kZ} style which should be applied to draw the edges.

\begin{tikzpicture}
\genealogytree
\[template=signpost,\]
\[box={enhanced jigsaw,opacityback=0.75},\]
\[edges={\]
\[custom={\myedgedraw}{\myedgedraw}{\myedgedraw}{\myedgedraw},\]
\[}\]
\{input{example.option.graph}\}
\end{tikzpicture}
This is a special \texttt{/gtr/edge/custom} style which simply draws nothing. May be used for just this purpose or to replace automatic edge drawing by manual edge drawing.

\begin{tikzpicture}
\genealogytree[template=signpost,edges=none]
\input{example.option.graph}
\draw[Latex,-green!50!black,thick]
(GpSm1949) edge[out=270,in=270,"husband"] (GmSm1952)
edge[out=225,in=180,"father"] (John1980)
edge[out=180,in=180,"grandfather"] (Arth2008)
(Bert2010) edge[out=90,in=270,"daughter"] (Jane1982)
(Jane1982) edge[out=90,in=270,"daughter"] (GmDo1956);
\end{tikzpicture}
8.3 Edge Parameters

\texttt{/gtr/edge/foreground=\{(tikz options)\}} \hspace{1cm} (style, no default)

Defines the foreground \texttt{(tikz options)} for drawing the edges between the nodes.

```latex
\begin{tikzpicture}
genealogytree[template=signpost,
  edges={foreground={line width=2pt,red,dashed,line cap=butt},no background}]
{input{example.option.graph}}
\end{tikzpicture}
```

\texttt{/gtr/edge/no foreground} \hspace{1cm} (style, no value)

Removes the \texttt{/gtr/edge/foreground} edges.

```latex
\begin{tikzpicture}
genealogytree[template=signpost,level distance=1.7cm,
  edges={rounded,foreground={line width=2pt,red,Circle-LaTeX,shorten <=-4pt},
  background={line width=3pt,yellow}}]
{input{example.option.graph}}
\end{tikzpicture}
```
/tikz options (style, no default)
Defines the background /tikz options for drawing the edges between the nodes.

\begin{tikzpicture}
genealogytree[template=signpost, edges={foreground={line width=0.5pt,red}, background={line width=2pt,yellow}}]
{input{example.option.graph}}
\end{tikzpicture}

\begin{tikzpicture}
genealogytree[template=signpost, edges={swing=5mm,anchoring=center, foreground={line width=4mm},background={line width=5mm}}]
{input{example.option.graph}}
\end{tikzpicture}

/gtr/edge/background\{/(tikz options)\} (style, no default)
Defines the background /tikz options for drawing the edges between the nodes.

/gtr/edge/no background (style, no value)
Removes the /gtr/edge/background edges.

/gtr/edge/anchoring=periphery|center (no default, initially periphery)
Defines anchoring points for the edges. Feasible value are periphery and center.
/gtr/edge/hide single leg = true | false
(default true, initially true)

If set to true, the orphan leg of a family with just one member is hidden.

\begin{tikzpicture}
\genealogytree[template=signpost, edges={hide single leg}]
{ parent{ g{Orphan} } }
\end{tikzpicture}

\begin{tikzpicture}
\genealogytree[template=signpost, edges={hide single leg=false}]
{ parent{ g{Orphan} } }
\end{tikzpicture}

/gtr/edge/xshift = (length)
(no default, in initially 0pt)

Shifts the edge core position horizontally by \textit{(length)}.

\begin{tikzpicture}
\genealogytree[template=signpost, edges={swing,xshift=5mm}]
{ input{example.option.graph} }
\end{tikzpicture}
\begin{tikzpicture}
genealogytree[template=signpost,edges={swing,yshift=-3mm}]
{input{example.option.graph}}
\end{tikzpicture}
8.4 Edge Labels

\texttt{/gtr/label={\{text\}}} \quad \text{(style, no default)}

Adds a label \texttt{\{text\}} to the current family. This is realized by a TikZ node with \texttt{/gtr/label options}. The current family is determined by a surrounding \texttt{/gtr/family} \textsuperscript{P.105} or \texttt{/gtr/options for family} \textsuperscript{P.104}.

\begin{genealogypicture}[template=signpost, label options={fill=white,node font=\footnotesize}, options for family={SmithDoe}{label={\gtrsymMarried-2006}} ] input{example.option.graph} \end{genealogypicture}

\begin{genealogypicture}[template=signpost, label options={fill=green!20,node font=\footnotesize}, label={\gtrsymMarried} ] input{example.option.graph} \end{genealogypicture}

\texttt{/gtr/label options={\{options\}}} \quad \text{(style, no default)}

Sets TikZ node \texttt{\{options\}} to be used for \texttt{/gtr/label}. See \texttt{/gtr/label} for an example.
Shortcut for using `/gtr/label` \(\Rightarrow\) P. 217 inside `/gtr/family` \(\Rightarrow\) P. 105.

Shortcut for using `/gtr/label` \(\Rightarrow\) P. 217 inside `/gtr/subtree` \(\Rightarrow\) P. 108.
8.5 Edge Labels Database

Analog to database processing for nodes, see Chapter 7 on page 161, the edge labels can be formatted by database style entries.

The database content for edge labels has to be given inside the option list for a parent or child using /gtr/family database.

/gtr/family database={⟨data keys⟩} (no default, initially empty)
Sets ⟨data keys⟩ for the edge labeling of the current family. For ⟨data keys⟩, any setting from Section 7.3 on page 165 can be used, but only marriage information or similar may be reasonable.

%...
child[id=SmitBowd1742,family database={marriage={1742-03-02}{London}}]{%
%...

/gtr/label database options={⟨options⟩} (no default)
The ⟨options⟩ settings define how the /gtr/family database values are used to create label content. The default operations are /gtr/use family database and /gtr/database format → P. 174 = marriage. Note that setting /gtr/database format → P. 174 inside /gtr/label database options does only change the format for edge labels, but not for nodes.

%...
label database options={
    database format=marriage, % that is the default value
    place text={⟨⟩⟨⟩} % changed only for labels
},%
%...

/gtr/ignore family database (no value)
If set, then all /gtr/family database values are simply ignored. This has to be used inside /gtr/label database options to have an effect.

%...
label database options={ignore family database},%
%...

/gtr/use family database (no value)
If set, then all /gtr/family database values are processed to generate label content. This has to be used inside /gtr/label database options to have an effect.
Charles Smith
★ ca. 1722
London
× October 12, 1764
Copper smith, soldier.

Jane Bowden
★ ca. 1724
† July 7, 1802
New York

Abraham Bowden
★ January 4, 1740
London
† February 23, 1740
London

Elizabeth “Liz” Smith
★ February 2, 1744
London
† April 12, 1812
Boston
Had a store in Boston.

Michael Smith
★† March 1, 1758

March 2, 1742 (London)
8.6 Adding and Removing Nodes from Edge Drawing

/\texttt{\texttt{gtr/add child=(child) to (family)}} \hspace{1cm} (style, no default)

Connect a node of an existing graph as \texttt{(child)} to a \texttt{(family)} of the current graph. The auto-layout algorithm is not aware of this addition.

\begin{tikzpicture}[scale=0.9,transform shape]
\gtrset{template=signpost}
\genealogytree[
edges={foreground={red!50!blue,line width=2pt},
   background={red!50!blue!15!white,line width=3pt},
   options for node={GmDo1956}{box={colback=red!30!white}}]
\input{example.option.graph}
\genealogytree[
edges={foreground={green!50!blue,line width=2pt},
   background={green!50!blue!15!white,line width=3pt},
   box={colback=green!30!white},
   adjust node=PhDo1982 right of Harr1987 distance 3mm,
   add child=GmDo1956 to GreatDoe}
child[id=GreatDoe,pivot shift=1.7cm]{
g[id=GgpDo1910,male]{Great-Grandpa Doe \gtrsymbol{Born},1910}
p[id=GgmDo1918,female]{Great-Grandma Doe \gtrsymbol{Born},1918}
child{
g[id=JaDo1957,male]{Jack Doe \gtrsymbol{Born},1957}
p[id=MaJo1960,female]{Mary Jones \gtrsymbol{Born},1960}
child{
g[id=PhDo1982,male]{Phil Doe \gtrsymbol{Born},1982}
p[id=SyPo1982,female]{Sybil Porter \gtrsymbol{Born},1982}
c[id=Will2005,male]{Will \gtrsymbol{Born},2005}
c[id=Xave2005,male]{Xaver \gtrsymbol{Born},2005}
c[id=Zeb2010,male]{Zeb \gtrsymbol{Born},2010}
}
c[id=AnDo1984,female]{Anne Doe \gtrsymbol{Born},1984}
}
}
\end{tikzpicture}
Connect a node of an existing graph as \texttt{parent} to a \texttt{family} of the current graph. The auto-layout algorithm is not aware of this addition.
An alternative approach to /gtr/add child \textsuperscript{P.221} and /gtr/add parent \textsuperscript{P.222} is to draw the interconnecting node twice (the first one could be drawn as /gtr/phantom\textsuperscript{P.127}). The second instance is drawn over the first instance using /gtr/set position\textsuperscript{P.111}. Both instances need to have different /gtr/id\textsuperscript{P.92} values. Note that both parts of the graph can still be overlapping and may have to be adjusted manually, since the auto-layout algorithms handles each \genealogytree\textsuperscript{P.55} separately. The second tree gets an /gtr/id prefix\textsuperscript{P.94} of 2: to address the second Uncle Harry by 2:Harr1987.

\begin{tikzpicture}
\gtrset{template=signpost}
\genealogytree[
edges={foreground={red!50!blue,line width=2pt},
background={red!50!blue!15!white,line width=3pt}},
]
{input{example.option.graph}}
\genealogytree[id prefix=2:,
edges={foreground={green!50!blue,line width=2pt},
background={green!50!blue!15!white,line width=3pt}},
box={colback=green!30!white},
set position=2:Harr1987 at Harr1987,
]
{
parent[id= DoeJones]{
  g[id=Deir2012,female]{Deirdre\\gtrsymBorn\,2012}
  p[id=Harr1987,male,box={colback=red!30!white}]{Uncle Harry\\gtrsymBorn\,1987}
}
parent[id= Jones]{
  g[id=Mary1988,female,distance=1.4cm]{Aunt Mary\\gtrsymBorn\,1988}
  p[id=JimJ1944,male]{Jim Jones\\gtrsymBorn\,1944}
  p[id=Jenn1949,female]{Jenny Jones\\gtrsymBorn\,1949}
}
}
\end{tikzpicture}
/gtr/remove child=(child) from (family)  (style, no default)

Removes a node as (child) from a (family) of the current graph. The auto-layout algorithm is not aware of this removal.

\begin{tikzpicture}
\genealogytree[template=signpost, options for node={Jane1982}{pivot=child}, remove child=Harr1987 from Doe, extra edges prepend for family= {Doe}{GmDo1956}{Harr1987}{foreground={dotted,line cap=round}, no background}]
\end{tikzpicture}

/gtr/remove parent=(parent) from (family)  (style, no default)

Removes a node as (parent) from a (family) of the current graph. The auto-layout algorithm is not aware of this removal.

\begin{tikzpicture}
\genealogytree[template=signpost, remove parent=GpSm1949 from Smith ]
\end{tikzpicture}
Using this option inside node options disconnects the current node from the edges of the current graph. Using this option elsewhere may cause unwanted side effects. The auto-layout algorithm is not aware of this removal. Depending of the given \(<value>\), the node is disconnected as parent or child or both.

Feasible values are
- **child**: disconnect the node as child of a family.
- **parent**: disconnect the node as parent of a family. Note that a \(g\) node is only removed from its primary family, but not from connected union families.
- **both**: disconnect the node as child and as parent.

/gtr/remove child \(^{P.224}\) and /gtr/remove parent \(^{P.224}\) allow more precise control, but /gtr/disconnect needs no /gtr/id \(^{P.92}\) values.
8.7 Extra Edges

\[ /gtr/extra\ edges={\langle\text{parents}\rangle}\{\langle\text{children}\rangle\}\{\langle\text{edge\ options}\rangle\} \] (style, no default)

Appends an extra set of edges to the current family. The edges are drawn between the given \langle parents \rangle list and the given \langle children \rangle list using the \langle edge options \rangle. Note that parents and children are defined by their \text{/gtr/id} values. They do not necessarily have to be real members of the current family. The current family is given by a surrounding \text{/gtr/family} or \text{/gtr/options for family}.

This is a shortcut for embedding \text{/gtr/extra edges} into \text{/gtr/family}.
This is a shortcut for embedding \texttt{/gtr/extra edges for family} into \texttt{/gtr/options for family}. This allows to set \texttt{/gtr/extra edges for family} for multiple families. Therefore, the \texttt{family list} is a comma separated list of entries of type \texttt{x={family}\{parents\}\{children\}\{edge options\}}.
\gtr{extra edges prepend}{\{parents\}\{children\}\{edge options\}} \hspace{1cm} (style, no default)

Appends an extra set of edges to the current family. The edges are drawn between the given \textit{parents} list and the given \textit{children} list using the \textit{edge options}. This is identical to \gtr{extra edges} \textsuperscript{P.226}, but the drawing lies under the normal edges.

\begin{tikzpicture}
\genealogytree
\options for family={SmithDoe}{extra edges prepend={Jane1982}{Arth2008}{
foreground={red!25!yellow,line width=5pt},no background}},
\end{tikzpicture}

\gtr{family extra edges prepend}{\{parents\}\{children\}\{edge options\}} \hspace{1cm} (style, no default)

This is a shortcut for embedding \gtr{extra edges prepend} into \gtr{family} \textsuperscript{P.105}.
This is a shortcut for embedding `/gtr/extra edges prepend` \(\cdot P.228\) into `/gtr/options for family` \(\cdot P.104\).

This allows to set `/gtr/extra edges prepend for family` for multiple families. Therefore, the `siblings` is a comma separated list of entries of type $x=\{(family)\}\{(_edge options)\}$

\begin{tikzpicture}
\genealogytree[template=signpost,
extra edges prepend for families={x=\{Doe\}{GmDo1956}{Jane1982},
x=\{SmithDoe\}{Jane1982}{Arth2008}\}
foreground=(red!25!yellow,line width=5pt),no background],
] {input{example.option.graph}}
\end{tikzpicture}
This style is used to scope the drawing of extra edges. It may be redefined e.g. to draw edges on a certain layer. This scope is embedded into the general scope of \texttt{/tikz/genealogytree edges scope} \cite{p.207}.

% draw extra edges on the background layer
\usetikzlibrary{backgrounds}
\tikzset{genealogytree extra edges scope/.style={on background layer}}

Note that changing the drawing layer for extra edges applies to all extra edges, but does not change the drawing layer for normal edges. Therefore, all extra edges would be drawn behind normal edges, if the example above is used.
8.8 Edge Shifting

Typically, the edge core position of a family is located on half the way between the parents and the children. But, this is true only, if no union constructs are used, where a man or woman has two or more relationships with or without children. For example, two relationships, the edge core positions are located on one-third and on two-thirds of the way between the parents and the children. This edge shift is computed automatically.

\begin{genealogypicture}[template=signpost]
child{
  p{Husband1}
  g{Wife} c{Child1} c{Child2}
  union{ p{Husband2} c{Child3} c{Child4} c{Child5} c{Child6} }
}
\end{genealogypicture}

Depending on the situation, the edges may cross or not.

\begin{genealogypicture}[template=signpost]
child{
  p{Husband1}
  g{Wife} c{Child1} c{Child2} c{Child3} c{Child4}
  union{ p{Husband2} c{Child5} }
}
\end{genealogypicture}

The automatic shift does not prevent edge crossing, but ensures to a certain degree that one edge does not hide another edge.
Nodes could be shifted inside their level (here horizontally) to influence the edge drawing, see /gtr/pivot → P.97, /gtr/pivot shift → P.106, /gtr/distance → P.96, etc.

\begin{genealogypicture}[template=signpost]
  child{
    p{Husband1}
    g{Wife} c{Child1} c{Child2} c{Child3} c{Child4} c[pivot]{Child4}
    union{ p{Husband2} c{Child5} }
  }
\end{genealogypicture}

In the following, options for manual edge shifting to avoid edge crossings are described.

\textbf{/gtr/edges shift=(length)} (style, no default) N 2017-12-08

Shifts the edge core position by \texttt{(length)} towards the parents. In contrast to /gtr/edge/xshift → P.215 and /gtr/edge/yshift → P.216, the /gtr/timeflow → P.80 is automatically respected. Also note that this is a /gtr/family edges → P.205 option, i.e. it can be applied directly inside the option list of a family.

\begin{genealogypicture}[template=signpost]
  child[edges shift=-3mm]{
    p{Husband1}
    g{Wife} c{Child1} c{Child2} c{Child3} c{Child4}
    union[edges shift=3mm]{ p{Husband2} c{Child5} }
  }
\end{genealogypicture}
/gtr/edges up by\langle number\rangle of \langle parts\rangle

(\text{style, no default})

Shifts the edge core position by fraction \langle number\rangle/\langle parts\rangle of the /gtr/level distance\textsuperscript{P.83} towards the parents. This is a style which uses /gtr/edges shift\textsuperscript{P.232}.

\begin{genealogypicture}[template=signpost]
\child[edges down by=2 of 4]{
p{Husband1}
g{Wife} c{Child1} c{Child2} c{Child3} c{Child4}
union[edges up by=1 of 4]{ p{Husband2} } 
union[edges up by=1 of 4]{ p{Husband3} }
}
\end{genealogypicture}

/gtr/edges down by\langle number\rangle of \langle parts\rangle

(\text{style, no default})

Shifts the edge core position by fraction \langle number\rangle/\langle parts\rangle of the /gtr/level distance\textsuperscript{P.83} towards the children. This is a style which uses /gtr/edges shift\textsuperscript{P.232}.

\begin{genealogypicture}[template=signpost]
\child[edges down]{
p{Husband1}
g{Wife} c{Child1} c{Child2} c{Child3} c{Child4} c{Child5}
union[edges up]{ p{Husband2} c{Child5} }
}
\end{genealogypicture}

/gtr/edges up

(\text{no value})

Shifts the edge core position by one third of the /gtr/level distance\textsuperscript{P.83} towards the parents. This is a special case of /gtr/edges up by usable for two relationships.

\begin{genealogypicture}[template=signpost]
\child[edges down]{
p{Husband1}
g{Wife} c{Child1} c{Child2} c{Child3} c{Child4}
union[edges up]{ p{Husband2} c{Child5} }
}
\end{genealogypicture}

/gtr/edges down

(\text{no value})

Shifts the edge core position by one third of the /gtr/level distance\textsuperscript{P.83} towards the children. This is a special case of /gtr/edges up by usable for two relationships.
The following options can only be applied globally for a graph or the whole document. They cannot be applied locally to influence just one family.

`/gtr/reset edge level shift` (no value, initially set)
This sets the default algorithm for the automatic level shift. The first relationship is closest to the parents while the last relationship is closest to the children.

```
\begin{genealogypicture}[template=signpost,reset edge level shift]
  \child{
    p{Husband1}
    g{Wife} c{Child1} c{Child2} c{Child3} c[pivot]{Child4}
    union( p{Husband2} c{Child5} )
  }
\end{genealogypicture}
```

```
Husband1
  \node{Wife}
  Child1 Child2 Child3 Child4
  \node{Husband2}
  Child5
```

`/gtr/switch edge level shift` (no value, initially set)
This switches the algorithm for the automatic level shift. The first relationship is closest to the children while the last relationship is closest to the parents.

```
\begin{genealogypicture}[template=signpost,switch edge level shift]
  \child{
    p{Husband1}
    g{Wife} c{Child1} c{Child2} c{Child3} c[pivot]{Child4}
    union( p{Husband2} c{Child5} )
  }
\end{genealogypicture}
```

```
Husband1
  \node{Wife}
  Child1 Child2 Child3 Child4
  \node{Husband2}
  Child5
```
Sets all edge core positions to be on half the way between the parents and the children. Note that edge overlapping has to be prevented manually for all families. Use this option only, if you are really sure about that.

\begin{genealogypicture}[template=signpost,nullify edge level shift]
  child{
    p{Husband1}
    g{Wife} c{Child1} c{Child2} c{Child3} c[pivot]{Child4}
    union{ p{Husband2} c{Child5} }
  }
\end{genealogypicture}
9.1 Symbol Color Settings

If the genealogy symbols are only needed in black color, there is nothing special to consider. Currently, the symbols are drawn as pictures and saved in boxes for efficiency. If different colors are needed, the symbols have to be redrawn. The named color \texttt{gtrsymbol} holds the (current) symbol color.

9.1.1 Global Color Settings

In the preamble, the color of all genealogy symbols can be set by redefining the color \texttt{gtrsymbol}. For example, if all symbols should be created in blue, one can use:

\begin{verbatim}
% ...
\colorlet{gtrsymbol}{blue}
% ...
\begin{document}
% ...
\end{document}
\end{verbatim}

Note that this setting has to be given inside the preamble \textit{after} the package is loaded and \textit{before} \texttt{\begin{document}}.

9.1.2 Local Color Settings

If symbols with a color different from the global symbol color should be used inside the document, one of the following commands can be used.

\texttt{\textbackslash gtrSymbolsSetCreate\{\textit{color}\}\}

Recreates all symbols for the current \TeX{} group with the given \textit{color}. The named color \texttt{gtrsymbol} will also be set to \textit{color}. Use this macro, if it is expected that many symbols of this color will be used inside the current \TeX{} group.

\begin{verbatim}
\gtrsymBorn\,14.XI.1475
\{\gtrSymbolsSetCreate\{red\}\gtrsymMarried\,22.II.1502, \gtrsymDied\,8.X.1553,\}\gtrsymBuried\,10.X.1553
\end{verbatim}

\begin{verbatim}
\gtrSymbolsSetCreateSelected\{\textit{color}\}\{\textit{list}\}\}

Recreates all symbols from the given comma separated \textit{list} for the current \TeX{} group with the given \textit{color}. The named color \texttt{gtrsymbol} will also be set to \textit{color}. The \textit{list} contains the base names of the selected symbols, e.g. \texttt{Born} for \texttt{gtrsymBorn} \textsuperscript{P.239}. Symbols which are not present in this list, will keep their old color. Use this macro, if it is expected that many symbols of this color will be used inside the current \TeX{} group.

\begin{verbatim}
\gtrSymbolsSetCreateSelected\{blue\}\{Male\}\gtrSymbolsSetCreateSelected\{red\}\{Female\}\gtrSymbolsSetCreateSelected\{yellow!50!black\}\{Born,Died\}\gtrsymBorn, \gtrsymMale, \gtrsymFemale, \gtrsymNeuter, \gtrsymDied.
\end{verbatim}

\texttt{\textbackslash gtrSymbolsSetDraw\{\textit{color}\}\}

Inside the current \TeX{} group, every symbol is drawn with the given \textit{color} when it is used. It is drawn again, if it is used again. The named color \texttt{gtrsymbol} will also be set to \textit{color}. Use this macro, if it is expected that only few symbols of this color will be used inside the current \TeX{} group or if colors constantly change.

\begin{verbatim}
\gtrsymBorn\,14.XI.1475
\{\gtrSymbolsSetDraw\{red\}\gtrsymMarried\,22.II.1502, \gtrSymbolsSetDraw\{blue\}\gtrsymDied\,8.X.1553,\}\gtrsymBuried\,10.X.1553
\end{verbatim}

\begin{verbatim}
\gtrsymBorn\,14.XI.1475 \gtrsymMarried\,22.II.1502, \gtrsymDied\,8.X.1553, \gtrsymBuried\,10.X.1553
\end{verbatim}

238
9.2 List of Symbols

\textbf{\LaTeXsymBorn}\, \star

Birth / born (Unicode \texttt{U+2A}).

\begin{itemize}
\item Johann Maier \LaTeXsymBorn, 14.XI.1475
\item Johann Maier \star, 14.XI.1475
\end{itemize}

\textbf{\LaTeXsymBornoutofwedlock}\, (\textasteriskcentered)

Born out of wedlock / illegitimate.

\begin{itemize}
\item Johann Maier \LaTeXsymBornoutofwedlock, 14.XI.1475
\item Johann Maier (\textasteriskcentered), 14.XI.1475
\end{itemize}

\textbf{\LaTeXsymStillborn}\, \dagger\star

Stillborn.

\begin{itemize}
\item \textit{Anonymus} Maier \LaTeXsymStillborn, 14.XI.1475
\item Anonymus Maier \dagger\star, 14.XI.1475
\end{itemize}

\textbf{\LaTeXsymDiedonbirthday}\, \star\dagger

Died on the birthday.

\begin{itemize}
\item Johann Maier \LaTeXsymDiedonbirthday, 14.XI.1475
\item Johann Maier \star\dagger, 14.XI.1475
\end{itemize}

\textbf{\LaTeXsymBaptized}\, \approx

Baptism / baptized (Unicode \texttt{U+2248}).

\begin{itemize}
\item Johann Maier \LaTeXsymBaptized, 14.XI.1475
\item Johann Maier \approx, 14.XI.1475
\end{itemize}
Engagement / engaged (Unicode U+26AC).

Marriage / married (Unicode U+26AD).

Divorce / divorced (Unicode U+26AE).

Partnership / unmarried (Unicode U+26AF).

Death / died (Unicode U+2020, U+271D).

Killed in action / fallen (Unicode U+2694).
Johann Maier \textgrantsignKilled\,14.XI.1475

Johann Maier \textgrantsign14.XI.1475

\textgrantsignBuried \emptyset

Burial / buried (Unicode U+26B0).

Johann Maier \textgrantsignBuried\,14.XI.1475

Johann Maier \textgrantsign14.XI.1475

\textgrantsignFuneralurn \emptyset

Funeral urn / cremated (Unicode U+26B1).

Johann Maier \textgrantsignFuneralurn\,14.XI.1475

Johann Maier \textgrantsign14.XI.1475

\textgrantsignFloruit ☼

Floruit / flourished.

Johann Maier \textgrantsignFloruit\,1475--1503

Johann Maier \textgrantsign1475–1503

\textgrantsignFemale ♀

Female (Unicode U+2640).

Maria Maier \textgrantsignFemale

Maria Maier ♀

\textgrantsignMale ♂

Male (Unicode U+2642).

Johann Maier \textgrantsignMale

Johann Maier ♂
Neuter / Unknown sex (Unicode U+26B2).

\textit{Anonymus} Maier \textit{\textgreek{\textalpha}}

Anonymus Maier \textit{\textgreek{\textalpha}}
9.3 Legend to Symbols

The further macros and options allow to create a legend to symbols. This legend contains either all symbols or only the currently used symbols. Also, the description texts can be adapted to different languages or individual settings.

9.3.1 Printing a Legend

\gtrSymbolsRecordReset
The occurrence of a symbol inside the document text is recorded. \gtrSymbolsLegend\textsuperscript{P. 244} prints all recorded symbols. To clear the current recording (locally), \gtrSymbolsRecordReset can be used. Note that records are taken globally, but resets are local to the current \TeX group.

\gtrSymbolsRecordReset
Use symbol: \gtrsymBorn\par
{\gtrSymbolsRecordReset
Use symbol inside group: \gtrsymMarried\par
{Use symbol further inside: \gtrsymDied\par
}Local legend inside group: \gtrSymbolsLegend\par
}Global legend: \gtrSymbolsLegend

Use symbol: ★
Use symbol inside group: ⊗
Use symbol further inside: †
Local legend inside group: ⊗=married, †=died.
Global legend: ★=born, ⊗=married, †=died.

/gtr/symbols record reset (no value)
Identical to \gtrSymbolsRecordReset. This option is useful for application inside \genealogytree\textsuperscript{P. 55} or genealogypicture\textsuperscript{P. 57}. See \gtrSymbolsLegend\textsuperscript{P. 244} for an example.

\textsl{\gtrSymbolsFullLegend[(language)]}
Prints a full unabridged legend to symbols according to /gtr/language\textsuperscript{P. 247} or optionally according to \langle language\rangle.

\textsl{\gtrSymbolsFullLegend[english]}
★=born, (★)=born out of wedlock, ★★=stillborn, ★†=died on the birthday, ≈=baptized, ○=engaged, ⊗=married, ○○=divorced, ○●=unmarried partnership, ★★=floruit, †=died, X= killed in action, 0=buried, 0=cremated, ♀=female, ♂=male, ♀=neuter.
Prints a legend to symbols according to /gtr/language \P.247 or optionally according to \langle language \rangle. The legend only contains these symbols which were actually used inside the document so far or since the last call to \gtrSymbolsRecordReset \P.243 or /gtr/symbols record reset \P.243.

\begin{genealogypicture}[node size=3.2cm, level size=2.5cm, box={fit basedim=9pt, boxsep=2pt, colback=red!10, segmentation style={solid}}, symbols record reset, code={\gtrSymbolsSetCreate(blue)}, after tree={\node[font={\scriptsize\itshape}, text width=3cm, above right, fill=blue!5] at ([xshift=1cm]Michael.south east) {\gtrSymbolsLegend};}, ]
  child{
    g{id=Charles}{Charles Smith
      \gtrsymBorn-1722 London
      \gtrsymBaptized-13.IV.1722\ London
      \gtrsymKilled-12.X.1764}
    p{Jane Bowden
      \gtrsymBorn-1724\ London
      \gtrsymDied-7.VII.1802\ New York
      \gtrsymMarried-2.III.1742\ London}
    c{Abraham Bowden
      \gtrsymBornoutofwedlock-4.I.1740 London
      \gtrsymDied-23.II.1740 London}
    c{Elizabeth Smith
      \gtrsymBorn-2.II.1742 London
      \gtrsymFloruit-1780--1805 New York
      \gtrsymDied-12.IV.1812 Boston}
    c{id=Michael}{Michael Smith
      \gtrsymDiedonbirthday-1.III.1761}
  }
\end{genealogypicture}

9.3.2 Description Texts and Language Settings

The following options of the key family /gtr/symlang/ are language dependent and can be set globally by /gtr/language \P.247.

\begin{english}
\gtrset{language=english}
% ...
\gtrSymbolsFullLegend
\end{english}
\gtrset{language=de}
% ...
\gtrSymbolsFullLegend

\gtr/symlang/Born=(text) \hspace{1cm} (no default, initially born)
Legend \text{(text)} used for \text{\(\star\).}

\gtr/symlang/Bornoutofwedlock=(text) \hspace{1cm} (no default, initially born out of wedlock)
Legend \text{(text)} used for \text{\(\star\).}

\gtr/symlang/Stillborn=(text) \hspace{1cm} (no default, initially stillborn)
Legend \text{(text)} used for \text{\(\star\star\).}

\gtr/symlang/Diedonbirthday=(text) \hspace{1cm} (no default, initially died on the birthday)
Legend \text{(text)} used for \text{\(\star\star\).}

\gtr/symlang/Baptized=(text) \hspace{1cm} (no default, initially baptized)
Legend \text{(text)} used for \text{\(\approx\).}

\gtr/symlang/Engaged=(text) \hspace{1cm} (no default, initially engaged)
Legend \text{(text)} used for \text{\(\bigcirc\).}

\gtr/symlang/Married=(text) \hspace{1cm} (no default, initially married)
Legend \text{(text)} used for \text{\(\bigcirc\bigcirc\).}

\gtr/symlang/Divorced=(text) \hspace{1cm} (no default, initially divorced)
Legend \text{(text)} used for \text{\(\bigcirc\bigcirc\).}

\gtr/symlang/Partnership=(text) \hspace{1cm} (no default, initially unmarried partnership)
Legend \text{(text)} used for \text{\(\bigcirc\bigcirc\).}

\gtr/symlang/Floruit=(text) \hspace{1cm} (no default, initially floruit)
Legend \text{(text)} used for \text{\(\approx\).}

\gtr/symlang/Died=(text) \hspace{1cm} (no default, initially died)
Legend \text{(text)} used for \text{\(\dagger\).}

\gtr/symlang/Killed=(text) \hspace{1cm} (no default, initially killed in action)
Legend \text{(text)} used for \text{\(\times\).}

\gtr/symlang/Buried=(text) \hspace{1cm} (no default, initially buried)
Legend \text{(text)} used for \text{\(\emptyset\).}

\gtr/symlang/Inurn=(text) \hspace{1cm} (no default, initially cremated)
Legend \text{(text)} used for \text{\(\emptyset\).}

\gtr/symlang/Female=(text) \hspace{1cm} (no default, initially female)
Legend \text{(text)} used for \text{\(\varphi\).}

\gtr/symlang/Male=(text) \hspace{1cm} (no default, initially male)
Legend \text{(text)} used for \text{\(\sigma\).}

\gtr/symlang/Neuter=(text) \hspace{1cm} (no default, initially neuter)
Legend \text{(text)} used for \text{\(\varphi\).}
The document `genealogytree-languages.pdf` displays the effects of language specific settings.

### 10.1 Preamble Settings

```
\documentclass{...}
%...
\gtrset{language=german-austrian}
%...
\begin{document}
%...
\end{document}
```

The current language name is stored inside \texttt{\gtrlanguagename}.

---

ootnote{\texttt{chinese} needs the \texttt{CJK} package and its \texttt{CJK} or \texttt{CJK*} environment.}
The current language is 'english'.

The /gtr/language \textsuperscript{P. 247} option sets various keys for description texts. These texts can be customized selectively, if needed.

\begin{verbatim}
gtrset{language=german} \gtrSymbolsRecordReset \gtrsymBorn\ 1775, \gtrsymDied\ 1832. \hfill(\gtrSymbolsLegend) \gtrset{symlang/Born=geb.} \gtrSymbolsRecordReset \gtrsymBorn\ 1775, \gtrsymDied\ 1832. \hfill(\gtrSymbolsLegend) \gtrset{language=english} \gtrSymbolsRecordReset \gtrsymBorn\ 1775, \gtrsymDied\ 1832. \hfill(\gtrSymbolsLegend)
\end{verbatim}

\begin{itemize}
\item 1775, \textsuperscript{†} 1832. (\textsuperscript{★}=geboren, \textsuperscript{†}=gestorben.)
\item 1775, \textsuperscript{†} 1832. (\textsuperscript{★}=geb., \textsuperscript{†}=gestorben.)
\item 1775, \textsuperscript{†} 1832. (\textsuperscript{★}=born, \textsuperscript{†}=died.)
\end{itemize}

\texttt{\gtrloadlanguage\{list of languages\}}

Loads a comma separated \texttt{\{list of languages\}}. This has to be given inside the preamble, if more than one language should be used in the document. Every loaded language can be used by /gtr/language \textsuperscript{P. 247} inside the document. For a list of feasible language names, see /gtr/language \textsuperscript{P. 247}.

\begin{verbatim}
\documentclass{...}
\%...
\gtrloadlanguage{english,german}
\%...
\begin{document}
\%...
\end{document}
\end{verbatim}

10.2 Document Settings

Switching between languages inside the document is done by setting /gtr/language \textsuperscript{P. 247}. Note that every language to be used has to be loaded inside the preamble by \gtrloadlanguage.
The library is loaded by a package option or inside the preamble by:

\gtruselibrary{debug}

This also loads the packages `array` and `tabularx` and the `breakable` library of `tcolorbox`.

## 11.1 Parser Debugging

The debugger for the parser can be used to check a manually or automatically generated tree source code to be well-formed. In this context, well-formedness means correct \LaTeX\ grouping and correct nesting with subgraph elements following the given graph grammar, see Chapter 4. It is not checked, if all mandatory graph elements are present or if too many elements are given.

Also, the debugger gives a formal structured view of the given data which is useful to search for input errors if the graphical representation fails.
Parses the given \(graph\) \(content\). If the content is well-formed, a structured list of the given data is produced. The families are automatically colored in the list. Any \(options\) are checked by setting them and they are logged in the produced list.

```python
\gtrparserdebug\{(options)\}\{(graph content)\}
```

```
parent{%
c[id=pB]{B\(child\)}%
g[id=pA]{A\(proband\)}%
c[id=pC]{C\(child\)}%
c[id=pD]{D\(child\)}%
p[id=pE]{E\(parent\)}%
p[id=pF]{F\(parent\)}%
}
```
\texttt{\textbackslash gtrparserdebuginput\{(options)\}{\{file name\}}}  

Loads the file denoted by \texttt{\{file name\}} and parses its content. If the content is well-formed, a structured list of the given data is produced. The families are automatically colored in the list. Any \texttt{(options)} are checked by setting them and they are logged in the list.

The following example uses the graph from Section 15.1 on page 357.
### 11.2 Processor Debugging

\gtrprocessordebug\{(options)\}\{(graph content)\}

Processes the given \(\text{graph content}\). If the content can be processed without error, a structured list of the processed data is produced. The families are automatically colored in the list. Any \(\text{options}\) are set for processing.

\gtrprocessordebug{
  parent{
    c[id=pB]{B\(\text{child}\)}
    g[id=pA]{A\(\text{proband}\)}
    c[id=pC]{C\(\text{child}\)}
    c[id=pD]{D\(\text{child}\)}
    p[id=pE]{E\(\text{parent}\)}
    p[id=pF]{F\(\text{parent}\)}
  }
}

---

Genealogytree Processor Debugger

#### Family 1

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>par</td>
</tr>
<tr>
<td>id</td>
<td>(none)</td>
</tr>
<tr>
<td>fam</td>
<td>(none)</td>
</tr>
<tr>
<td>offset</td>
<td>0.0pt</td>
</tr>
<tr>
<td>pos</td>
<td>7.11319pt</td>
</tr>
<tr>
<td>cwest@anchor</td>
<td>1</td>
</tr>
<tr>
<td>ceast@anchor</td>
<td>4</td>
</tr>
<tr>
<td>g</td>
<td>2</td>
</tr>
<tr>
<td>par</td>
<td>5, 6</td>
</tr>
<tr>
<td>chi</td>
<td>1, 2, 3, 4</td>
</tr>
<tr>
<td>patpar</td>
<td>5, 6</td>
</tr>
<tr>
<td>patchi</td>
<td>1, 2, 3, 4</td>
</tr>
<tr>
<td>union</td>
<td>(none)</td>
</tr>
<tr>
<td>ps</td>
<td>0pt</td>
</tr>
<tr>
<td>x</td>
<td>0.0pt</td>
</tr>
<tr>
<td>y</td>
<td>(none)</td>
</tr>
<tr>
<td>frac</td>
<td>0.5</td>
</tr>
<tr>
<td>opt@family</td>
<td>(none)</td>
</tr>
<tr>
<td>opt@subtree</td>
<td>(none)</td>
</tr>
</tbody>
</table>

**Parents of Family 1**
### Person 5

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>pE</td>
</tr>
<tr>
<td>fam</td>
<td>1</td>
</tr>
<tr>
<td>chiof</td>
<td>(none)</td>
</tr>
<tr>
<td>parof</td>
<td>1</td>
</tr>
<tr>
<td>x</td>
<td>0.0pt</td>
</tr>
<tr>
<td>y</td>
<td>113.8111pt</td>
</tr>
<tr>
<td>dim</td>
<td>71.13188pt</td>
</tr>
<tr>
<td>cwest@val</td>
<td>0.0pt</td>
</tr>
<tr>
<td>cwest@next</td>
<td>(none)</td>
</tr>
<tr>
<td>cwest@thread</td>
<td>(none)</td>
</tr>
<tr>
<td>cwest@tgap</td>
<td>(none)</td>
</tr>
<tr>
<td>ceast@val</td>
<td>71.13188pt</td>
</tr>
<tr>
<td>ceast@next</td>
<td>(none)</td>
</tr>
<tr>
<td>ceast@thread</td>
<td>(none)</td>
</tr>
<tr>
<td>ceast@tgap</td>
<td>(none)</td>
</tr>
</tbody>
</table>

**Childs of Family 1**

### Person 1

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>pB</td>
</tr>
<tr>
<td>fam</td>
<td>1</td>
</tr>
<tr>
<td>chiof</td>
<td>1</td>
</tr>
<tr>
<td>parof</td>
<td>(none)</td>
</tr>
<tr>
<td>x</td>
<td>-72.5545pt</td>
</tr>
<tr>
<td>y</td>
<td>0.0pt</td>
</tr>
<tr>
<td>dim</td>
<td>71.13188pt</td>
</tr>
<tr>
<td>cwest@val</td>
<td>-72.5545pt</td>
</tr>
<tr>
<td>cwest@next</td>
<td>5</td>
</tr>
<tr>
<td>cwest@thread</td>
<td>(none)</td>
</tr>
<tr>
<td>cwest@tgap</td>
<td>(none)</td>
</tr>
<tr>
<td>ceast@val</td>
<td>-1.42262pt</td>
</tr>
<tr>
<td>ceast@next</td>
<td>6</td>
</tr>
<tr>
<td>ceast@thread</td>
<td>(none)</td>
</tr>
<tr>
<td>ceast@tgap</td>
<td>(none)</td>
</tr>
</tbody>
</table>

---

**E (parent)**

**F (parent)**

**B (child)**
### Person 2

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>pA</td>
<td>identifier (also node alias)</td>
</tr>
<tr>
<td>fam</td>
<td>1</td>
<td>enclosing family</td>
</tr>
<tr>
<td>chiof</td>
<td>1</td>
<td>child of family</td>
</tr>
<tr>
<td>parof</td>
<td>⟨none⟩</td>
<td>parent of family</td>
</tr>
<tr>
<td>x</td>
<td>1.42264pt</td>
<td>x anchor</td>
</tr>
<tr>
<td>y</td>
<td>0.0pt</td>
<td>y anchor</td>
</tr>
<tr>
<td>dim</td>
<td>71.13188pt</td>
<td>width (or height)</td>
</tr>
<tr>
<td>cwest@val</td>
<td>1.42264pt</td>
<td>west contour value</td>
</tr>
<tr>
<td>cwest@thread</td>
<td>⟨none⟩</td>
<td>west contour thread</td>
</tr>
<tr>
<td>cwest@tgap</td>
<td>⟨none⟩</td>
<td>west contour thread gap</td>
</tr>
<tr>
<td>ceast@val</td>
<td>72.55452pt</td>
<td>east contour value</td>
</tr>
<tr>
<td>ceast@next</td>
<td>6</td>
<td>east contour successor</td>
</tr>
<tr>
<td>ceast@thread</td>
<td>⟨none⟩</td>
<td>east contour thread</td>
</tr>
<tr>
<td>ceast@tgap</td>
<td>⟨none⟩</td>
<td>east contour thread gap</td>
</tr>
</tbody>
</table>

**A (proband)**

### Person 3

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>pC</td>
<td>identifier (also node alias)</td>
</tr>
<tr>
<td>fam</td>
<td>1</td>
<td>enclosing family</td>
</tr>
<tr>
<td>chiof</td>
<td>1</td>
<td>child of family</td>
</tr>
<tr>
<td>parof</td>
<td>⟨none⟩</td>
<td>parent of family</td>
</tr>
<tr>
<td>x</td>
<td>75.39978pt</td>
<td>x anchor</td>
</tr>
<tr>
<td>y</td>
<td>0.0pt</td>
<td>y anchor</td>
</tr>
<tr>
<td>dim</td>
<td>71.13188pt</td>
<td>width (or height)</td>
</tr>
<tr>
<td>cwest@val</td>
<td>75.39978pt</td>
<td>west contour value</td>
</tr>
<tr>
<td>cwest@next</td>
<td>⟨none⟩</td>
<td>west contour successor</td>
</tr>
<tr>
<td>cwest@thread</td>
<td>⟨none⟩</td>
<td>west contour thread</td>
</tr>
<tr>
<td>cwest@tgap</td>
<td>⟨none⟩</td>
<td>west contour thread gap</td>
</tr>
<tr>
<td>ceast@val</td>
<td>146.53166pt</td>
<td>east contour value</td>
</tr>
<tr>
<td>ceast@next</td>
<td>6</td>
<td>east contour successor</td>
</tr>
<tr>
<td>ceast@thread</td>
<td>⟨none⟩</td>
<td>east contour thread</td>
</tr>
<tr>
<td>ceast@tgap</td>
<td>⟨none⟩</td>
<td>east contour thread gap</td>
</tr>
</tbody>
</table>

**C (child)**

### Person 4

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>pD</td>
<td>identifier (also node alias)</td>
</tr>
<tr>
<td>fam</td>
<td>1</td>
<td>enclosing family</td>
</tr>
<tr>
<td>chiof</td>
<td>1</td>
<td>child of family</td>
</tr>
<tr>
<td>parof</td>
<td>⟨none⟩</td>
<td>parent of family</td>
</tr>
<tr>
<td>x</td>
<td>149.37692pt</td>
<td>x anchor</td>
</tr>
<tr>
<td>y</td>
<td>0.0pt</td>
<td>y anchor</td>
</tr>
<tr>
<td>dim</td>
<td>71.13188pt</td>
<td>width (or height)</td>
</tr>
<tr>
<td>cwest@val</td>
<td>149.37692pt</td>
<td>west contour value</td>
</tr>
<tr>
<td>cwest@next</td>
<td>⟨none⟩</td>
<td>west contour successor</td>
</tr>
<tr>
<td>cwest@thread</td>
<td>⟨none⟩</td>
<td>west contour thread</td>
</tr>
<tr>
<td>cwest@tgap</td>
<td>⟨none⟩</td>
<td>west contour thread gap</td>
</tr>
<tr>
<td>ceast@val</td>
<td>220.5088pt</td>
<td>east contour value</td>
</tr>
<tr>
<td>ceast@next</td>
<td>6</td>
<td>east contour successor</td>
</tr>
<tr>
<td>ceast@thread</td>
<td>⟨none⟩</td>
<td>east contour thread</td>
</tr>
<tr>
<td>ceast@tgap</td>
<td>⟨none⟩</td>
<td>east contour thread gap</td>
</tr>
</tbody>
</table>

**D (child)**

End of Genealogytree Processor Debugger
\gtrprocessordebuginput{(options)}{(file name)}

Loads the file denoted by \texttt{(file name)} and processes its content. If the content can be processed without error, a structured list of the processed data is produced. The families are automatically colored in the list. Any \texttt{(options)} are set for processing.

The following example uses the graph from Section 15.1 on page 357.

\gtrprocessordebuginput{example.option.graph}

Genealogytree Processor Debugger

Family 1

| type: par     | type of family |
| id: SmithDoe | identifier     |
| fam: (none)  | enclosing family |
| offset: 0.0pt| x (or y) offset relative to enclosing family |
| pos: 7.11319pt| y (or x) absolute position |
| cwest@anchor: 1 | west contour starting node |
| ceast@anchor: 3 | east contour starting node |
| g: 1 | g-node of the family |
| par: 4, 7 | parent nodes |
| chi: 1, 2, 3 | child nodes |
| patpar: 4, 7 | patchwork parent nodes |
| patchi: 1, 2, 3 | patchwork child nodes |
| union: (none) | further partner families |
| ps: 0pt | pivot shift length (parents vs childs) |
| x: 0.0pt | x anchor |
| y: (none) | y anchor |
| frac: 0.5 | line positioning fraction |
| opt@family: (none) | options for the family |
| opt@subtree: (none) | options for the subtree |

Parents of Family 1

Person 4

| id: John1980 | identifier (also node alias) |
| fam: 2 | enclosing family |
| chiof: 2 | child of family |
| parof: 1 | parent of family |
| x: 38.41121pt | x anchor |
| y: 113.811pt | y anchor |
| dim: 71.13188pt | width (or height) |
| cwest@val: 38.41121pt | west contour value |
| cwest@next: 5 | west contour successor |
| cwest@thread: (none) | west contour thread |
| cwest@gap: (none) | west contour thread gap |
| ceast@val: 109.54309pt | east contour value |
| ceast@next: 6 | east contour successor |
| ceast@thread: (none) | east contour thread |
| ceast@gap: (none) | east contour thread gap |

John Smith ★ 1980
## Person 7

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>id:</td>
<td>Jane1982</td>
</tr>
<tr>
<td>fam:</td>
<td>3</td>
</tr>
<tr>
<td>chief:</td>
<td>3</td>
</tr>
<tr>
<td>parof:</td>
<td>1</td>
</tr>
<tr>
<td>x:</td>
<td>157.91275pt</td>
</tr>
<tr>
<td>y:</td>
<td>113.8111pt</td>
</tr>
<tr>
<td>dim:</td>
<td>71.13188pt</td>
</tr>
</tbody>
</table>

## Childs of Family 1

### Person 1

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>id:</td>
<td>Arth2008</td>
</tr>
<tr>
<td>fam:</td>
<td>1</td>
</tr>
<tr>
<td>chief:</td>
<td>1</td>
</tr>
<tr>
<td>parof:</td>
<td>(none)</td>
</tr>
<tr>
<td>x:</td>
<td>24.18484pt</td>
</tr>
<tr>
<td>y:</td>
<td>0.0pt</td>
</tr>
<tr>
<td>dim:</td>
<td>71.13188pt</td>
</tr>
</tbody>
</table>

### Person 2

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>id:</td>
<td>Bert2010</td>
</tr>
<tr>
<td>fam:</td>
<td>1</td>
</tr>
<tr>
<td>chief:</td>
<td>1</td>
</tr>
<tr>
<td>parof:</td>
<td>(none)</td>
</tr>
<tr>
<td>x:</td>
<td>98.16199pt</td>
</tr>
<tr>
<td>y:</td>
<td>0.0pt</td>
</tr>
<tr>
<td>dim:</td>
<td>71.13188pt</td>
</tr>
</tbody>
</table>

---

Jane Doe

★ 1982

Arthur

★ 2008

Berta

★ 2010
### Person 3

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Char2014</td>
</tr>
<tr>
<td>fam</td>
<td>1</td>
</tr>
<tr>
<td>chief</td>
<td>1</td>
</tr>
<tr>
<td>parof</td>
<td>(none)</td>
</tr>
<tr>
<td>x</td>
<td>172.13913pt</td>
</tr>
<tr>
<td>y</td>
<td>0.0pt</td>
</tr>
<tr>
<td>dim</td>
<td>71.13188pt</td>
</tr>
<tr>
<td>cwest@val</td>
<td>172.13913pt</td>
</tr>
<tr>
<td>cwest@next</td>
<td>(none)</td>
</tr>
<tr>
<td>cwest@thread</td>
<td>(none)</td>
</tr>
<tr>
<td>ceast@val</td>
<td>243.27101pt</td>
</tr>
<tr>
<td>ceast@next</td>
<td>8</td>
</tr>
<tr>
<td>ceast@thread</td>
<td>(none)</td>
</tr>
</tbody>
</table>

### Family 2

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>par</td>
</tr>
<tr>
<td>id</td>
<td>Smith</td>
</tr>
<tr>
<td>fam</td>
<td>1</td>
</tr>
<tr>
<td>offset</td>
<td>0pt</td>
</tr>
<tr>
<td>pos</td>
<td>120.9242pt</td>
</tr>
<tr>
<td>cwest@anchor</td>
<td>4</td>
</tr>
<tr>
<td>ceast@anchor</td>
<td>4</td>
</tr>
<tr>
<td>g</td>
<td>4</td>
</tr>
<tr>
<td>par</td>
<td>5, 6</td>
</tr>
<tr>
<td>chi</td>
<td>4</td>
</tr>
<tr>
<td>patpar</td>
<td>5, 6</td>
</tr>
<tr>
<td>patchi</td>
<td>4</td>
</tr>
<tr>
<td>union</td>
<td>(none)</td>
</tr>
<tr>
<td>ps</td>
<td>0pt</td>
</tr>
<tr>
<td>x</td>
<td>0.0pt</td>
</tr>
<tr>
<td>y</td>
<td>(none)</td>
</tr>
<tr>
<td>frac</td>
<td>0.5</td>
</tr>
<tr>
<td>opt@family</td>
<td>(none)</td>
</tr>
<tr>
<td>opt@subtree</td>
<td>(none)</td>
</tr>
</tbody>
</table>

### Parents of Family 2

- Family 2
- Charles
Person 5

<table>
<thead>
<tr>
<th>Id: GpSm1949</th>
<th>Identifier (also node alias)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fam: 2</td>
<td>Enclosing family</td>
</tr>
<tr>
<td>Chief: (none)</td>
<td>Child of family</td>
</tr>
<tr>
<td>Parof: 2</td>
<td>Parent of family</td>
</tr>
<tr>
<td>X: 0.0pt</td>
<td>X anchor</td>
</tr>
<tr>
<td>Y: 227.62201pt</td>
<td>Y anchor</td>
</tr>
<tr>
<td>Dim: 71.13188pt</td>
<td>Width (or height)</td>
</tr>
</tbody>
</table>

Grandpa Smith ★ 1949

Person 6

<table>
<thead>
<tr>
<th>Id: GmSm1952</th>
<th>Identifier (also node alias)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fam: 2</td>
<td>Enclosing family</td>
</tr>
<tr>
<td>Chief: (none)</td>
<td>Child of family</td>
</tr>
<tr>
<td>Parof: 2</td>
<td>Parent of family</td>
</tr>
<tr>
<td>X: 76.82242pt</td>
<td>X anchor</td>
</tr>
<tr>
<td>Y: 227.62201pt</td>
<td>Y anchor</td>
</tr>
<tr>
<td>Dim: 71.13188pt</td>
<td>Width (or height)</td>
</tr>
</tbody>
</table>

Grandma Smith ★ 1952

Childs of Family 2

Person 4

<table>
<thead>
<tr>
<th>Id: John1980</th>
<th>Identifier (also node alias)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fam: 2</td>
<td>Enclosing family</td>
</tr>
<tr>
<td>Chief: 2</td>
<td>Child of family</td>
</tr>
<tr>
<td>Parof: 1</td>
<td>Parent of family</td>
</tr>
<tr>
<td>X: 38.41121pt</td>
<td>X anchor</td>
</tr>
<tr>
<td>Y: 113.811pt</td>
<td>Y anchor</td>
</tr>
<tr>
<td>Dim: 71.13188pt</td>
<td>Width (or height)</td>
</tr>
</tbody>
</table>

John Smith ★ 1980
Family 3

type: par
id: Doe
fam: 1
offset: 156.49011pt
pos: 120.9242pt
cwest@anchor: 7
ceast@anchor: 8
g: 7
par: 9, 10
chi: 7, 8
patpar: 9, 10
patchi: 7, 8
union: (none)
ps: 0pt
x: 156.49011pt
y: (none)
frac: 0.5
opt@family: (none)
opt@subtree: (none)

Parents of Family 3

Person 9

id: GpDo1955
fam: 3
ch: (none)
parof: 3
x: 156.49011pt
y: 227.62201pt
dim: 71.13188pt
cwest@val: 0.0pt
cwest@next: (none)
cwest@thread: (none)
cwest@tgap: (none)
ceast@val: 71.13188pt
ceast@next: (none)
ceast@thread: (none)
ceast@tgap: (none)

Person 10

id: GmDo1956
fam: 3
ch: (none)
parof: 3
x: 233.31253pt
y: 227.62201pt
dim: 71.13188pt
cwest@val: 76.82242pt
cwest@next: (none)
cwest@thread: (none)
cwest@tgap: (none)
ceast@val: 147.9543pt
ceast@next: (none)
ceast@thread: (none)
ceast@tgap: (none)
### Childs of Family 3

#### Person 7
- **id:** Jane1982
- **fam:** 3
- **chiof:** 3
- **parof:** 1
  - **x:** 157.91275pt
  - **y:** 113.811pt
  - **dim:** 71.13188pt
- **cwest@val:** 1.42264pt
- **cwest@next:** 9
- **cwest@thread:** ⟨none⟩
- **ceast@val:** 72.55452pt
- **ceast@next:** 10
- **ceast@thread:** ⟨none⟩

#### Person 8
- **id:** Harr1987
- **fam:** 3
- **chiof:** 3
- **parof:** ⟨none⟩
  - **x:** 231.8899pt
  - **y:** 113.811pt
  - **dim:** 71.13188pt
- **cwest@val:** 75.39978pt
- **cwest@next:** ⟨none⟩
- **cwest@thread:** ⟨none⟩
- **ceast@val:** 146.53166pt
- **ceast@next:** 10
- **ceast@thread:** ⟨none⟩

---

*End of Genealogytree Processor Debugger*
11.3 Graphical Debugging

\texttt{\textbackslash gtrdebugdrawcontour\{\textit{options}\}\{\textit{path options}\}}

After a \texttt{\genealogytree} is drawn inside a \texttt{tikzpicture} environment, the auto-layout contour lines of a family can be displayed with this macro. For \texttt{\{options\}}, the keys \texttt{/gtr/debug/family number}, \texttt{/gtr/debug/family id}, \texttt{/gtr/debug/contour} may be used to specify the family and the contour lines to draw. The \texttt{\{path options\}} are used to draw a TikZ path.

- Contour lines for the root family should always be displayed correctly.
- Contour lines for embedded families may be displayed prolonged, because these are used to build the contour lines of their embedding families. Note that \texttt{\gtrdebugdrawcontour} shows the remains of the building process, but not the dynamics of the process.
- Contour lines for \underline{union} families are not displayed, since they are melted to their embedding \underline{child} family.

\begin{tikzpicture}
\genealogytree[template=formal graph]
{
child{
g{P_1} p{P_2} c{C_1} c{C_2}
child{
g{C_3} p{P_3} c{C_4} c{C_5} c{C_6}
}
}
}
\end{tikzpicture}
/gtr/debug/family number=\textit{(number)} \textit{(no default, initially 1)}
Selects a family by \textit{(number)} inside the option list of \texttt{\textbackslash gtrdebugdrawcontour} \textsuperscript{-P.261}.

\begin{tikzpicture}
genealogytree[template=formal graph,show family]
{ 
  child{
    g{P_1} p{P_2} c{C_1} c{C_2}
    child{
      g{C_3} p{P_3} c{C_4} c{C_5} c{C_6}
    }
  }
}
gtrdebugdrawcontour{family number=2}{draw=blue,line width=2pt}
\end{tikzpicture}

/gtr/debug/family id=\textit{(id)} \textit{(no default, initially unset)}
Selects a family by \textit{(id)} inside the option list of \texttt{\textbackslash gtrdebugdrawcontour} \textsuperscript{-P.261}.

\begin{tikzpicture}
genealogytree[template=formal graph,options for family={fam_a}{box={colback=red!50}}]
{ 
  child{
    g{P_1} p{P_2} c{C_1} c{C_2}
    child[id=fam_a]{
      g{C_3} p{P_3} c{C_4} c{C_5} c{C_6}
    }
  }
}
gtrdebugdrawcontour{family id=fam_a}{draw=blue,line width=2pt}
\end{tikzpicture}
The two contour lines are always referred to as \texttt{west} and \texttt{east} contour lines independent of the /\texttt{gtr/timeflow} setting. With this option, a partial contour drawing can be used.

\begin{tikzpicture}
\genealogytree[template=formal graph,
    options for family={fam_a}{box={colback=blue!30}},
    options for family={fam_b}{box={colback=red!30}},
]
\begin{code}
\begin{verbatim}
\begin{tikzpicture}
\genealogytree[template=formal graph,
    options for family={fam_a}{box={colback=blue!30}},
    options for family={fam_b}{box={colback=red!30}},
][
\begin{itemize}
    \item \textbf{child{ g\{P\_1\} p\{P\_2\} c\{C\_1\} c\{C\_2\} }}
    \item \textbf{child[id=fam\_a]{ g\{C\_3\} p\{P\_3\} c\{C\_4\} c\{C\_5\} c\{C\_6\} }}
    \item \textbf{child[id=fam\_b]{ g\{C\_7\} p\{P\_4\} c\{C\_8\} }}
\end{itemize}
\end{tikzpicture}
\end{verbatim}
\end{code}
\end{tikzpicture}

P\_1 P\_2
C\_1 C\_2 C\_3 P\_3 C\_4 C\_5 C\_6
C\_7 P\_4 C\_8

263
\begin{tikzpicture}
\genealogytree[template=signpost]{
\parent[id=myid]{
  c[id=pB]{B\(\text{child}\)}
  g[id=pA,box={colback=red!20!white}]{A\(\text{proband}\)}
  c[id=pC]{C\(\text{child}\)}
  c[id=pD]{D\(\text{child}\)}
\parent[id=partial,family={box={colback=red!5}}]{
  g[id=pX]{X\(\text{parent}\)}
  p(A) p(B) c(C) c(D) c(E)
}
\parent[id=partial2,family={box={colback=green!5}}]{
  g[id=pY]{Y\(\text{parent}\)}
  p(U) p(V)
}
}
\gtrdebugdrawcontour{family id=partial}{
  preaction={draw=red,line width=1mm,opacity=.5},draw=red,line width=0.4pt}
\gtrdebugdrawcontour{family id=partial2}{
  preaction={draw=green,line width=1mm,opacity=.5},draw=green,line width=0.4pt}
\gtrdebugdrawcontour{family id=myid}{
  preaction={draw=blue,line width=1mm,opacity=.5,dashed},draw=blue,line width=0.4pt}
\end{tikzpicture}
11.4 Show Information

Note that most options in this section only work, if a \texttt{/gtr/processing} based on a box from the \texttt{tcolorbox} package is chosen (this is the default setting).

\texttt{/gtr/show=\langle text \rangle} \hspace{2cm} (style, no default)

Shows a \langle text \rangle overlay for each node of the tree.

\begin{tikzpicture}
genealogytree[template=signpost,show=\langle Test \rangle]
{input{example.option.graph}}
\end{tikzpicture}

\texttt{/gtr/show id} \hspace{2cm} (style, no value)

Shows the \texttt{/gtr/id} values of every node and every family. This can be very valuable not only for debugging, but also for visual identification of nodes to manipulate.

\begin{tikzpicture}
genealogytree[template=signpost,show id]
{input{example.option.graph}}
\end{tikzpicture}
/gtr/show level

Shows the level numbers of every node. This information can be used for setting /gtr/level → P. 109 and /gtr/level n → P. 110.

\begin{tikzpicture}
genealogytree [template=signpost, show level]
{input{example.option.graph}}
\end{tikzpicture}

\begin{tikzpicture}
\genealogytree[template=signpost, show level]
{input{example.option.graph}}
\end{tikzpicture}

/gtr/show number

Shows the internal numbers of every node and every family. It is strongly recommended to reference a node by a chosen /gtr/id → P. 92 and not by its internal number, because numbers may easily change when editing the tree.

\begin{tikzpicture}
\genealogytree[template=signpost, show number]
{input{example.option.graph}}
\end{tikzpicture}

\begin{tikzpicture}
\genealogytree[template=signpost, show number]
{input{example.option.graph}}
\end{tikzpicture}
/gtr/show family

Shows the internal family numbers each node belongs to. A \textit{g} node can be part of many families, but only one family is the \textit{enclosing} family. For a \textit{union} family, the family number is displayed, but the \textit{enclosing} family is the family of the \textit{g} node.

\begin{tikzpicture}
genealogytree[template=signpost,show family]
\{input{example.option.graph}\}
\end{tikzpicture}

/gtr/show type

Show the node type for every node.

\begin{tikzpicture}
genealogytree[template=signpost,show type]
\{input{example.option.graph}\}
\end{tikzpicture}
The library is loaded by a package option or inside the preamble by:

\gtruselibrary{templates}

### 12.1 Using Templates

/\gtr/template\{(name)\} \hspace{1cm} \text{(style, no default)}

Sets a predefined style \textit{name} for a genealogytree graph. A template does not provide new functionality, but combines various options for specific trees, e.g., used inside this documentation. It serves as a shortcut. If a template is used, it is recommended to apply it as very first option.

#### 12.2 Template 'formal graph'

\texttt{template=formal\ graph}

This style is based on /\gtr/processing\textsuperscript{P.138}=\texttt{tcbox*}. The box content is set as formula in mathematical mode. For further examples, see Section 5.3 on page 83.

```latex
\begin{tikzpicture}
\genealogytree[template=formal\ graph]{
child{
  g(P_1)
  p(P_2)
  c(C_1)
  c(C_2)
  c(C_3)
}
}
\end{tikzpicture}
```
### 12.3 Template 'signpost'

```latex
\begin{tikzpicture}
genealogytree[template=signpost]{
  child{
    g{Father}
    p{Mother}
    c{Child 1}
    c{Child 2}
    c{Child 3}
  }
}\end{tikzpicture}
```

This style is based on `/gtr/processing`\textsuperscript{P.138}=`fit`. For further examples, see Section 5.2 on page 80 and many more.

### 12.4 Template 'symbol nodes'

```latex
\begin{tikzpicture}
genealogytree[template=symbol nodes]{
  child{
    gm pf cf
    child{gm pf cf cm}
    child{gm pf cm c- cm}
  }
}\end{tikzpicture}
```

This style is based on `/gtr/processing`\textsuperscript{P.138}='tcbox'. For the content, a single token \texttt{m} selects a male node (also \texttt{male}), a single token \texttt{f} selects a female node (also \texttt{female}), and every other token selects a neuter node. The symbol coloring with \texttt{\gtrSymbolsSetCreateSelected}\textsuperscript{P.238} has to be done \textit{before} entering a \texttt{tikzpicture} environment.

```latex
\gtrSymbolsSetCreateSelected{blue}{Male}
\gtrSymbolsSetCreateSelected{red}{Female}
\gtrSymbolsSetCreateSelected{black}{Neuter}
\begin{tikzpicture}
genealogytree[template=symbol nodes]{
  child{
    gm pf cf
    child{gm pf cf cm}
    child{gm pf cm c- cm}
  }
}\end{tikzpicture}
```
12.5 Template 'tiny boxes'

This style is based on /gtr/processing P.138 = tcbox*. The content of all boxes is removed. Therefore, a single token like ‘-’ is enough to declare the content. For further examples, see Chapter 14 on page 347.

\begin{tikzpicture}
\genealogytree[template=tiny boxes]{
child{
  g-p-c-
  child{g-p-c-c-}
  child{g-p-c-c-c-}
}
}
\end{tikzpicture}

12.6 Template 'tiny circles'

This style is based on /gtr/processing P.138 = tcbox*. The content of all boxes is removed. Therefore, a single token like ‘-’ is enough to declare the content. All distances are set equally and edges are drawn meshed. For further examples, see Chapter 14 on page 347.

\begin{tikzpicture}
\genealogytree[template=tiny circles]{
child{
  g-c-
  child{g-c-c-}
  child{g-c-c-c-}
}
}
\end{tikzpicture}
This style is based on \texttt{/gtr/processing}\textsuperscript{P.138}=\texttt{tcbox*} and sets \texttt{/gtr/timeflow}\textsuperscript{P.80}=\texttt{right}. Note that optimal level sizes have to be set manually.

\begin{genealogypicture}\[template=directory, \\
level 0/.style={level size=11mm}, \\
level -1/.style={level size=11mm}, \\
level -2/.style={level size=15mm}, \\
level -3/.style={level size=31mm}, \\
level -4/.style={level size=62mm}, \\
]\child{ g{tds} \child[subtree box={colback=green!20}]{ g{doc} \child{ g{latex} \child{ g{genealogytree} \c{genealogytree.pdf} \c{genealogytree.doc.sources.zip} \c{README} } } } \child[subtree box={colback=red!15}]{ g{tex} \child{ g{latex} \child{ g{genealogytree} \c{genealogytree.sty} \c{gtrcore.*.code.tex} \c{gtrlang.*.code.tex} \c{gtrlib.*.code.tex} } } } \end{genealogypicture}
12.8 Template 'database pole'

This style is based on /gtr/processing→P.138=database and sets /gtr/database format→P.174=full marriage above.

The boxes are quite small for placing many nodes horizontally. Also, many settings are adapted for this style.

The following example uses a file documented in Section 15.2 on page 358.

The next example uses the graph data from Section 2.3.5 on page 39.
12.9 Template 'database pole reduced'

This style is identical to database pole (Section 12.8 on page 273), but every size property is scaled by factor $\frac{1}{10}$. Therefore, the resulting graphs are only 10 percent of width and height of corresponding graphs made with database pole.

- \TeX/\LaTeX length values are limited by about 575\,cm. Due to internal calculations, the maximum width and height of a graph may even be smaller. Using this reduced layout size, this limit is avoided to a certain degree.
- For virtual PDF 'paper' on a computer screen, the tiny layout irrelevant, since this vector format can be zoomed without loss.
- For printing, the PDF also can be zoomed or cut into several pages which can be zoomed.
- Note that a freely scalable text font is needed for the reduced layout size! The standard \LaTeX font is not scalable.
- Examples with more than 1500 nodes compiled successfully. Note that the compiler memory settings may have to be set to increased values for graphs with many nodes.

The following example uses the graph data from Section 2.3.5 on page 39.
12.10 Template 'database poleportrait'

This style is based on /gtr/processing P.138 database and sets /gtr/database format P.174 full marriage above.

In contrast to database pole (Section 12.8 on page 273), portraits are drawn, if present.

\begin{genealogypicture}
[template=database poleportrait]
input{example.database.graph}
\end{genealogypicture}
The next example uses the graph data from Section 2.3.5 on page 39.
12.11 Template 'database poleportrait reduced'

This style is identical to database poleportrait (Section 12.10 on page 276), but every size property is scaled by factor \( \frac{1}{10} \). See Section 12.9 on page 275 for more explanations.

The following example uses the graph data from Section 2.3.5 on page 39.

\begin{tikzpicture}
    \spy on (0.6,-0.1) in node [left] at (7,1);
    \spy using outlines={circle, magnification=10, size=5cm, connect spies}
\end{tikzpicture}
12.12 Template 'database portrait'

This style is based on /gtr/processing=P.138=database and sets /gtr/database format=P.174=short no marriage.

The boxes are quite small for placing many nodes horizontally. Also, many settings are adapted for this style. If /gtr/database/image=P.166 is present, the corresponding image is inserted. Otherwise, a symbolic portrait is drawn.

\begin{genealogypicture}
[template=database portrait]
child{
  g{male,name=Hans Test,
    birth={1520-02-17}{Footown},
    death={1588-12-12}{Footown}}
  p{female,name=Berta,
    death={1592-03-02}{Footown}}
  c{name=Unknown,
    birth+={1553-01-12}{Footown}
    {stillborn}}
}
\end{genealogypicture}

The next example uses the graph data from Section 2.3.5 on page 39.
12.13  Template 'database portrait reduced'

This style is identical to database portrait (Section 12.12 on page 279), but every size property is scaled by factor $\frac{1}{10}$. See Section 12.9 on page 275 for more explanations.

The following example uses the graph data from Section 2.3.5 on page 39.

\begin{tikzpicture}
    \spy using outlines={circle, magnification=10, size=5cm, connect spies}
    \genealogytreeinput[template=database portrait reduced]{example.gauss.graph}
    \spy on (0.6,-0.1) in node [left] at (7,1);
\end{tikzpicture}
12.14 Template 'database traditional'

This style is based on /gtr/processing =database, sets /gtr/database format =short no marriage and /gtr/timeflow =down.

Using this template, a sober black-and-white drawing with only short information is created. The box content is not framed.

- For p nodes, the content is bottom aligned.
- For c nodes, the content is top aligned.
- For g nodes, the content is center aligned. While this is usually reasonable, g nodes in families without children or parents may have to be adapted manually. The root node is treated automatically.

The following example uses a file documented in Section 15.2 on page 358.

```latex
\begin{genealogypicture}[
  template=database traditional,
  level size=1.3cm
]
  input{example.database.graph}
\end{genealogypicture}
```

The next example uses the graph data from Section 2.3.5 on page 39.

```latex
\begin{genealogypicture}[
  template=database traditional,
  level size=1.2cm,node size=2.2cm,date format=yyyy,list separators={\par}{ }{}{}
]
  input{example.gauss.graph}
\end{genealogypicture}
```
This style is identical to database traditional (Section 12.14 on page 282), but every size property is scaled by factor $\frac{1}{10}$. See Section 12.9 on page 275 for more explanations.

The following example uses the graph data from Section 2.3.5 on page 39.
12.16 Template 'database sideways'

This style is based on /gtr/processing\textsuperscript{P.138}=database, sets /gtr/database format\textsuperscript{P.174}=medium and /gtr/timeflow\textsuperscript{P.80}=left.

Here, the boxes are positioned sideways and have a large variety of height. Therefore, the content will seldom be resized.

The following example uses a file documented in Section 15.2 on page 358.

\begin{genealogypicture}[template=database sideways]
\input{example.database.graph}
\end{genealogypicture}

Charles Smith
ca. 1722 in London
October 12, 1764
Copper smith, soldier. Invented the square wheel.

Jane Bowden
March 2, 1742 in London
1724 to 1802

Abraham Bowden
\checkmark January 4, 1740 in London
\checkmark February 23, 1740 in London

Elizabeth "Liz" Smith
\star February 2, 1744 in London
\checkmark April 12, 1812 in Boston
Had a store in Boston.

Michael Smith
\march March 1, 1758

The next example uses the graph data from Section 2.3.5 on page 39.

\begin{genealogypicture}[template=database sideways,
level size=3.2cm,ignore level=3,database format=name]
\input{example.gauss.graph}
\end{genealogypicture}

Karoline
\march 1790
\march 1812

Johanna Elisabeth Rosina Osthoff
Johann Carl Friedrich Gauß
Friederica Wilhelmine Waldeck

Gebhard Dietrich Gauß

Dorothea Benze

Jürgen Gooss
Katharina Magdalena Eggenlings
Christoph Bentze
Katharina Krone

284
12.17 Template 'database sideways reduced'

This style is identical to database sideways (Section 12.16 on page 284), but every size property is scaled by factor $\frac{1}{10}$. See Section 12.9 on page 275 for more explanations.

The following example uses the graph data from Section 2.3.5 on page 39.

\begin{tikzpicture}
   \spy using outlines={circle, magnification=10, size=5cm, connect spies}
   \genealogytreeinput{template=database sideways reduced}{example.gauss.graph}
   \spy on (0.6,-0.5) in node [left] at (7,1);
\end{tikzpicture}
12.18 Template 'database sidewaysportrait'

This style is based on /gtr/processing → P. 138 database, sets /gtr/database format → P. 174 medium and /gtr/timeflow → P. 80 left.

In contrast to database sideways (Section 12.16 on page 284), portraits are drawn, if present.

The next example uses the graph data from Section 2.3.5 on page 39.
12.19 Template 'database sidewaysportrait reduced'

This style is identical to database sidewaysportrait (Section 12.18 on page 287), but every size property is scaled by factor \( \frac{1}{10} \). See Section 12.9 on page 275 for more explanations.

The following example uses the graph data from Section 2.3.5 on page 39.

\begin{tikzpicture}
\spy using outlines={circle, magnification=10, size=5cm, connect spies} on (-0.2,-0.5) in node [left] at (7,1);
\genealogytreeinput[template=database sidewaysportrait reduced]{example.gauss.graph}
\spy on (-0.2,-0.5) in node [left] at (7,1);
\end{tikzpicture}
This template is intended to be used for diagrams which show the relationship of a person X to a person Y with common ancestors. If /gtr/database/image is present, the corresponding image is inserted.
12.21 Template 'ahnentafel 3'

This style is based on /gtr/processing and sets /gtr/timeflow = left.

Note that this style is very restrictive and its sole intended use is to easily set up predefined ancestor tables with three generations of ancestors. One should apply only parent, p, and g constructs which gives a binary tree.

```latex
\begin{genealogypicture}{template=ahnentafel 3,empty name text={},
   autofill parents male female=3,
   date format=d mon yyyy}
\begin{align*}
\text{parent} & \{ \\
\text{g} (\text{male}, \text{name} = \text{pref}\{\text{Frederik}\} \text{\ surn}\{\text{Smith}\}, \\
\text{birth} = \{1900-01-01\} \{\text{New York}\}, \text{death} = \{1970-01-01\} \{\text{New York}\}, \\
\text{marriage} = \{1929-01-01\} \{\text{New York}\}, \text{comment} = \{\text{Used Cars Salesman}\})
\}
\text{parent} & \{ \\
\text{g} (\text{male}, \text{name} = \text{pref}\{\text{Ernest}\} \text{\ surn}\{\text{Smith}\}, \\
\text{birth} = \{1870-02-02\} \{\text{London}\}, \text{death} = \{1940-02-02\} \{\text{London}\}, \\
\text{marriage} = \{1899-02-02\} \{\text{London}\}, \text{comment} = \{\text{Milkman}\})
\}
\text{parent} & \{ \\
\text{g} (\text{male}, \text{name} = \text{pref}\{\text{Dominik}\} \text{\ surn}\{\text{Schmidt}\}, \\
\text{birth} = \{1840-03-03\} \{\text{Berlin}\}, \text{death} = \{1910-03-03\} \{\text{London}\}, \\
\text{marriage} = \{1869-03-03\} \{\text{Berlin}\}, \text{comment} = \{\text{Baker}\})
\}
\text{p} (\text{male}, \text{name} = \text{pref}\{\text{Christian}\} \text{\ surn}\{\text{Schmied}\}, \\
\text{birth} = \{1810-04-04\} \{\text{Vienna}\}, \text{death} = \{1870-04-04\} \{\text{Vienna}\}, \\
\text{marriage} = \{1839-04-04\} \{\text{Vienna}\}, \text{comment} = \{\text{Blacksmith}\})
\}
\end{align*}
\end{genealogypicture}
```
Frederik Smith
★ 1 Jan 1900 in New York
★★ 1 Jan 1929 in New York
★★★ 1 Jan 1970 in New York
Used Cars Salesman.

Ernest Smith
★ 2 Feb 1870 in London
★★ 2 Feb 1899 in London
★★★ 2 Feb 1940 in London
Milkman.

Dominik Schmidt
★ 3 Mar 1840 in Berlin
★★ 3 Mar 1869 in Berlin
★★★ 3 Mar 1910 in London
Baker.

Christian Schmied
★ 4 Apr 1810 in Vienna
★★ 4 Apr 1839 in Vienna
★★★ 4 Apr 1870 in Vienna
Blacksmith.
12.22  Template 'ahnentafel 4'

This style is based on \texttt{/gtr/processing} -database and sets \texttt{/gtr/timeflow} -left.

Note that this style is very restrictive and its sole intended use is to easily set up predefined ancestor tables with four generations of ancestors. One should apply only parent, p, and g constructs which gives a binary tree. Since the first parent generation is shifted, the diagram should always contain mother and father of the proband to avoid overlapping.

\begin{genealogypicture} \template=ahnentafel 4,empty name text={},
autofill parents male female=4,
date format=d mon yyyy
\end{genealogypicture}
Frederik Smith
1 Jan 1900 in New York
1 Jan 1929 in New York
1 Jan 1970 in New York
Used Cars Salesman.

Ernest Smith
* 2 Feb 1870 in London
&o 2 Feb 1899 in London
† 2 Feb 1940 in London Milkman.

Dominik Schmidt
* 3 Mar 1840 in Berlin
&o 3 Mar 1869 in Berlin
† 3 Mar 1910 in London Baker.

Christian Schmied
* 4 Apr 1810 in Vienna
&o 4 Apr 1839 in Vienna
† 4 Apr 1870 in Vienna Blacksmith.

Bartholomäus Schmid
* 5 May 1780 in Eger
&o 5 May 1809 in Eger
† 5 May 1840 in Eger Blacksmith.
This style is based on `/gtr/processing` database and sets `/gtr/timeflow` = left.

Note that this style is very restrictive and its sole intended use is to easily set up predefined ancestor tables with five generations of ancestors. One should apply only `parent`, `p`, and `g` constructs which gives a binary tree. Since the first parent generation is shifted, the diagram should always contain mother and father of the proband to avoid overlapping.

```latex
\begin{genealogypicture}[template=ahnentafel 5,empty name text={},
autofill parents male female=5,
date format=d mon yyyy]
  parent{
    g{male,name=\textit{Frederik} \textit{Smith},
    marriage={1929-01-01}{New York},comment={Used Cars Salesman}}
  }
  parent{
    g{male,name=\textit{Ernest} \textit{Smith},
    birth={1870-02-02}{London},death={1940-02-02}{London},
    marriage={1899-02-02}{London},comment={Milkman}}
  }
  parent{
    g{male,name=\textit{Dominik} \textit{Schmidt},
    birth={1840-03-03}{Berlin},death={1910-03-03}{London},
    marriage={1869-03-03}{Berlin},comment={Baker}}
  }
  parent{
    g{male,name=\textit{Christian} \textit{Schmied},
    birth={1810-04-04}{Vienna},death={1870-04-04}{Vienna},
    marriage={1839-04-04}{Vienna},comment={Blacksmith}}
  }
  parent{
    g{male,name=\textit{Bartholom"aus} \textit{Schmid},
    birth={1780-05-05}{Eger},death={1840-05-05}{Eger},
    marriage={1809-05-05}{Eger},comment={Blacksmith}}
  }
  p{male,name=\textit{Abraham} \textit{Schmid},
    birth={1750-06-06}{St. Joachimsthal},death={1810-06-06}{Eger},
    marriage={1779-06-06}{Eger},comment={Miner}}
\end{genealogypicture}
```
Frederik Smith  
* 1 Jan 1900 in New York  
** 1 Jan 1929 in New York  
† 1 Jan 1970 in New York  
Used Cars Salesman.

Ernest Smith  
* 2 Feb 1870 in London  
** 2 Feb 1899 in London  
† 2 Feb 1940 in London  
Milkman.

Dominik Schmidt  
* 3 Mar 1840 in Berlin  
** 3 Mar 1869 in Berlin  
† 3 Mar 1910 in London  
Baker.

Christian Schmied  
* 4 Apr 1810 in Vienna  
** 4 Apr 1839 in Vienna  
† 4 Apr 1870 in Vienna  
Blacksmith.

Bartholomäus Schmid  
* 5 May 1780 in Eger  
** 5 May 1809 in Eger  
† 5 May 1840 in Eger  
Blacksmith.

Abraham Schmid,  
* 6 Jun 1750 in St. Joachimsthal  
** 6 Jun 1779 in Eger  
† 6 Jun 1810 in Eger  
Blacksmith.
This style is based on /gtr/processing→P.138 database and sets /gtr/database format→P.174 full marriage below.

Note that this style is very restrictive and its sole intended use is to easily set up predefined ancestor tables with three generations of ancestors to be printed on landscape A4 paper. One should apply only parent, p, and g constructs which gives a binary tree.

The graph is automatically filled to three generations by /gtr/autofill parents male female→P.133 and also pruned to three generations. The total size is 283 mm:196 mm with intended use to be centered on a 297 mm:210 mm paper (border adaption see Section 12.35 on page 304). Typically, the content should be readable when printed on A4 sized paper.
12.25 Template 'a4paper 4'

This style is based on /gtr/processing and sets /gtr/database format = full marriage below.

Note that this style is very restrictive and its sole intended use is to easily set up predefined ancestor tables with four generations of ancestors to be printed on landscape A4 paper. One should apply only parent, p, and g constructs which gives a binary tree.

The graph is automatically filled to four generations by /gtr/autofill parents male female and also pruned to four generations. The total size is 283 mm:196 mm with intended use to be centered on a 297 mm:210 mm paper (border adaption see Section 12.35 on page 304). Typically, the content should be readable when printed on A4 sized paper.
12.26 Template 'a3paper 5'

This style is based on \gtr/processing\textsuperscript{P.138} database and sets \gtr/database format\textsuperscript{P.174} full marriage below.

Note that this style is very restrictive and its sole intended use is to easily set up predefined ancestor tables with five generations of ancestors to be printed on landscape A3 paper. One should apply only parent, p, and g constructs which gives a binary tree.

The graph is automatically filled to five generations by \gtr/autofill parents male female\textsuperscript{P.133} and also pruned to five generations. The total size is 406 mm : 283 mm with intended use to be centered on a 420 mm : 297 mm paper (border adaption see Section 12.35 on page 304). Typically, the content should be readable when printed on A3 sized paper.
12.27 Template 'letterpaper 3'

This style is based on /gtr/processing\footnote{P.138} \texttt{database} and sets \texttt{/gtr/database format} \footnote{P.174} \texttt{full marriage below}.

Note that this style is very restrictive and its sole intended use is to easily set up predefined ancestor tables with three generations of ancestors to be printed on landscape letter paper. One should apply only \texttt{parent}, \texttt{p}, and \texttt{g} constructs which gives a binary tree.

The graph is automatically filled to three generations by /gtr/autofill parents male female* \footnote{P.133} and also pruned to three generations. The total size is 10.5 in : 8 in with intended use to be centered on a 11 in : 8.5 in paper (border adaption see Section 12.35 on page 304). Typically, the content should be readable when printed on letter sized paper.

12.28 Template 'letterpaper 4'

This style is based on /gtr/processing\footnote{P.138} \texttt{database} and sets \texttt{/gtr/database format} \footnote{P.174} \texttt{full marriage below}.

Note that this style is very restrictive and its sole intended use is to easily set up predefined ancestor tables with four generations of ancestors to be printed on landscape A4 paper. One should apply only \texttt{parent}, \texttt{p}, and \texttt{g} constructs which gives a binary tree.

The graph is automatically filled to four generations by /gtr/autofill parents male female* \footnote{P.133} and also pruned to four generations. The total size is 10.5 in : 8 in with intended use to be centered on a 11 in : 8.5 in paper (border adaption see Section 12.35 on page 304). Typically, the content should be readable when printed on letter sized paper.
This style is based on /gtr/processing and sets /gtr/database format and also pruned to three generations. The total size is 316 mm:176 mm with intended use to be centered on a 320 mm:180 mm paper (border adaption see Section 12.35 on page 304). Typically, the content should be readable on a Full HD screen with 1920×1080 pixels.
This style is based on \texttt{/gtr/processing \textsuperscript{P.138} database} and sets
\texttt{/gtr/database format \textsuperscript{P.174} full marriage below.}

Note that this style is very restrictive and its sole intended use is to easily set up predefined ancestor tables with four generations of ancestors to be shown on a 16:9 display. One should apply only parent, p, and g constructs which gives a binary tree.

The graph is automatically filled to four generations by \texttt{/gtr/autofill parents male female \textsuperscript{P.133}} and also pruned to four generations. The total size is 316 mm : 176 mm with intended use to be centered on a 320 mm : 180 mm paper (border adaption see Section 12.35 on page 304). Typically, the content should be readable on a Full HD screen with 1920×1080 pixels.
This style is based on \gtr/processing\textsuperscript{P.138} database and sets \gtr/database format\textsuperscript{P.174} full marriage below.

Note that this style is very restrictive and its sole intended use is to easily set up predefined ancestor tables with five generations of ancestors to be shown on a 16:9 display. One should apply only parent, p, and g constructs which gives a binary tree.

The graph is automatically filled to five generations by \gtr/autofill parents male female\textsuperscript{P.133} and also pruned to five generations. The total size is 316 mm : 176 mm with intended use to be centered on a 320 mm : 180 mm paper (border adaption see Section 12.35 on page 304). Typically, the content should be readable on a UHD screen with 3840 × 2180 pixels (4K).

\begin{document}
\begin{tcolorbox}[blanker,spread,halign=center,valign=center]
\begin{document}
\pagestyle{empty}
\begin{tikzpicture}
\begin{tcolorbox}
\begin{document}
\begin{documentclass}{article}
\usepackage{notocondensed}
\usepackage[ngerman]{babel}
\usepackage{notocondensed}
\usepackage[paperwidth=320mm,paperheight=180mm]{geometry}
\begin{tikzpicture}
\begin{tcolorbox}
\begin{document}
\begin{example.neumann.graph}
\end{document}
\end{tcolorbox}
\end{tikzpicture}
\end{document}
\end{document}
\end{tikzpicture}
\end{document}
\end{tcolorbox}
\end{document}
12.32 Template 'display 16:10 3’

This style is based on /gtr/processing+P.138=database and sets /gtr/database format+P.174=full marriage below.

Note that this style is very restrictive and its sole intended use is to easily set up predefined ancestor tables with three generations of ancestors to be shown on a 16:10 display. One should apply only parent, p, and g constructs which gives a binary tree.

The graph is automatically filled to three generations by /gtr/autofill parents male female*+P.133 and also pruned to three generations. The total size is 316 mm : 196 mm with intended use to be centered on a 320 mm : 200 mm paper (border adaption see Section 12.35 on page 304). Typically, the content should be readable on a WUXGA screen with 1920×1200 pixels.

12.33 Template 'display 16:10 4’

This style is based on /gtr/processing+P.138=database and sets /gtr/database format+P.174=full marriage below.

Note that this style is very restrictive and its sole intended use is to easily set up predefined ancestor tables with four generations of ancestors to be shown on a 16:9 display. One should apply only parent, p, and g constructs which gives a binary tree.

The graph is automatically filled to four generations by /gtr/autofill parents male female*+P.133 and also pruned to four generations. The total size is 316 mm : 196 mm with intended use to be centered on a 320 mm : 200 mm paper (border adaption see Section 12.35 on page 304). Typically, the content should be readable on a WUXGA screen with 1920×1200 pixels.

12.34 Template 'display 16:10 5’

This style is based on /gtr/processing+P.138=database and sets /gtr/database format+P.174=full marriage below.

Note that this style is very restrictive and its sole intended use is to easily set up predefined ancestor tables with five generations of ancestors to be shown on a 16:10 display. One should apply only parent, p, and g constructs which gives a binary tree.

The graph is automatically filled to five generations by /gtr/autofill parents male female*+P.133 and also pruned to five generations. The total size is 316 mm : 196 mm with intended use to be centered on a 320 mm : 200 mm paper (border adaption see Section 12.35 on page 304). Typically, the content should be readable on a QWUXGA screen with 3840×2400 pixels.
12.35 Border Size Options

The following border size options can be used to adapt the templates which are listed below. Setting `/gtr/template horizontal border` and `/gtr/template vertical border` to 0 mm means that, e.g. `a4paper 4` fills the complete area of an A4 sized paper. The options also accept negative lengths which leads to a magnification.

### /gtr/template horizontal border=

<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>a4paper 3</td>
<td>7 mm</td>
</tr>
<tr>
<td>a4paper 4</td>
<td>7 mm</td>
</tr>
<tr>
<td>a3paper 5</td>
<td>7 mm</td>
</tr>
<tr>
<td>letterpaper 3</td>
<td>0.25 in</td>
</tr>
<tr>
<td>letterpaper 4</td>
<td>0.25 in</td>
</tr>
<tr>
<td>display 16:9 3</td>
<td>2 mm</td>
</tr>
<tr>
<td>display 16:9 4</td>
<td>2 mm</td>
</tr>
<tr>
<td>display 16:9 5</td>
<td>2 mm</td>
</tr>
<tr>
<td>display 16:10 3</td>
<td>2 mm</td>
</tr>
<tr>
<td>display 16:10 4</td>
<td>2 mm</td>
</tr>
<tr>
<td>display 16:10 5</td>
<td>2 mm</td>
</tr>
</tbody>
</table>

### /gtr/template vertical border=

<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>a4paper 3</td>
<td>7 mm</td>
</tr>
<tr>
<td>a4paper 4</td>
<td>7 mm</td>
</tr>
<tr>
<td>a3paper 5</td>
<td>7 mm</td>
</tr>
<tr>
<td>letterpaper 3</td>
<td>0.25 in</td>
</tr>
<tr>
<td>letterpaper 4</td>
<td>0.25 in</td>
</tr>
<tr>
<td>display 16:9 3</td>
<td>2 mm</td>
</tr>
<tr>
<td>display 16:9 4</td>
<td>2 mm</td>
</tr>
<tr>
<td>display 16:9 5</td>
<td>2 mm</td>
</tr>
<tr>
<td>display 16:10 3</td>
<td>2 mm</td>
</tr>
<tr>
<td>display 16:10 4</td>
<td>2 mm</td>
</tr>
<tr>
<td>display 16:10 5</td>
<td>2 mm</td>
</tr>
</tbody>
</table>
Johann Balthasar Neumann
27.1.1687 in Eger (Böhmen)
19.8.1753 in Würzburg (Unterfranken)
Baumeister des Barock und des Rokoko.

Johann Christoph Neumann
1631 in Eger (Böhmen)
8.8.1713 in Eger (Böhmen)
Bürger und Tuchmacher in Eger.

Andreas Neumann
13.11.1605 in Eger (Böhmen)
9.12.1671 in Eger (Böhmen)
Bürger und Tuchmacher in Eger.

Andreas Neumann
23.9.1576 in Eger (Böhmen)
ca. 1640 in Eger (Böhmen)
Bürger und Tuchmacher in Eger.

Barbara Keßler
ca. 1570 in Eger (Böhmen)
ca. 1650 in Eger (Böhmen)
Bürgerstochter aus Eger.

Ester Hoyer
ca. 1600 in Sebenbach bei Eger (Böhmen)
ca. 1670 in Eger (Böhmen)
Bauerstochter aus Schönbach.

Lorenz Hoyer
ca. 1560 ca. 1640 in Sebenbach bei Eger (Böhmen)
Bauer in Schönbach.

Lorenz Grassoldt
ca. 1550 ca. 1630 in Eger (Böhmen)
Bürger und Tuchmacher in Eger.

Rosina Grassoldt
29.11.1651 in Eger (Böhmen)
26.11.1699 in Eger (Böhmen)
Tuchmacherstochter aus Eger.

Hieronimus Grassoldt
ca. 1600 in Eger (Böhmen)
ca. 1690 in Eger (Böhmen)
Bürger und Tuchmacher in Eger.

Lorenz Grassoldt
ca. 1550 ca. 1630 in Eger (Böhmen)
Bürger und Tuchmacher in Eger.

Barbara Michel
3.11.1571 in Eger (Böhmen)
ca. 1630 in Eger (Böhmen)
25.10.1587 in Eger (Böhmen).

Margaretha Hammerschmid
8.3.1623 in Eger (Böhmen)
ca. 1690 in Eger (Böhmen)
Tuchmacherstochter aus Eger.

Katharina Dietl
20.12.1587 in Eger (Böhmen)
ca. 1660 in Eger (Böhmen)
Weißgerberstochter aus Eger.

Nikolaus Hammerschmid
ca. 1570 in Schöba (Böhmen)
1623 in Eger (Böhmen)
Bürger und Tuchmacher zu Eger.

Lorenz Grassoldt
ca. 1550 ca. 1630 in Eger (Böhmen)
Bürger und Tuchmacher in Eger.

Barbara Michel
3.11.1571 in Eger (Böhmen)
ca. 1630 in Eger (Böhmen)
25.10.1587 in Eger (Böhmen)
12.36 Predefined Colors of the Library

The following colors are predefined. They are used as default colors in some templates.
12.37 Auxiliary Control Sequences

Also see Section 5.15 on page 132 for adding ancestors automatically.

\gtrparent1

This control sequence inserts a pair of parents with content male and female.

\begin{genealogypicture}
[template=symbol nodes]
parent{
g{male}
insert{gtrparent1}
}
\end{genealogypicture}

\gtrparent2

This control sequence inserts two generations of parents with content male and female.

\begin{genealogypicture}
[template=symbol nodes]
parent{
g{male}
insert{gtrparent2}
}
\end{genealogypicture}

\gtrparent3

This control sequence inserts three generations of parents with content male and female.

\begin{genealogypicture}
[template=symbol nodes]
parent{
g{male}
insert{gtrparent3}
}
\end{genealogypicture}
\gtrparent4
This control sequence inserts four generations of parents with content \textit{male} and \textit{female}.

\begin{genealogypicture}
\[template=symbol nodes\]
parent{
\(g\{male\}\)
insert{\gtrparent4}
}
\end{genealogypicture}

\gtrparent5
This control sequence inserts five generations of parents with content \textit{male} and \textit{female}.

\gtrparent6
This control sequence inserts six generations of parents with content \textit{male} and \textit{female}.

\gtrparent7
This control sequence inserts seven generations of parents with content \textit{male} and \textit{female}.

\gtrDrawSymbolicPortrait
Inserts Ti\kZ code to draw a symbolic portrait. The colors are frame and back color of a \texttt{tcolorbox}. Therefore, the intended application is inside a \texttt{tcolorbox}.

\begin{tcolorbox}[enhanced,width=2.5cm,height=4cm, title=Test,halign title=center, colframe=green!25!black,colback=yellow!50, underlay={\begin{tcbclipinterior}\% \begin{genealogypicture}\[template=symbol nodes\]parent{\(g\{male\}\)insert{\gtrparent4}} \end{genealogypicture}\} \]
\end{tcolorbox}

Test
The library is loaded by a package option or inside the preamble by:

\gtruselibrary{fanchart}

### 13.1 Fan Chart Diagrams

A *fan chart* is a specialized diagram which differs in many ways from all other graphs of the package. Especially, the autolayout algorithm is replaced by another fixed layout algorithm. Thus, most geometry related options are also replaced by special ones for the fan chart.

- Only **parent** trees without children can by used as *(tree contents)* for a fan chart. Nevertheless, children *(c)* may be present inside the graph, but are not displayed.
- The nodes are displayed as segments of a circle, but the node text still is set inside a rectangular (invisible) box which is fitted to a segment of circle. The box dimensions are computed automatically to inscribe the box with maximum area. See Section 13.2 on page 312 for segment dimensions.
- Node data processing is limited to database processing, see Chapter 7 on page 161. Other processings are not supported.
- All customization options for database nodes are feasible. Only the node background settings are ignored and are replaced by segment coloring, see Section 13.4 on page 320.
- Templates from the \texttt{templates} library are not applicable for obvious reasons, but there are some fan chart templates instead, see Section 13.5 on page 333.
\texttt{\textbackslash{}gtrfanchart[(options)]\{tree contents\}}

Draws a fan chart diagram according to the given \textit{tree contents} controlled by the given \textit{options}.

Note that \textit{tree contents} has to be a parent tree (without children). \texttt{\textbackslash{}gtrfanchart} corresponds to \texttt{\textbackslash{}genealogytree} \textsuperscript{P.55} and has to be placed into a \texttt{tikzpicture} environment or a derived environment like \texttt{autosizetikzpicture} \textsuperscript{P.61}.

Database processing for the nodes is present.

\begin{autosizetikzpicture}{\linewidth}{\}
\texttt{\textbackslash{}gtrfanchart[}
\texttt{\textbackslash{}autofill parents male female=3,}
\texttt{\textbackslash{}fanchart template=malefemale sober,}
\texttt{]}
\texttt{\{}
\texttt{\textbackslash{}parent{}
\texttt{\textbackslash{}g{male,name=Proband}}
\texttt{\textbackslash{}parent{}
\texttt{\textbackslash{}g{male,name=Father}}
\texttt{\textbackslash{}p{male,name=Grandfather}}
\texttt{\textbackslash{}p{female,name=Grandmother}}
\texttt{}
\texttt{\textbackslash{}p{female,name=Mother}}
\texttt{}}
\texttt{}}
\texttt{\end{autosizetikzpicture}
Steps for setting up a fan chart:

1. Decide on the number \( n \) of displayed generations (levels). In theory, the library supports up to 24 generations which means 16777216 segments on the outermost ring. In reality, \TeX{} memory limitations will force a distinct smaller number \( n \).

2. Always use \texttt{/gtr/autofill parents male female=} \( n \) or \texttt{/gtr/autofill parents female male=} \( n \) to auto fill and restrict the number of generations to the desired level \( n \), if your graph is not without gap. Expect deviations in placing male/female nodes otherwise. If visible gaps are preferred, an added \texttt{/gtr/complemented phantom} \( n \) does the trick.

3. If a \texttt{/gtr/fanchart template} is to be used, it should be applied before other customization options, because a template sets a lot of options itself.

4. Adapt the geometry of the fan chart to your liking, see Section 13.2 on page 312. This also determines the size for the node text.

5. Possibly adapt the \texttt{/gtr/database format} in dependency of the \texttt{/gtr/level}, if outer nodes should contain more concise information than inner nodes.
A (full) segment is shown inside a red frame above. It decomposes into a core segment, an optional marker area and an optional gap. The actual node box containing the node text is inscribed into the (core) segment.

$\text{/gtr/fanchart radii}=${\langle radius1 \rangle, \langle radius2 \rangle, \ldots} \quad \text{(initially 2.5cm, 3.5cm, 3cm)}$

Defines the radii of the fan chart rings as comma separated list of length values \langle radius1 \rangle, \langle radius2 \rangle, \ldots The last value of this list is taken for all following radii.

$\text{/gtr/fanchart inner offset}=${\langle offset1 \rangle, \langle offset2 \rangle, \ldots} \quad \text{(no default, initially 0mm)}$

Defines the inner offset values as comma separated list of length values \langle offset1 \rangle, \langle offset2 \rangle, \ldots The last value of this list is taken for all following inner offsets. For standard fan charts, the inner offset describes the gap between two rings.

$\text{/gtr/fanchart outer offset}=${\langle offset1 \rangle, \langle offset2 \rangle, \ldots} \quad \text{(no default, initially 0mm)}$

Defines the outer offset values as comma separated list of length values \langle offset1 \rangle, \langle offset2 \rangle, \ldots The last value of this list is taken for all following outer offsets. For standard fan charts, the outer offset describes the marker area of a segment.

$\text{/gtr/fanchart minor angle}=$(angle) \quad \text{(no default, initially -20)}$

Defines the minor \langle angle \rangle of a fan chart. $\text{/gtr/fanchart minor angle}$ has to be smaller than $\text{/gtr/fanchart major angle}$.

$\text{/gtr/fanchart major angle}=$(angle) \quad \text{(no default, initially 200)}$

Defines the major \langle angle \rangle of a fan chart. $\text{/gtr/fanchart major angle}$ has to be greater than $\text{/gtr/fanchart minor angle}$.
/gtr/fanchart angles=\langle{\text{minor angle}}\rangle:\langle{\text{major angle}}\rangle\quad\text{(style, no default)}

Style to set /gtr/fanchart minor angle→P.312=\langle{\text{minor angle}}\rangle and /gtr/fanchart major angle→P.312=\langle{\text{major angle}}\rangle.

\begin{autosizetikzpicture}{\linewidth}{6cm}
\gtrfanchart[ 
  fanchart angles=-20:80, 
  autofill parents male female=3, 
  fanchart template=wave opulent, 
] 
{ parent{ g{} } } 
\end{autosizetikzpicture}

/gtr/fanchart open full

(style)

Style to set /gtr/fanchart minor angle→P.312=-180 and /gtr/fanchart major angle→P.312=180 (full circle).

\begin{autosizetikzpicture}{\linewidth}{6cm}
\gtrfanchart[ 
  fanchart angles=-20:80, 
  autofill parents male female=3, 
  fanchart template=wave opulent, 
] 
{ parent{ g{} } } 
\end{autosizetikzpicture}
\begin{autosizetikzpicture}{\linewidth}{6cm}
\gtrfanchart
  \fanchart open up=150,
  autofill parents male female=3,
  fanchart template=wave opulent,
\}
{ parent{ g{} } }
\end{autosizetikzpicture}

\begin{autosizetikzpicture}{\linewidth}{6cm}
\gtrfanchart
  \fanchart open down=270,
  autofill parents male female=3,
  fanchart template=wave opulent,
\}
{ parent{ g{} } }
\end{autosizetikzpicture}
Style to set \texttt{/gtr/fanchart open left} and \texttt{/gtr/fanchart major angle} such that the fan chart opens direction left with the given opening \texttt{⟨angle⟩}.

\begin{autosizetikzpicture}{\linewidth}{6cm}
gtrfanchart[
  fanchart open left=200,
  autofill parents male female=3,
  fanchart template=wave opulent,
]
{ parent{ g{} } }
\end{autosizetikzpicture}

Style to set \texttt{/gtr/fanchart open right} and \texttt{/gtr/fanchart major angle} such that the fan chart opens direction right with the given opening \texttt{⟨angle⟩}.

\begin{autosizetikzpicture}{\linewidth}{6cm}
gtrfanchart[
  fanchart open right=120,
  autofill parents male female=3,
  fanchart template=wave opulent,
]
{ parent{ g{} } }
\end{autosizetikzpicture}
/gtr/fanchart open for\texttt{=}\langle width\rangle\times\langle height\rangle \quad \text{(style, no default)}

Style to set /gtr/fanchart minor angle\textsuperscript{→P.312} and /gtr/fanchart major angle\textsuperscript{→P.312} such that the width and height of the fan chart diagram has the same ratio as the given \langle width\rangle and \langle height\rangle. The result is either /gtr/fanchart open up\textsuperscript{→P.314} or /gtr/fanchart open right\textsuperscript{→P.315}. Also, /gtr/fanchart text portrait\textsuperscript{→P.319} or /gtr/fanchart text landscape\textsuperscript{→P.319} are adapted automatically.

- The graph is not resized, but the ratio is adapted.
- The root circle is not considered, but the outermost arc only.

\begin{autosizetikzpicture}{9cm}{6cm}
\gtrfanchart
\[fanchart open for=9 \times 6,\]
\[autofill parents male female=3,\]
\[fanchart template=wave opulent,\]
\]
\{ parent{ g{} } \}
\end{autosizetikzpicture}

\begin{autosizetikzpicture}{4cm}{6cm}
\gtrfanchart
\[fanchart open for=4 \times 6,\]
\[autofill parents male female=3,\]
\[fanchart template=wave opulent,\]
\]
\{ parent{ g{} } \}
\end{autosizetikzpicture}
When TikZ calculates the bounding box for an arc, it takes the start, end and control points of the Bezier curves into account. Since the control points usually lie outside the arc, the resulting bounding box may be too large for some situations. To avoid the problem, \texttt{/gtr/fanchart reset bounds} removes the summarized TikZ bounding box after drawing the fan chart and computes a new bounding box.

This removes the mentioned problem but introduces new caveats:

- The computed bounding box presumes a full outermost arc of the fan chart. If some parts are missing, their space is taken into account however. You may call this a drawback or you may call this a feature.
- The draw line width of the outermost arc is not considered. But the resulting bounding box is extended in all directions by \texttt{/gtr/fanchart bounds border} to cope with the missing half line width.
- Every bounding box share of any TikZ path element before \texttt{\gtrfanchart} \textsuperscript{P.310} is also removed.
- TikZ path elements after \texttt{\gtrfanchart} \textsuperscript{P.310} are added as usually to the TikZ bounding box.

\begin{autosizetikzpicture}{9cm}{6cm}
\gtrfanchart[
  fanchart open for=9 x 6, 
  autofill parents male female=3, 
  fanchart template=wave opulent, 
  fanchart reset bounds=false,
]{ parent{ g{} } }
\end{autosizetikzpicture}

If \texttt{/gtr/fanchart reset bounds} is true, the resulting bounding box is extended in all directions by \texttt{(length)}.
13.3 Segment Text Options

\begin{autosizetikzpicture}{\linewidth}{\linewidth}
\gtrfanchartinput[ autofill parents male female=3,
  fanchart template=colorwheel serious,
  fanchart text portrait=inwards,
  fanchart landscape from level=2
]{example.neumann.graph}
\end{autosizetikzpicture}
Feasible values for \( \langle \text{selection} \rangle \) are

- **inwards**: Textflow for portrait in direction of the center.
- **outwards**: Textflow for portrait in direction of the rim.
- **auto**: Upper half inwards, lower half outwards.

Feasible values for \( \langle \text{selection} \rangle \) are

- **clockwise**: Textflow for landscape clockwise.
- **counterclockwise**: Textflow for landscape counterclockwise.
- **auto**: Left half counterclockwise, right half clockwise.
13.4 Color and Style Options

\texttt{/gtr/fanchart boundary color} = \texttt{(color)} \hspace{1cm} \text{(no default, initially \texttt{gray!50})} \\
All fan chart segments are framed with a line in the given \texttt{(color)}. This color is also available with the name \texttt{gtrfanchartboundary}.

\begin{autosizetikzpicture}{\linewidth}{6cm}
\gtrfanchart
[ autofill parents male female=3,
  fanchart template = spartan,
  fanchart boundary color = red,
]
{ parent{ g{} } }
\end{autosizetikzpicture}

\texttt{/gtr/fanchart boundary width} = \texttt{(length)} \hspace{1cm} \text{(no default, initially 0.4pt)} \\
Sets the (base) line width for the boundary line of each segment. This line width is reduced by some heuristic algorithm for very small outer segments.

\begin{autosizetikzpicture}{\linewidth}{6cm}
\gtrfanchart
[ autofill parents male female=3,
  fanchart template = wave opulent,
  fanchart boundary width = 2mm,
]
{ parent{ g{} } }
\end{autosizetikzpicture}
Sets TikZ \texttt{\langle options\rangle} for the root circle path.

\begin{autosizetikzpicture}{\linewidth}{6cm}
gtrfanchart[ autofill parents male female=1, 
fanchart template = spartan, 
fanchart root style = {inner color=red!20,outer color=blue!20}, 
] 
{ parent{ g{name=root} } } 
\end{autosizetikzpicture}

Sets root to display a male or female style according to \texttt{/gtr/fanchart male style} \textsuperscript{P.332}, etc.

\begin{autosizetikzpicture}{\linewidth}{6cm}
gtrfanchart[ autofill parents male female=1, 
fanchart template = spartan, 
fanchart root malefemale, 
] 
{ parent{ g{female,name=root} } } 
\end{autosizetikzpicture}
/gtr/fanchart segment style={⟨options⟩}  
(no default)  

Sets TikZ ⟨options⟩ for a (core) segment path. 

\begin{autosizetikzpicture}{\linewidth}{6cm} 
\gtrfanchart[ autofill parents male female=4, 
fanchart template = spartan, 
fanchart segment style = {top color=red!20,bottom color=blue!20}, ] 
{ parent{ g{} } } 
\end{autosizetikzpicture} 

/gtr/fanchart segment malefemale  
(style, initially set)  

Sets segment to display a male or female style according to /gtr/fanchart male style •P.332, etc. 

\begin{autosizetikzpicture}{\linewidth}{6cm} 
\gtrfanchart[ autofill parents male female=4, 
fanchart template = spartan, 
fanchart segment malefemale, ] 
{ parent{ g{} } } 
\end{autosizetikzpicture}
/gtr/fanchart segment relation (style, initially set)
Sets segment to display a color depending on the relation status of a person set by /gtr/database/relation^{P.167}. The colors are defined by /gtr/fanchart ancestor style^{P.332}, etc.

\begin{autosizetikzpicture}{\linewidth}{6cm}
\gtrfanchartinput[ autofill parents male female=4, 
fanchart template = spartan, 
fanchart text portrait = inwards, 
fanchart segment relation, ]
{example.neumann.graph}
\end{autosizetikzpicture}

/gtr/fanchart segment wave (style)
Sets segment to display a brightened wave spectrum color depending on the angle of the segment. See [2] for the wave color model.

\begin{autosizetikzpicture}{\linewidth}{6cm}
\gtrfanchart[ autofill parents male female=4, 
fanchart template = spartan, 
fanchart segment wave, ]
{ parent{ g{} } }
\end{autosizetikzpicture}
/gtr/fanchart segment colorwheel

Sets segment to display a brightened color wheel color depending on the angle of the segment. See [2] for the \texttt{tHsb} color model and its customization by changing \texttt{\textbackslash rangeHsb}.

\begin{autosizetikzpicture}{\linewidth}{6cm}
\gtrfanchart[ autofill parents male female=4,
  fanchart template = spartan,
  fanchart segment colorwheel,
] { parent{ g{} } }
\end{autosizetikzpicture}

/gtr/fanchart segment radial

Sets segment to display a brightened color series color depending on the level of the current segment. The color is taken from a predefined color series \texttt{fanchart} which can be redefined. See [2] for defining color series.

\begin{autosizetikzpicture}{\linewidth}{6cm}
\gtrfanchart[ autofill parents male female=6,
  fanchart template = spartan,
  fanchart segment radial,
] { parent{ g{} } }
\end{autosizetikzpicture}
Sets TikZ \( \langle \text{options} \rangle \) for a marker path.

\begin{autosizetikzpicture}{\linewidth}{6cm}
\gtrfanchart
\[ autofill parents male female=4, 
  fanchart template = spartan, 
  fanchart outer offset = {5mm}, 
  fanchart marker style = {top color=red!20,bottom color=blue!20}, 
\]
{ parent{ g{} } }
\end{autosizetikzpicture}

Sets marker to display a male or female style according to \gtrfanchart male style \( \rightarrow \) P.332, etc.

\begin{autosizetikzpicture}{\linewidth}{6cm}
\gtrfanchartinput
\[ autofill parents male female=4, 
  fanchart template = spartan, 
  fanchart outer offset = {5mm}, 
  fanchart text portrait=inwards, 
  fanchart male style={fill=blue!50}, fanchart female style={fill=red!50}, 
  fanchart marker malefemale, 
\]
{example.neumann.graph}
\end{autosizetikzpicture}
Sets marker to display a color depending on the relation status of a person set by \texttt{/gtr/database/relation \textsuperscript{P.167}}. The colors are defined by \texttt{/gtr/fanchart ancestor style \textsuperscript{P.332}}, etc.

\begin{autosizetikzpicture}{\linewidth}{6cm}
\gtrfanchartinput
\begin{itemize}
\item autofill parents male female=4,
\item fanchart template = spartan,
\item fanchart outer offset = \{5mm\}, fanchart text portrait = inwards,
\item fanchart marker relation,
\end{itemize}
{example.neumann.graph}
\end{autosizetikzpicture}

Sets marker to display a wave spectrum color depending on the angle of the segment. See \cite{2} for the wave color model.

\begin{autosizetikzpicture}{\linewidth}{6cm}
\gtrfanchart
\begin{itemize}
\item autofill parents male female=4,
\item fanchart template = spartan,
\item fanchart outer offset = \{5mm\},
\item fanchart marker wave,
\end{itemize}
\{ parent{ g{} } \}
\end{autosizetikzpicture}
Sets marker to display a color wheel color depending on the angle of the segment. See [2] for the tHsb color model and its customization by changing `rangeHsb`.

\begin{autosizetikzpicture}{\linewidth}{6cm}
\gtrfanchart [ autofill parents male female=4,  
  fanchart template  = spartan,  
  fanchart outer offset = {5mm},  
  fanchart marker colorwheel,  
]
{ parent{ g{} } } 
\end{autosizetikzpicture}

Sets marker to display a color series color depending on the level of the current segment. The color is taken from a predefined color series `fanchart` which can be redefined. See [2] for defining color series.

\begin{autosizetikzpicture}{\linewidth}{6cm}
\gtrfanchart[ autofill parents male female=4,  
  fanchart template  = spartan,  
  fanchart outer offset = {5mm},  
  fanchart marker radial,  
]
{ parent{ g{} } } 
\end{autosizetikzpicture}
/tikz/gtr set color wave={\langle name\rangle}  (no default)

Defines the color \( \langle name\rangle \) to store the wave spectrum color depending on the angle of the current segment. This option can be used inside /gtr/fanchart segment style → P. 322 or /gtr/fanchart marker style → P. 325. See [2] for the wave color model.

\begin{autosizetikzpicture}{\linewidth}{6cm}
\gtrfanchart
  [ autofill parents male female=4, 
    fanchart template = spartan, 
    fanchart segment style = 
    {
      gtr set color wave = mycolor, 
      top color = mycolor!50!white, 
      bottom color = mycolor!50!gray, 
    }, 
  ] 
{ parent{ g{} } } 
\end{autosizetikzpicture}

/tikz/gtr set color colorwheel={\langle name\rangle}  (no default)

Defines the color \( \langle name\rangle \) to store the color wheel color depending on the angle of the current segment. This option can be used inside /gtr/fanchart segment style → P. 322 or /gtr/fanchart marker style → P. 325. See [2] for the \texttt{tHsb} color model and its customization by changing \texttt{\rangeHsb}. This also includes reduction to e.g. two or four colors for branches:

% example for a color wheel definition with 2 colors
\def\rangeHsb{0.05,240;179.95,240;180.05,0;359.95,0}

% example for a color wheel definition with 4 colors
\def\rangeHsb{0.05,240;89.95,240;90.05,120;179.95,120;180.05,60;269.95,60;270.05,0;359.95,0}

/tikz/gtr set color series={\langle name\rangle}  (no default)

Defines the color \( \langle name\rangle \) to store the color wheel color depending on the level of the current segment. This option can be used inside /gtr/fanchart segment style → P. 322 or /gtr/fanchart marker style → P. 325. The color is taken from a predefined color series \texttt{fanchart} which can be redefined. See [2] for defining color series.
The given TikZ ⟨options⟩ are added to the options given by /gtr/fanchart segment style → P. 322, if the current segment is a complemented segment (by autofill).

\begin{autosizetikzpicture}{\linewidth}{6cm} \gtrfanchart[ autofill parents male female=2, fanchart template = malefemale sober, fanchart complemented segment style = { shade=none, fill=gtrfanchartboundary!5 }, ] { parent{ g{male,name=A} parent{ g{female,name=B} p{male,name=C} } } } \end{autosizetikzpicture}

The given TikZ ⟨options⟩ are added to the options given by /gtr/fanchart marker style → P. 325, if the current segment is a complemented segment (by autofill).

\begin{autosizetikzpicture}{\linewidth}{6cm} \gtrfanchart[ autofill parents male female=2, fanchart template = colorwheel serious, fanchart complemented marker style = { fill=gtrfanchartboundary!20 }, ] { parent{ g{male,name=A} parent{ g{female,name=B} p{male,name=C} } } } \end{autosizetikzpicture}
The given TikZ \textit{options} are added to the options given by /gtr/fanchart segment style \textendash{}P.~322, if the current node /gtr/id is inside the given comma separated \textit{id list}.

\begin{autosizetikzpicture}{\linewidth}{6cm}
\gtrfanchartinput[ autofill parents male female=4,
  fanchart template = spartan,
  fanchart text portrait = inwards,
  fanchart segment style for ids = \{NeumAndr1605,MichNikl1530\}{fill=blue!10} ]
\end{autosizetikzpicture}

The given TikZ \textit{options} are added to the options given by /gtr/fanchart marker style \textendash{}P.~325, if the current node /gtr/id is inside the given comma separated \textit{id list}.

\begin{autosizetikzpicture}{\linewidth}{6cm}
\gtrfanchartinput[ autofill parents male female=4,
  fanchart template = spartan,
  fanchart outer offset = \{5mm\}, fanchart text portrait = inwards,
  fanchart marker style for ids = \{NeumAndr1605,MichNikl1530\}
    \{fill=blue!50!gray!50\} ]
\end{autosizetikzpicture}
Define the styles used by \texttt{/gtr/fanchart segment malefemale $\rightarrow$ P.322} and \texttt{/gtr/fanchart marker malefemale $\rightarrow$ P.325} with the given TikZ \texttt{(options)}.

\begin{verbatim}
/gtr/fanchart male style={⟨options⟩}
  (no default,
   initially top color=blue!1!white,bottom color=blue!6!white)
\end{verbatim}

\begin{verbatim}
/gtr/fanchart female style={⟨options⟩}
  (no default,
   initially top color=red!1!white,bottom color=red!6!white)
\end{verbatim}

\begin{verbatim}
/gtr/fanchart neuter style={⟨options⟩}
  (no default,
   initially top color=gray!1!white,bottom color=gray!6!white)
\end{verbatim}

Define the styles used by \texttt{/gtr/fanchart marker relation $\rightarrow$ P.326} and \texttt{/gtr/fanchart segment relation $\rightarrow$ P.323} with the given TikZ \texttt{(options)}.

\begin{verbatim}
/gtr/fanchart ancestor style={⟨options⟩}
  (no default,
   initially fill=red!50!gray!50)
\end{verbatim}

\begin{verbatim}
/gtr/fanchart descendant style={⟨options⟩}
  (no default,
   initially fill=red!50!gray!50)
\end{verbatim}

\begin{verbatim}
/gtr/fanchart sibling style={⟨options⟩}
  (no default,
   initially fill=blue!50!gray!50)
\end{verbatim}

\begin{verbatim}
/gtr/fanchart unrelated style={⟨options⟩}
  (no default,
   initially fill=gray!15)
\end{verbatim}
13.5  Templates

\gtr/fanchart template=(name)  (style, no default)

Sets a predefined style \langle name \rangle for a fan chart graph. A template does not provide new functionality, but combines various options for specific fan charts, e.g. used inside this documentation. It serves as a shortcut. If a template is used, it is recommended to apply it at the begin of the option list. Further customization options can be added behind.

13.5.1  Template 'spartan'

\begin{autosizetikzpicture}{\linewidth}{7.5cm}
\gtrfanchartinput[ autofill parents male female=3, fanchart template=spartan, fanchart text portrait=inwards ]
{example.neumann.graph}
\end{autosizetikzpicture}

Johann Balthasar Neumann
January 27, 1687 in Eger (Böhmen)
August 19, 1753 in Würzburg (Unterfranken)

Baumeister des Barock und des Rokoko.

Johann Christoph Neumann
1631 in Eger (Böhmen)
August 8, 1713 in Eger (Böhmen)
Bürger und Tuchmacher in Eger.

Andreas Neumann
November 13, 1605 in Eger (Böhmen)
December 9, 1671 in Eger (Böhmen)
Bürger und Tuchmacher in Eger.

Andreas Neumann
September 23, 1576 in Eger (Böhmen)
ca. 1640 in Eger (Böhmen)
Bürger und Tuchmacher in Eger.

Barbara Keßler
ca. 1570 in Eger (Böhmen)
April 12, 1599 in Eger (Böhmen)
ca. 1650 in Eger (Böhmen)

Bürgerstochter aus Eger.

Ester Hoyer
ca. 1600 in Sebenbach bei Eger (Böhmen)
February 11, 1631 in Eger (Böhmen)
ca. 1670 in Eger (Böhmen)

Bauerstochter aus Schönbach.

Lorenz Hoyer
ca. 1560 in Sebenbach bei Eger (Böhmen)
ca. 1640 in Eger (Böhmen)
Bauer in Schönbach.

Rosina Grassoldt
November 29, 1651 in Eger (Böhmen)
August 2, 1672 in Eger (Böhmen)
November 26, 1699 in Eger (Böhmen)

Tuchmacherstochter aus Eger.

Hieronimus Grassoldt
ca. 1600 in Eger (Böhmen)
ca. 1690 in Eger (Böhmen)
Bürger und Tuchmacher in Eger.

Lorenz Grassoldt
ca. 1550 in Eger (Böhmen)
ca. 1630 in Eger (Böhmen)
Bürger und Tuchmacher in Eger.

Barbara Michel
November 3, 1571 in Eger (Böhmen)
October 25, 1587 in Eger (Böhmen)
ca. 1630 in Eger (Böhmen)

Margaretha Hammerschmid
March 8, 1623 in Eger (Böhmen)
September 26, 1645 in Eger (Böhmen)
ca. 1690 in Eger (Böhmen)

Tuchmacherstochter aus Eger.

Nikolaus Hammerschmid
ca. 1570 in Schöba (Böhmen)
1623 in Eger (Böhmen)
Bürger und Tuchmacher zu Eger.

Katharina Diettl
December 20, 1587 in Eger (Böhmen)
January 2, 1611 in Eger (Böhmen)
ca. 1660 in Eger (Böhmen)

Weißgerberstochter aus Eger.


13.5.2 Template 'malefemale sober'

\begin{autosizetikzpicture}{\linewidth}{7.5cm}
\fanchartinput[ autofill parents male female=3,
fanchart template=malefemale sober,
fanchart text portrait=inwards ]
{example.neumann.graph}
\end{autosizetikzpicture}

13.5.3 Template 'malefemale relation'

\begin{autosizetikzpicture}{\linewidth}{7.5cm}
\fanchartinput[ autofill parents male female=3,
fanchart template=malefemale relation,
fanchart text portrait=inwards ]
{example.neumann.graph}
\end{autosizetikzpicture}
13.5.4 Template 'colorwheel sober'

\begin{autosizetikzpicture}{\linewidth}{7.5cm}
\gtrfanchartinput[ autofill parents male female=3, fanchart template=colorwheel sober, fanchart text portrait=inwards ] {example.neumann.graph} \end{autosizetikzpicture}

13.5.5 Template 'colorwheel serious'

\begin{autosizetikzpicture}{\linewidth}{7.5cm}
\gtrfanchartinput[ autofill parents male female=3, fanchart template=colorwheel serious, fanchart text portrait=inwards ] {example.neumann.graph} \end{autosizetikzpicture}
13.5.6 Template 'colorwheel malefemale'

\begin{autosizetikzpicture}{\linewidth}{7.5cm}
\gtrfanchartinput{ autofill parents male female=3,  
  fanchart template=colorwheel malefemale,  
  fanchart text portrait=inwards }
{example.neumann.graph}
\end{autosizetikzpicture}

13.5.7 Template 'colorwheel rich'

\begin{autosizetikzpicture}{\linewidth}{7.5cm}
\gtrfanchartinput{ autofill parents male female=3,  
  fanchart template=colorwheel rich,  
  fanchart text portrait=inwards }
{example.neumann.graph}
\end{autosizetikzpicture}
13.5.8 Template 'colorwheel opulent'

\begin{autosizetikzpicture}{\linewidth}{7.5cm}
\gtrfanchartinput[ autofill parents male female=3,  
    fanchart template=colorwheel opulent,  
    fanchart text portrait=inwards ]
{example.neumann.graph}
\end{autosizetikzpicture}

13.5.9 Template 'wave sober'

\begin{autosizetikzpicture}{\linewidth}{7.5cm}
\gtrfanchartinput[ autofill parents male female=3,  
    fanchart template=wave sober,  
    fanchart text portrait=inwards ]
{example.neumann.graph}
\end{autosizetikzpicture}
13.5.10 Template 'wave serious'

\begin{autosizetikzpicture}{\linewidth}{7.5cm}
\gtrfanchartinput[ autofill parents male female=3,
  fanchart template=wave serious,
  fanchart text portrait=inwards ]
{example.neumann.graph}
\end{autosizetikzpicture}

13.5.11 Template 'wave malefemale'

\begin{autosizetikzpicture}{\linewidth}{7.5cm}
\gtrfanchartinput[ autofill parents male female=3,
  fanchart template=wave malefemale,
  fanchart text portrait=inwards ]
{example.neumann.portrait}
\end{autosizetikzpicture}
13.5.12 Template 'wave rich'

\begin{autosizetikzpicture}{\linewidth}{7.5cm}
\gtrfanchartinput[ autofill parents male female=3, fanchart template=wave rich, fanchart text portrait=inwards ] {example.neumann.graph}
\end{autosizetikzpicture}

13.5.13 Template 'wave opulent'

\begin{autosizetikzpicture}{\linewidth}{7.5cm}
\gtrfanchartinput[ autofill parents male female=3, fanchart template=wave opulent, fanchart text portrait=inwards ] {example.neumann.graph}
\end{autosizetikzpicture}
13.5.14 Template 'radial sober'

\begin{autosizetikzpicture}{\linewidth}{7.5cm}
gtrfanchartinput[ autofill parents male female=3, fanchart template=radial sober, fanchart text portrait=inwards ]
{example.neumann.graph}
\end{autosizetikzpicture}

13.5.15 Template 'radial serious'

\begin{autosizetikzpicture}{\linewidth}{7.5cm}
gtrfanchartinput[ autofill parents male female=3, fanchart template=radial serious, fanchart text portrait=inwards ]
{example.neumann.graph}
\end{autosizetikzpicture}
13.5.16 Template 'radial malefemale'

\begin{autosizetikzpicture}{\linewidth}{7.5cm}
\fanchartinput[ autofill parents male female=3, 
  fanchart template=radial malefemale, 
  fanchart text portrait=inwards ]
{example.neumann.graph}
\end{autosizetikzpicture}

13.5.17 Template 'radial rich'

\begin{autosizetikzpicture}{\linewidth}{7.5cm}
\fanchartinput[ autofill parents male female=3, 
  fanchart template=radial rich, 
  fanchart text portrait=inwards ]
{example.neumann.graph}
\end{autosizetikzpicture}
13.5.18 Template 'radial opulent'

\begin{autosizetikzpicture}{\linewidth}{7.5cm}
\gtrfanchartinput[ autofill parents male female=3,
  fanchart template=radial opulent,
  fanchart text portrait=inwards ]
\{example.neumann.graph\}
\end{autosizetikzpicture}
13.6 Advanced Customization

This section is an API description for people who want to customize the fan chart diagrams even further. It is not intended for “the normal user”.

The \texttt{fanchart} library is written mainly in expl3, but options are defined with \texttt{pgfkeys} from TikZ.

The library redefines the drawing capabilities of \texttt{genealogytree}, but parsing and boxing the node contents is done by common routines of the package. The node content and settings are predominantly not available for segment drawing inside the library with few exceptions:

- \texttt{\textbackslash gtrDBsex}: set by \texttt{/gtr/database/sex}\textsuperscript{P.165}.
- \texttt{\textbackslash gtrDBrelation}: set by \texttt{/gtr/database/relation}\textsuperscript{P.167}.
- \texttt{\textbackslash gtrcomplemented}: set as \texttt{true} or \texttt{false} by \texttt{/gtr/autofill parents male female}\textsuperscript{P.133} and friends.

The following options can be used to replace the drawing procedures.

\begin{verbatim}
N 2020-06-05 /gtr/fanchart-segment-definition={⟨code⟩} (no default)
Used to set up the core segment. This ⟨code⟩ is called after the segment initialization and before the segment is drawn. \texttt{/gtr/fanchart segment style}\textsuperscript{P.322} sets the ⟨code⟩ to \texttt{\getree_fanchart_set_segment_style:n}\textsuperscript{P.344}{#1}.

N 2020-06-05 /gtr/fanchart-marker-definition={⟨code⟩} (no default)
Used to set up the marker. This ⟨code⟩ is called after the segment initialization and before the segment is drawn. \texttt{/gtr/fanchart marker style}\textsuperscript{P.325} sets the ⟨code⟩ to \texttt{\getree_fanchart_set_marker_style:n}\textsuperscript{P.344}{#1}.

N 2020-06-05 /gtr/fanchart-segment-code={⟨code⟩} (no default)
Used to draw the current segment of the circle. This ⟨code⟩ is called after the code of \texttt{/gtr/fanchart-segment-definition} and \texttt{/gtr/fanchart-marker-definition}. Initially, this ⟨code⟩ is equal to \texttt{\getree_fanchart_draw_segment_standard}\textsuperscript{P.344}.

N 2020-06-05 /gtr/fanchart-root-code={⟨code⟩} (no default)
This ⟨code⟩ is used to draw the root node (center of the fan chart). \texttt{/gtr/fanchart root style}\textsuperscript{P.321} sets the ⟨code⟩ to \texttt{\getree_fanchart_draw_root_style:n}\textsuperscript{P.344}{#1}.

N 2020-06-05 \l_getree_fanchart_minor_angle_tl (use read-only)
N 2020-06-05 \l_getree_fanchart_major_angle_tl (use read-only)
These token lists contain the values from \texttt{/gtr/fanchart minor angle}\textsuperscript{P.312} and \texttt{/gtr/fanchart major angle}\textsuperscript{P.312}.

N 2020-06-05 \l_getree_fanchart_angle_a_tl (use read-only)
N 2020-06-05 \l_getree_fanchart_angle_b_tl (use read-only)
These token lists contain the minor angle and the major angle of the current segment of circle.

N 2020-06-05 \l_getree_fanchart_radius_a_tl (use read-only)
N 2020-06-05 \l_getree_fanchart_radius_b_tl (use read-only)
These token lists contain the minor radius and the major radius of the current segment of circle.

N 2020-06-05 \l_getree_fanchart_offset_a_tl (use read-only)
N 2020-06-05 \l_getree_fanchart_offset_b_tl (use read-only)
These token lists contain the inner offset and the outer offset of the current segment of circle.
\end{verbatim}
\lgetree_fanchart_line_width_tl (use read-only) N 2020-06-05
This token list contains the boundary line width of the current segment of circle.

\lgetree_fanchart_level_tl (use read-only) N 2020-06-05
This token list contains the level number of the current segment of circle.

\lgetree_fanchart_ratio_tl (use read-only) N 2020-06-18
This token list contains the relative angle of the center of the current segment taken the interval \([0, 1]\) where 0 means the minor angle and 1 means the major angle. It can be used to compute angle dependent coloring or similar things.

\lgetree_fanchart_id_tl (use read-only) N 2020-06-18
This token list contains the node id of the current segment of circle. It is empty, if the node has no id.

\cgetree_fanchart_maximum_rings_tl (constant 24) N 2020-06-05
This constant token list contains the maximum level number, i.e. the maximum number of segment rings.

\getree_fanchart_if_complemented_node_p:
\getree_fanchart_if_complemented_node:TF \langle true code \rangle \langle false code \rangle
Tests, if the current node is a complemented node, and continues expansion based on the result.

\getree_fanchart_set_segment_style:n \langle style \rangle N 2020-06-05
Sets a Ti\TeX \langle style \rangle for the (core) segment. The \langle style \rangle is used by \getree_fanchart_draw_segment_standard:.

\getree_fanchart_set_marker_style:n \langle style \rangle N 2020-06-05
Sets a Ti\TeX \langle style \rangle for the marker. The \langle style \rangle is used by \getree_fanchart_draw_segment_standard:.

\getree_fanchart_set_color_wave:n \langle name \rangle N 2020-06-05
Defines the color \langle name \rangle to store the wave spectrum color depending on the angle of the current segment.

\getree_fanchart_set_color_colorwheel:n \langle name \rangle N 2020-06-05
Defines the color \langle name \rangle to store the color wheel color depending on the angle of the current segment.

\getree_fanchart_draw_path:n \langle options \rangle N 2020-06-05
Draws the current segment of circle as Ti\TeX \texttt{path} with the given \langle options \rangle.

\getree_fanchart_draw_path:nnn \langle radius1 \rangle \langle radius2 \rangle \langle options \rangle N 2020-06-05
Draws the current segment of circle, but with \langle radius1 \rangle and \langle radius2 \rangle, as Ti\TeX \texttt{path} with the given \langle options \rangle.

\getree_fanchart_draw_root_style:n \langle options \rangle N 2020-06-05
Draws the root node (center) as Ti\TeX \texttt{path} with the given \langle options \rangle.

\getree_fanchart_draw_segment_standard:
Draws the full segment (gap, core, marker) according to all settings.
The customization example above considers a two colored approach where the colors fade from the center to the rim. The segment color is dependent from the angle (\l_getree_fanchart_ratio_tl → \l_getree_fanchart_level_tl) and from the radius (\l_getree_fanchart_ratio_tl → \l_getree_fanchart_level_tl).
14
Auto-Layout Algorithm

14.1 Preliminaries

As discussed before in Chapter 1 on page 9 and Chapter 4 on page 63, genealogy trees can be considered as rooted ordered trees of unbounded degree with annotations. Therefore, an auto-layout algorithm for genealogy trees should be some extension of a known algorithm for tree layout which considers the family-centric approach.

The basic ideas for aesthetic properties and implementation are taken from Reingold and Tilford [3], Walker [6], and Buchheim, Jünger, and Leipert [1]. To dampen expectations early, the actual implementation is some extended Reingold and Tilford algorithm and does not consider the aesthetic balancing of small subtrees as presented in more recent research. There are multifold reasons for this ranging from performance and implementation complexity considerations in pure \LaTeX to the simply question, if balancing is needed or even obstructive for this special application. We will come back to this later.

14.1.1 Aesthetic Properties

First, let us consider aesthetic properties which are usually desired when drawing trees. The following wording is intended for vertically (mainly top-down) oriented trees:

(A1) The $y$ coordinate of a node is given by its level.

(A2) The edges do not cross each other and nodes on the same level have a minimal horizontal distance.

(A3) Isomorphic subtrees are drawn identically up to translation.

(A4) The order of the children of a node is displayed in the drawing.

(A5) The drawing of the reflection of a tree is the reflected drawing of the original tree.

Some of these properties cannot be guaranteed by the implementation and some are even violated deliberately.
14.1.2 Genealogy Trees

In supplement to typical graph theory notions, there is the additional family term for genealogy trees:

(G1) A family is an ordered set of parent nodes and child nodes.

(G2) All parent nodes of a family are connected with edges to all child nodes of the same family.

(G3) A node is child to zero or none family and is parent to zero or arbitrary many families.

These three properties alone would allow to construct genealogy graphs since they do not restrict to a tree-like structure.

A parent node of a family is called a leaf, if it is not child to another family. A child node of a family is called a leaf, if it is not parent to another family.

For genealogy trees, the graph is required to be connected and to comply with exactly one of the following requirements:

(G4a) All child nodes of a family are leaf nodes with exception of at most one (parent tree).

(G4b) All parent nodes of a family are leaf nodes with exception of at most one (child tree).

Finally, we always consider rooted graphs. If (G4a) is fulfilled, there has to be a root family where all child nodes of a family are leaf nodes. If (G4b) is fulfilled, there has to be a root family where all parent nodes of a family are leaf nodes.

It is quite obvious that there are genealogy trees fulfilling (G1)–(G4) which cannot comply with (A2). Edge crossing is quite likely, but should still be minimized. The minimal distance of nodes on the same level may be deliberately different to emphasize different node affiliations.

14.1.3 Graph Grammar

Genealogy trees fulfilling (G1)–(G4) are described by the graph grammar of Chapter 4 on page 63. This given grammar is certainly not without alternative. Also, there are child trees fulfilling (G1)–(G4b) which cannot be represented in this grammar.

- parent constructs including g, c, and p represent parent trees fulfilling (G1)–(G4a).
- child constructs including g, c, p, and union represent child trees fulfilling (G1)–(G4b).
- sandclock constructs are an extension. They are a handy combination of a parent tree and a child tree.

- Nodes which are parent to one family and child to another family are g-nodes.
- The root node of a parent tree or child tree is the g-node of the root family.
14.2 Requirements

The aesthetic properties (A1)–(A5) are generally desired for the implemented auto-layout algorithm. While (A1), (A3), and (A4)\(^1\) are considered to be fulfilled, (A2) cannot be guaranteed for child tree. This was discussed before and can be seen explicitly in Section 14.2.2 on page 350. Property (A5) is loosely considered in the following alignment requirements, but is not covered by special algorithmic efforts. Besides the effects known from the Reingold and Tilford [3] algorithm, there are additional violations of reflections for edge drawing; also see Section 14.2.2 on page 350 for this.

14.2.1 Parent and Child Alignment

The following wording is intended for top-down oriented trees, but applied analogously for other growing directions.

For a family, the parent nodes and the child nodes should be placed centered to each other. This means that the center point of all parents should be vertically in congruence with the center point of all children.

Here, every parent is parent of just one family. If a parent node is parent to more than one family, see Section 14.2.2 on page 350.

\(^1\)One can argue about fulfillment of (A4). The graph grammar restricts children of union constructs to be grouped together while the children of the embedding child can be placed freely. The algorithm displays the order of the children as far as it is described by the grammar, but one could construct trees fulfilling (G1)–(G4b) which cannot by described by the grammar.
14.2.2 Patchwork Families

If a parent node is parent to more than one family, this is described by a child construct which embeds one or more union constructs. The g-node of the child family is also the implicit g-node of all union families. In this case, the combination of the child family with all directly embedded union families is called the patchwork family of the g-node which is parent to all children of the patchwork family.

The parent and child alignment considered in Section 14.2.1 on page 349 is now done for the whole patchwork family. This means that the center point of all parents should be vertically in congruence with the center point of all children of the patchwork family.

While node placement is a straightforward extension to Section 14.2.1 on page 349, edge placement is more difficult since edge crossing is quite likely. Therefore, the interconnections are to be separated vertically. This does not hinder crosspoints, but reduces the probability for lines lying on other lines.

The fulfillment of this requirement results in a best-effort algorithm. The following still small example gives a highlight on the vast amount of possible edge configurations for complex patchwork families.

Consider a child family with three children together with a union with two children, e.g.

```
child{
  g-p-c-c-c-
  union{p-c-c-}
}
```

Depending on the order of the nodes, here, there are 24 different edge configurations:
14.2.3 Graph Growing Direction

The wording used in this chapter applies to top-down oriented trees, but the auto-layout algorithm should consider all four standard directions for graph growing.

The graph growing direction can be selected by setting the `/gtr/timeflow` option appropriately.
14.3 Algorithmic Steps

The following steps use the notations for child trees which grow top-down.

14.3.1 Recursive Family and Node Placement

The tree is constructed recursively in a bottom-up flow. The $y$-coordinate of a family is given by the current level while the $x$-coordinate is computed as relative offset to the enclosing family. This offset is initially 0pt.

During processing for a child family, every enclosed child and union is worked on recursively to construct their corresponding subtrees independently. This results in a list of children subtrees where the direct children of the original child family are the root g-nodes. This list also contains leaf child nodes and all direct leaf child nodes and child subtrees for all enclosed union families.

After the construction of this children list with all their subtrees, each leaf or subtree is placed step by step as close as possible to the right of the already placed last leaf or subtree. The placement is stored into the offset value of a subtree or directly for a leaf node.

The same procedure applies to the parent nodes of the original child family but with reduced complexity since all parents are leaf nodes.

Finally, the center points (pivot points) of all placed children and analogously of all placed parents are computed. All parents are shifted to get both points to congruence.

This concludes the computation for the current child family.

14.3.2 Contours

The core of the algorithm is to place one subtree (or leaf node) as close as possible to the right of another subtree (or leaf node). This is done following the ideas of Reingold and Tilford [3] by housekeeping contours for subtrees. Every child family keeps an anchor to a starting node for its west contour and its east contour. The west contour is the sequence of all leftmost nodes in the whole subtree, while the east contour is the sequence of all rightmost nodes in the whole subtree.

Every node itself has a west contour value and an east contour value describing the relative $x$-coordinate of the left and right border of this node in relation to its enclosing family.

When a west contour is followed starting from its anchor, the next node in the west contour after the current contour node is the leftmost leaf (patchwork) child or the parent node of the very first (patchwork) child, if the current node is no leaf node. Otherwise, a thread is used to note the next node plus a thread gap which is saved for housekeeping.

Analogously, when an east contour is followed starting from its anchor, the next node in the east contour after the current contour node is the rightmost leaf (patchwork) child or the parent node of the very last (patchwork) child, if the current node is no leaf node. Otherwise, a thread is used to note the next node plus a thread gap.

\[2^\text{As described in detail in this document, this algorithm can be adapted with various option settings, e.g. to change the pivot alignment procedure} \]
For direct children or child families, the relative position is known or computed by the family offset values and the stored contour values. For nodes on a thread, this cannot be done and, therefore, the thread gap is needed.

The $\texttt{debug}$ library documented in Chapter 11 on page 249, provides the \texttt{\gtrprocessordebug} command which displays the offset value and the different contour values. Also, \texttt{\gtrdebugdrawcontour} depicts the contours.

### 14.3.3 Combining Subtrees

The combining or sewing of two adjacent subtrees traverses the east contour of the left subtree and the west contour of the right subtree. The distance comparisons of every two contour nodes on the same level gives the required offset value for the right subtree. The combined forest of the two trees inherits the west contour of the left subtree and the east contour of the right subtree. If one of them is shorter than the other, it is prolonged by a thread as required.

This is a shortened depiction of combining two leaf children step by step, followed by the rightmost subtree. Also, the two parent nodes of the new root family are added which are the new anchors for the east contour and the west contour.
14.4 Known Problems

As was already mentioned before, the aesthetic property (A5) is not guaranteed by the auto-layout algorithm. The classic example for this is depicted below using the implemented auto-layout algorithm. Note that the small inner subtrees are not evenly spread but are crowded on the left-hand side.

Next, the classic example is translated to genealogy trees. The effect is the same but arguable may be seen more negligible or at least acceptable. To avoid this automatically, some technique from [3, 6] would be needed.

Luckily, the algorithm is implemented in \LaTeX{} with a lot of intervention points using options. If (A5) is really needed for aesthetic reasons, one can simply cheat by adding some \texttt{/gtr/distance} \(^{\text{P.96}}\) options at the crucial small subtrees:
Another known problem is edge crossing which violates (A2), but this is for some patchwork families sheer unavoidable as even the small examples from Section 14.2.2 on page 350 show. Edge crossing can also happen for childless families, if the usual perpendicular edge drawing is used.

![Diagram of edge crossing](image)

The edge between the two parent nodes (red) of the childless family is overlapped with the edge of the sibling family.

To solve the problem manually, a child with gtr/phantom option can be added to the childless family:

![Diagram of phantom option](image)

The childless family (red) was given a child with the phantom option. This invisible child reserves the space needed for edge drawing.

See Section 5.14 on page 128 for a package aided insertion of such phantom nodes.
The following example graph files are used for various examples inside this document.

### 15.1 example.option.graph

```plaintext
parent[id=SmithDoe]{
g[id=Arth2008,male]{Arthur
\textless	extgreater\text{Born}
,2008}
c[id=Bert2010,female]{Berta
\textless	extgreater\text{Born}
,2010}
c[id=Char2014,male]{Charles
\textless	extgreater\text{Born}
,2014}
parent[id=Smith]{
g[id=John1980,male]{John Smith
\textless	extgreater\text{Born}
,1980}
p[id=GpSm1949,male]{Grandpa Smith
\textless	extgreater\text{Born}
,1949}
p[id=GmSm1952,female]{Grandma Smith
\textless	extgreater\text{Born}
,1952}
}
parent[id=Doe]{
g[id=Jane1982,female]{Jane Doe
\textless	extgreater\text{Born}
,1982}
c[id=Harr1987,male]{Uncle Harry
\textless	extgreater\text{Born}
,1987}
p[id=GpDo1955,male]{Grandpa Doe
\textless	extgreater\text{Born}
,1955}
p[id=GmDo1956,female]{Grandma Doe
\textless	extgreater\text{Born}
,1956}
}
}
```
15.2 example.database.graph

Also see Section 7.2 on page 163.
child[id=fam_A]{
g[id=na1,male]{a_1}
p[id=na2,female]{a_2}
}
child[id=fam_B]{
p[id=nb1,male]{b_1}
g[id=na3,female]{a_3}
c[id=nb2,male]{b_2}
}
child[id=fam_E]{
p[id=ne1,male]{e_1}
g[id=nb3,female]{b_3}
c[id=ne2,male]{e_2}
c[id=ne3,female]{e_3}
}
child[id=fam_C]{
g[id=na4,male]{a_4}
p[id=nc1,female]{c_1}
}
child[id=fam_F]{
g[id=nc2,male]{c_2}
p[id=nf1,female]{f_1}
c[id=nf2,male]{f_2}
c[id=nf3,female]{f_3}
c[id=nf4,male]{f_4}
}
union[id=fam_D]{
p[id=nd1,female]{d_1}
}
child[id=fam_G]{
p[id=ng1,male]{g_1}
g[id=nd2,female]{d_2}
c[id=ng2,male]{g_2}
c[id=ng3,female]{g_3}
}
union[id=fam_H]{
p[id=nh1,male]{h_1}
c[id=nh2,male]{h_2}
}
c[id=nd3,male]{d_3}
child[id=fam_I]{
g[id=nd4,male]{d_4}
p[id=ni1,female]{i_1}
c[id=ni2,female]{i_2}
c[id=ni3,female]{i_3}
c[id=ni4,female]{i_4}
}
c[id=na5,female]{a_5}
p{id=XxxxKath1545}{% female, name={\pref{Katharina} \surn{?}}, shortname={\pref{Katharina} \surn{?}}, unrelated, birth={((caAD)1545){}}, death={((caAD)1620){Eger (Böhmen)}}, %
%}
parent[id=KealBbbb1560]{%
g{id=KexlBarb1570}{% female, name={\pref{Barbara} \surn{Keßler}}, shortname={\pref{Barbara} \surn{Keßler}}, unrelated, birth={((caAD)1570){Eger (Böhmen)}}, marriage={1599-04-12}{Eger (Böhmen)}, death={((caAD)1650){Eger (Böhmen)}}, comment={Bürgerstochter aus Eger}, %
}
p{id=KexlHier1540}{% male, name={\pref{Hieronymus} \surn{Keßler}}, shortname={\pref{Hieronymus} \surn{Keßler}}, unrelated, birth={((caAD)1540){}}, death={((caAD)1590){Eger (Böhmen)}}, comment={Gemeinherr in Eger (1579-1584)}, %
}
p{id=XxxxAgat1550}{% female, name={\pref{Agathe} \surn{?}}, shortname={\pref{Agathe} \surn{?}}, unrelated, birth={((caAD)1550){}}, marriage={((caAD)1560){}}, death={((caAD)1630){Eger (Böhmen)}}, %
%}
parent[id=HoyeBbbb1600]{%
g{id=HoyeEste1600}{% female, name={\pref{Ester} \surn{Hoyer}}, shortname={\pref{Ester} \surn{Hoyer}}, unrelated, birth={((caAD)1600){Sebenbach bei Eger (Böhmen)}}, marriage={1631-02-11}{Eger (Böhmen)}, death={((caAD)1670){Eger (Böhmen)}}, comment={Bauerstochter aus Schönbach}, %
}
p{id=HoyeLore1560}{% male, name={\pref{Lorenz} \surn{Hoyer}}, shortname={\pref{Lorenz} \surn{Hoyer}}, unrelated, birth={((caAD)1560){}}, death={((caAD)1640){Sebenbach bei Eger (Böhmen)}}, comment={Bauer in Schönbach}, %
%}
parent[id=GrasHamm1645]{
  g[id=GrasRosi1651]{
    female,
    name={\pref{Rosina} \surn{Grassoldt}},
    shortname={\pref{Rosina} \surn{Grassoldt}},
    sibling,
    birth={1651-11-29}{Eger (Böhmen)},
    marriage={1672-08-02}{Eger (Böhmen)},
    death={1699-11-26}{Eger (Böhmen)},
    comment={Tuchmacherstochter aus Eger},
  }
  parent[id=GrasMich1587]{
    g[id=GrasHier1600]{
      male,
      name={\pref{Hieronimus} \surn{Grassoldt}},
      shortname={\pref{Hieronimus} \surn{Grassoldt}},
      unrelated,
      birth={(caAD)1600}{Eger (Böhmen)},
      death={(caAD)1690}{Eger (Böhmen)},
      comment={Bürger und Tuchmacher in Eger},
    }
    p[id=GrasLore1550]{
      male,
      name={\pref{Lorenz} \surn{Grassoldt}},
      shortname={\pref{Lorenz} \surn{Grassoldt}},
      unrelated,
      birth={(caAD)1550}{},
      death={(caAD)1630}{Eger (Böhmen)},
      comment={Bürger und Tuchmacher in Eger},
    }
    parent[id=MichBbbb1560]{
      g[id=MichBarb1571]{
        female,
        name={\pref{Barbara} \surn{Michel}},
        shortname={\pref{Barbara} \surn{Michel}},
        unrelated,
        birth={1571-11-03}{Eger (Böhmen)},
        marriage={1587-10-25}{Eger (Böhmen)},
        death={(caAD)1630}{Eger (Böhmen)},
      }
      p[id=MichNikl1530]{
        male,
        name={\pref{Niklas} \surn{Michel}},
        shortname={\pref{Niklas} \surn{Michel}},
        unrelated,
        birth={(caAD)1530}{},
        death={(caAD)1601}{Eger (Böhmen)},
        comment={Bürger und Tuchmacher in Eger},
      }
    }
  }
}
parent[id=HammDiet1611]{
  g[id=HammMarg1623]{
    female,
    name={\pref{Margaretha} \surn{Hammerschmid}},
    shortname={\pref{Margaretha} \surn{Hammerschmid}},
    sibling,
    birth={1623-03-08}{Eger (Böhmen)},
    marriage={1645-09-26}{Eger (Böhmen)},
    death={(caAD)1690}{Eger (Böhmen)},
    comment={Tuchmacherstochter aus Eger},
  }
  parent[id=HammBbbb1570]{
  }
}
name={\pref{Caspar} \textit{surn}{Trapp}},
shortname={\pref{Caspar} \textit{surn}{Trapp}},
ancestor,
birth={\textit{(caAD)1520}}{},
death={\textit{(caAD)1574}}{Eger (Böhmen)},
comment={Weißgerber zu Eger},
}\%
p[id=XxxxAnna1537]{%female,
name={\pref{Anna} \textit{surn}{?}},
shortname={\pref{Anna} \textit{surn}{?}},
ancestor,
birth={\textit{(caAD)1537}}{},
marriage={\textit{(caAD)1554}}{},
death={\textit{(caAD)1610}}{Eger (Böhmen)},
}
The `genealogytree` package provides an elementary stack mechanism which is used internally, but may also be applied elsewhere. This elementary stack stores and retrieves expanded text first-in-last-out (FILO). There are no safe-guarding mechanisms implemented.

### 16.1 Creating a Stack

\[
gtnewstack\{⟨name⟩\}
\]

Creates a new empty stack `⟨name⟩`.

```
\gtnewstack{foo}  \% new empty stack
```

Stack size: \`\gtnstacksize\{foo\}\`

Stack size: 0

\[
gtnstacksize\{⟨name⟩\}
\]

Returns the current stack size.

```
\gtnewstack{foo}
\gtnstacksize{foo}
\gtnstackpush{foo}{a}\par
```

Stack size: \`\gtnstacksize\{foo\}\`

Stack size: 0
Stack size: 1

### 16.2 Push to a Stack

\[
gtnstackpush\{⟨name⟩\}\{⟨content⟩\}
\]

Pushes `⟨content⟩` to a stack `⟨name⟩`. The `⟨content⟩` is expanded during pushing.

```
def\myx\{x\}
\gtnewstack{foo}
\gtnstackpush{foo}{\myx}
\gtnstackpop\{foo\}
```

\`X\`

365
16.3 Pop from a Stack

\texttt{\textbackslash \texttt{gtrstackpop\{}\langle \texttt{name} \rangle \texttt{\}\}}

Pops content from a stack \langle \texttt{name} \rangle. The last pushed content is popped first.

\begin{verbatim}
\texttt{\textbackslash gtrnewstack\{}\texttt{foo}\texttt{\}\}
\texttt{\textbackslash gtrstackpush\{}\texttt{foo}\texttt{\}\{}\texttt{This}\texttt{\}\}
\texttt{\textbackslash gtrstackpush\{}\texttt{foo}\texttt{\}\{}\texttt{is}\texttt{\}\}
\texttt{\textbackslash gtrstackpush\{}\texttt{foo}\texttt{\}\{}\texttt{a}\texttt{\}\}
\texttt{\textbackslash gtrstackpush\{}\texttt{foo}\texttt{\}\{}\texttt{hello}\texttt{\}\}
\texttt{\textbackslash gtrstackpush\{}\texttt{foo}\texttt{\}\{}\texttt{world}\texttt{\}\}
\texttt{\textbackslash gtrstackpop\{}\texttt{foo}\texttt{\}\textbackslash gtrstackpop\{}\texttt{foo}\texttt{\}\textbackslash gtrstackpop\{}\texttt{foo}\texttt{\}\textbackslash gtrstackpop\{}\texttt{foo}\texttt{\}\textbackslash gtrstackpop\{}\texttt{foo}\texttt{\}\}

\texttt{world hello a is This}
\end{verbatim}

\texttt{\textbackslash \texttt{gtrstackpopto\{}\langle \texttt{name} \rangle \texttt{\}\{}\langle \texttt{macro} \rangle \texttt{\}\}}

Pops content from a stack \langle \texttt{name} \rangle into a \langle \texttt{macro} \rangle.

\begin{verbatim}
\texttt{\textbackslash gtrnewstack\{}\texttt{foo}\texttt{\}\}
\texttt{\textbackslash gtrstackpush\{}\texttt{foo}\texttt{\}\{}\texttt{My}\texttt{\}\}
\texttt{\textbackslash gtrstackpush\{}\texttt{foo}\texttt{\}\{}\texttt{test}\texttt{\}\}
\texttt{\textbackslash gtrstackpopto\{}\texttt{foo}\texttt{\}\{}\texttt{\myA}\texttt{\}\}
\texttt{\textbackslash gtrstackpopto\{}\texttt{foo}\texttt{\}\{}\texttt{\myB}\texttt{\}\}
\texttt{\textquotesingle \texttt{\myA} and \textquotesingle \texttt{\myB}.}

\texttt{\textquotesingle test\textquotesingle and \textquotesingle My\textquotesingle.}
\end{verbatim}

16.4 Peek into a Stack

\texttt{\textbackslash \texttt{gtrstackpeek\{}\langle \texttt{name} \rangle \texttt{\}\}}

Reads from a stack \langle \texttt{name} \rangle without reducing the stack content.

\begin{verbatim}
\texttt{\textbackslash gtrnewstack\{}\texttt{foo}\texttt{\}\}
\texttt{\textbackslash gtrstackpush\{}\texttt{foo}\texttt{\}\{}\texttt{First entry}\texttt{\}\}
\texttt{\textquotesingle \texttt{\textbackslash gtrstackpeek\{}\texttt{foo}\texttt{\}\}\textquotesingle, \texttt{\textbackslash gtrstackpeek\{}\texttt{foo}\texttt{\}\};}
\texttt{\textbackslash gtrstackpush\{}\texttt{foo}\texttt{\}\{}\texttt{Second entry}\texttt{\}\}
\texttt{\textquotesingle \texttt{\textbackslash gtrstackpeek\{}\texttt{foo}\texttt{\}\}\textquotesingle, \texttt{\textbackslash gtrstackpeek\{}\texttt{foo}\texttt{\}\};}

\texttt{\textquotesingle First entry\textquotesingle, \textquotesingle First entry\textquotesingle; \textquotesingle Second entry\textquotesingle, \textquotesingle Second entry\textquotesingle;}
\end{verbatim}

\texttt{\textbackslash \texttt{gtrstackpeekto\{}\langle \texttt{name} \rangle \texttt{\}\{}\langle \texttt{macro} \rangle \texttt{\}\}}

Peeks content from a stack \langle \texttt{name} \rangle into a \langle \texttt{macro} \rangle.

\begin{verbatim}
\texttt{\textbackslash gtrnewstack\{}\texttt{foo}\texttt{\}\}
\texttt{\textbackslash gtrstackpush\{}\texttt{foo}\texttt{\}\{}\texttt{My}\texttt{\}\}
\texttt{\textbackslash gtrstackpush\{}\texttt{foo}\texttt{\}\{}\texttt{test}\texttt{\}\}
\texttt{\textbackslash gtrstackpeekto\{}\texttt{foo}\texttt{\}\{}\texttt{\myA}\texttt{\}\}
\texttt{\textbackslash gtrstackpeekto\{}\texttt{foo}\texttt{\}\{}\texttt{\myB}\texttt{\}\}
\texttt{\textquotesingle \texttt{\myA} and \textquotesingle \texttt{\myB}.}

\texttt{\textquotesingle test\textquotesingle and \textquotesingle test\textquotesingle.}
\end{verbatim}
16.5 Creating Stack Shortcuts

\gtrmakestack\{\langle name\rangle\}

Creates a new empty stack \langle name\rangle and creates new macros \langle name\rangle size, \langle name\rangle push, \langle name\rangle pop to, \langle name\rangle pop, \langle name\rangle peek to, and \langle name\rangle peek. These macros serve as shortcuts to the corresponding stack macros from above.

\gtrmakestack\{foo\}
\foopush\{First\}
\foopush\{Second\}
\foopush\{Third\}
The stack contains \foosize\ entries. The last one is '\foopeek'.
\foopop\{myA\}
The stack contains \foosize\ entries after '\myA' was removed.
The remaining entries are '\foopop' and '\foopop'.
Now, the stack contains \foosize\ entries.
Never pop an empty stack: \foopop \foosize

The stack contains 3 entries. The last one is 'Third'.
The stack contains 2 entries after 'Third' was removed.
The remaining entries are 'Second' and 'First'. Now, the stack contains 0 entries.
Never pop an empty stack: -1

\gtrmakestack\{foo\}
\foopush\{Mary\}\foopush\{had\}\foopush\{a\}\foopush\{little\}\foopush\{lamb\}
\loop\ifnum\foosize>0
\foopop,\repeat
lamb, little, a, had, Mary,


Index

a3paper 5 value, 298, 304
a4paper 3 value, 296, 304
a4paper 4 value, 297, 304
above value, 112, 113
AD value, 172
add child key, 221
add parent key, 222
adjust node key, 113
adjust position key, 112
after parser key, 111
after tree key, 115
age key, 167
age code key, 201
ahnentafel 3 value, 290
ahnentafel 4 value, 292
ahnentafel 5 value, 294
all key, 13
all value, 188
all but AD value, 172, 188
ancestor key, 167
ancestor value, 167
anchoring key, 214
auto value, 319
autofill parents female male key, 134
autofill parents female male* key, 134
autofill parents male female key, 133
autofill parents male female* key, 133
autofill parents none key, 134
autofill parents unspecific key, 132
autofill parents unspecific* key, 132
autosizetikzpicture environment, 61
autosizetikzpicture* environment, 61
background key, 214
baptism key, 169, 191
baptism+ key, 169
baptism- key, 169
Baptized key, 245
BC value, 172
below value, 112, 113
birth key, 169, 191
birth+ key, 169
birth- key, 169
Born key, 245
Bornoutofwedlock key, 245
both value, 97, 225, 263
box key, 98
box clear key, 98
burial key, 171, 192
burial+ key, 171
burial- key, 171
Buried key, 245
c grammar, 17, 19, 21, 55, 59, 60, 64, 73, 77, 78, 117, 120, 282, 309, 348
\c_getree_fanchart_maximum_rings_tl, 344
c value, 172
caAD value, 172
caBC value, 172
calendar print key, 188
calendar text for key, 188
catalan value, 247
center value, 214
child value, 97, 225
child distance key, 87
child distance in child graph key, 87
child distance in parent graph key, 86
chinese value, 247
clockwise value, 319
code key, 136
Colors
fanchart, 324, 327, 328
gtr_Blue_1, 306
gtr_Blue_2, 306
gtr_Blue_3, 306
gtr_Blue_4, 306
gtr_Blue_5, 306
gtr_Blue_6, 306
gtr_Bright_Red, 306
gtr_Female_Background, 306
gtr_Female_Frame, 306
gtr_Male_Background, 306
gtr_Male_Frame, 306
gtr_Neuter_Background, 306
gtr_Neuter_Frame, 306
gtr_Peach_Back, 306
gtr_Peach_Frame, 306
gtr_Yellow_Back, 306
gtr_Yellow_Frame, 306
\gtrfanchartboundary, 320
\gtrsymbol, 237, 238
colorwheel malefemale value, 336
colorwheel opulent value, 337
colorwheel rich value, 336
colorwheel serious value, 335
colorwheel sober value, 335
Commands
\c_getree_fanchart_maximum_rings_tl, 344
\genealogytree, 55
\genealogytreenodeinput, 56
\getree_fanchart_drawing_path:n, 344

371
\gtrSymbolsRecordReset, 243
\gtrSymbolsSetCreate, 238
\gtrSymbolsSetCreateSelected, 238
\gtrSymbolsSetDraw, 238
\gtrsymBorn, 239
\gtrsymBornoutofwedlock, 239
\gtrsymBuried, 241
\gtrsymDied, 240
\gtrsymDiedonbirthday, 239
\gtrsymDivorced, 240
\gtrsymEngaged, 240
\gtrsymFemale, 241
\gtrsymFloruit, 241
\gtrsymFuneralurn, 241
\gtrsymKilled, 240
\gtrsymMale, 241
\gtrsymMarried, 240
\gtrsymNeuter, 242
\gtrsymPartnership, 240
\gtrsymStillborn, 239
\gruslibrary, 13
\l_getree_fanchart_angle_a_tl, 343
\l_getree_fanchart_angle_b_tl, 343
\l_getree_fanchart_id_tl, 344
\l_getree_fanchart_level_tl, 344
\l_getree_fanchart_line_width_tl, 344
\l_getree_fanchart_major_angle_tl, 343
\l_getree_fanchart_minor_angle_tl, 343
\l_getree_fanchart_offset_a_tl, 343
\l_getree_fanchart_offset_b_tl, 343
\l_getree_fanchart_radius_a_tl, 343
\l_getree_fanchart_radius_b_tl, 343
\l_getree_fanchart_ratio_tl, 344
\nick, 184
\node, 138, 152
\pref, 184
\surn, 184
\tcbEXTERNALIZE, 49
\tcbbox, 138, 146, 149
\tcbboxfit, 138, 139, 161
comment key, 165
comment code key, 195
complemented key, 135
complemented family key, 135
complemented phantom key, 135
content interpreter key, 155
content interpreter code key, 156
content interpreter content key, 156
content interpreter id and content key, 159
contour key, 263
counterclockwise value, 319
cremated key, 192
cremated value, 171
custom key, 211
d M yyy value, 186
d mon yyyy value, 186
d mon yyyy value, 186
d month yyyy value, 186
d-M-yyyy value, 187
d-m-yyyy value, 187
d-mon-yyyy value, 187
d-month-yyyy value, 187
d.M.yyyy value, 186
d.m.yyyy value, 186
d.mon.yyyy value, 186
d.month yyyy value, 186
d/m yyy value, 186
d/M/yyyy value, 186
d/m/yyyy value, 186
d/mon/yyyy value, 186
d/month/yyyy value, 186
danish value, 247
database content interpreter key, 158
database format key, 174
database pole value, 273, 275, 276
database pole reduced value, 275
database poleportrait value, 276, 278
database poleportrait reduced value, 278
database portrait value, 279, 281
database portrait reduced value, 281
database relationship value, 289
database sideways value, 284, 286, 287
database sideways reduced value, 286
database sidewaysportrait value, 287, 288
database sidewaysportrait reduced value, 288
database traditional value, 282, 283
database traditional reduced value, 283
database unknown key key, 168
date value, 192
date code key, 188
date format key, 186
date range after key, 189
date range before key, 189
date range full key, 189
date range separator key, 189
dd mm yyyy value, 186
dd mon yyyy value, 186
dd mon yyyy value, 186
dd-mm-yyyy value, 186
dd-mon-yyyy value, 187
dd-month-yyyy value, 187
dd.mm.yyyy value, 186
dd.mon.yyyy value, 186
dd mon yyyy value, 186
dh/mm/yyyy value, 186
dh/month/yyyy value, 186
ddmomyyyy value, 187
death key, 170, 192
death+ key, 170
death- key, 170
debug key, 13
deletion content interpreter key, 157
descendant key, 167
descendant value, 167
Died key, 245
died key, 191
died value, 169
Diedonbirthday key, 245
directory value, 272
disconnect key, 225
display 16:10 3 value, 303, 304
display 16:10 4 value, 303, 304
display 16:10 5 value, 303, 304
display 16:9 3 value, 300, 304
display 16:9 4 value, 301, 304
display 16:9 5 value, 302, 304
distance key, 96
divorce key, 170, 192
divorce+ key, 170
divorce- key, 170
Divorced key, 245
don value, 80, 211, 282, 289
dutch value, 247
east value, 263
down value, 303, 304
edges key, 204
down key, 233
down by key, 233
for family key, 206
for subtree key, 206
shift key, 232
up key, 233
up by key, 233
empty value, 174
equal name text key, 185
Engaged key, 245
engagement key, 170, 191
engagement+ key, 170
engagement- key, 170
english value, 244, 247
Environments
  autosizetikzpicture, 61
  autosizetikzpicture*, 61
  exgenealogypicture, 57
genealogypicture, 57
gtreventlist, 193
gerinfolist, 197
gtprintlist, 193
tcolorbox, 138, 143
tikzpicture, 55, 57, 310
error value, 168
event code key, 192
event format key, 192
event text key, 192
exgenealogypicture environment, 57
extra edges key, 226
extra edges for families key, 227
extra edges for family key, 227
extra edges prepend key, 228
extra edges prepend for families key, 229
extra edges prepend for family key, 229
family key, 105
family box key, 99
family database key, 219
family edges key, 205
family extra edges key, 226
family extra edges prepend key, 228
family id key, 262
family label key, 218
family number key, 262
fanchart color, 324, 327, 328
fanchart key, 13
fanchart ancestor style key, 332
fanchart angles key, 313
fanchart boundary color key, 320
fanchart boundary width key, 320
fanchart bounds border key, 317
fanchart complemented marker style key, 329
fanchart complemented segment style key, 329
fanchart descendant style key, 332
fanchart female style key, 332
fanchart inner offset key, 312
fanchart landscape from level key, 318
fanchart major angle key, 312
fanchart male style key, 332
fanchart marker colorwheel key, 327
fanchart marker malefemale key, 325
fanchart marker radial key, 327
fanchart marker relation key, 326
fanchart marker style key, 325
fanchart marker style for ids key, 331
fanchart marker style for levels key, 330
fanchart marker wave key, 326
fanchart minor angle key, 312
fanchart neuter style key, 332
fanchart open down key, 314
fanchart open for key, 316
fanchart open full key, 313
fanchart open left key, 315
fanchart open right key, 315
fanchart open up key, 314
fanchart outer offset key, 312
fanchart radii key, 312
fanchart reset bounds key, 317
fanchart root malefemale key, 321
fanchart root style key, 321
fanchart segment colorwheel key, 324
fanchart segment malefemale key, 322
fanchart segment radial key, 324
fanchart segment relation key, 323
fanchart segment style key, 322
fanchart segment style for ids key, 331
fanchart segment style for levels key, 330
fanchart segment wave key, 323
fanchart sibling style key, 332
fanchart template key, 333
fanchart text landscape key, 319
fanchart text portrait key, 319
fanchart unrelated style key, 332
fanchart-marker-definition key, 343
fanchart-root-code key, 343
fanchart-segment-code key, 343
fanchart-segment-definition key, 343
Female key, 245
female key, 101, 165
female value, 165, 198
fill overzoom DBimage key, 200
fill zoom DBimage key, 200
fit value, 138, 139, 270
fit to family key, 116
fit to subtree key, 116
Floruit key, 245
floruit key, 170, 192
floruit+ key, 170
floruit- key, 170
foreground key, 213

g formal graph value, 269
genus 

genealogytree environment, 57
\genealogytree, 55
genealogytree edges scope key, 207
genealogytree extra edges scope key, 230
\genealogytreeinput, 56
german value, 245, 247
german-austrian value, 247
german-german value, 247
\getree_fanchart_draw_path:n, 344
\getree_fanchart_draw_path:nnn, 344
\getree_fanchart_draw_root_style:n, 344
\getree_fanchart_draw_segment_standard:, 344
\getree_fanchart_if_complemented_node:TF, 344
\getree_fanchart_if_complemented_node_p:, 344
\getree_fanchart_set_color_colorwheel:n, 344
\getree_fanchart_set_color_wave:n, 344
\getree_fanchart_set_marker_definition:n, 344
\getree_fanchart_set_segment_style:n, 344
\getree_set_options_for_family:nn, 104
\getree_set_options_for_node:nn, 95
\getree_set_options_for_subtree:nn, 107
GR value, 172
Grammar
c, 17, 19, 21, 55, 59, 60, 64, 73, 77, 78, 117, 120, 282, 309, 348
g, 10, 17, 19, 25, 26, 34, 55, 59, 60, 64, 65, 67, 69, 71, 73, 77, 78, 117, 225, 267, 282, 290, 292, 294, 296–303, 348, 350, 352
input, 29, 74
insert, 75
p, 17, 19, 21, 32, 55, 59, 60, 64, 73, 77, 78, 117, 282, 290, 292, 294, 296–303, 348
sandclock, 34, 39, 55, 60, 64, 71, 106, 128, 130, 348
union, 26, 27, 55, 64, 67, 69, 106, 128, 225, 231, 261, 267, 348–350, 352
gtr set color colorwheel key, 328
gtr set color series key, 328
gtr set color wave key, 328
gtr_BLUE_1 color, 306
gtr_BLUE_2 color, 306
gtr_BLUE_3 color, 306
gtr_BLUE_4 color, 306
gtr_BLUE_5 color, 306
gtr_BLUE_6 color, 306
gtr_Bright_Red color, 306
gtr_Female_Background color, 306
gtr_Female_Frame color, 306
gtr_Male_Background color, 306
gtr_Male_Frame color, 306
gtr_Neuter_Background color, 306
gtr_Neuter_Frame color, 306
gtr_Peach_Back color, 306
gtr_Peach_Frame color, 306
gtr_Yellow_Back color, 306
gtr_Yellow_Frame color, 306
\gtrautosizebox, 61
\gtrautosizebox*, 61
\gtBoxContent, 154
\gtrcomplemented, 343
\gtrDBage, 167
\gtrDBcomment, 165
\gtrDBimage, 38, 166
\gtrDBimageopt, 166
\gtrDBkekule, 167
\gtrDBname, 165
\gtrDBprofession, 166
\gtrDBrelation, 167
\gtrDBrelationship, 167
\gtrDBsex, 36, 165, 198, 343
\gtrDBshortname, 165
\gtrDBuuid, 166
\gtrdebugdrawcontour, 261
\gtDeclareDatabaseFormat, 182
\gtDrawSymbolicPortrait, 308
\gtredgeset, 207
\gtreventlist environment, 193
image key, 166
image prefix key, 200
imageopt key, 166
info separators key, 197
input grammar, 29, 74
insert grammar, 75
insert after family key, 122
insert after node key, 121
insert at begin family key, 123
insert at end family key, 124
insert for childless families key, 129
insert for childless families level limit key, 131
insert phantom for childless families key, 130
inwards value, 319
italian value, 247
JU value, 172
kekele key, 167

Keys

/gtr/
  add child, 221
  add parent, 222
  adjust node, 113
  adjust position, 112
  after parser, 111
  after tree, 115
  age code, 201
  autofill parents female male, 134
  autofill parents female male*, 134
  autofill parents male female, 133
  autofill parents male female*, 133
  autofill parents none, 134
  autofill parents unspecified, 132
  autofill parents unspecified*, 132
  box, 98
  box clear, 98
  calendar print, 188
  calendar text for, 188
  child distance, 87
  child distance in child graph, 87
  child distance in parent graph, 86
  code, 136
  comment code, 195
  complemented, 135
  complemented family, 135
  complemented phantom, 135
  content interpreter, 155
  content interpreter code, 156
  content interpreter content, 156
  content interpreter id and content, 159
  database content interpreter, 158
  database format, 174
  database unknown key, 168
  date code, 188
  date format, 186
  date range after, 189
  date range before, 189
  date range full, 189
  date range separator, 189
  deletion content interpreter, 157
  disconnect, 225
  distance, 96
  edges, 204
  edges down, 233
  edges down by, 233
  edges for family, 206
  edges for subtree, 206
  edges shift, 232
  edges up, 233
  edges up by, 233
  empty name text, 185
  event code, 192
  event format, 192
  event text, 192
  extra edges, 226
  extra edges for families, 227
  extra edges for family, 227
  extra edges prepend, 228
  extra edges prepend for families, 229
  extra edges prepend for family, 229
  family, 105
  family box, 99
  family database, 219
  family edges, 205
  family extra edges, 226
  family extra edges prepend, 228
  family label, 218
  fanchart ancestor style, 332
  fanchart angles, 313
  fanchart boundary color, 320
  fanchart boundary width, 320
  fanchart bounds border, 317
  fanchart complemented marker style, 329
  fanchart complemented segment style, 329
  fanchart descendant style, 332
  fanchart female style, 332
  fanchart inner offset, 312
  fanchart landscape from level, 318
  fanchart major angle, 312
  fanchart male style, 332
  fanchart marker colorwheel, 327
  fanchart marker malefemale, 325
  fanchart marker radial, 327
  fanchart marker relation, 326
  fanchart marker style, 325
  fanchart marker style for ids, 331
  fanchart marker style for levels, 330
  fanchart marker wave, 326
  fanchart minor angle, 312
  fanchart neuter style, 332
  fanchart open down, 314
fanchart open for, 316
fanchart open full, 313
fanchart open left, 315
fanchart open right, 315
fanchart open up, 314
fanchart outer offset, 312
fanchart radii, 312
fanchart reset bounds, 317
fanchart root malefemale, 321
fanchart root style, 321
fanchart segment colorwheel, 324
fanchart segment malefemale, 322
fanchart segment radial, 324
fanchart segment relation, 323
fanchart segment style, 322
fanchart segment style for ids, 331
fanchart segment style for levels, 330
fanchart segment wave, 323
fanchart sibling style, 332
fanchart template, 333
fanchart text landscape, 319
fanchart text portrait, 319
fanchart unrelated style, 332
fanchart-marker-definition, 343
fanchart-root-code, 343
fanchart-segment-code, 343
fanchart-segment-definition, 343
female, 101
further distance, 90
id, 92
id content interpreter, 158
id prefix, 94
id suffix, 94
ignore, 117
ignore family database, 219
ignore level, 119
ignore node, 118
ignore parent childs, 120
ignore subtree, 118
info separators, 197
insert after family, 122
insert after node, 121
insert at begin family, 123
insert at end family, 124
insert for childless families, 129
insert for childless families level limit, 131
insert phantom for childless families, 130
keysfrom, 136
label, 217
label database options, 219
label options, 217
language, 247
level, 109
level distance, 83
level n, 110
level size, 84
list separators, 194
list separators hang, 194
male, 101
name, 184
name code, 185
name font, 185
neuter, 101
nick code, 184
no content interpreter, 157
node, 96
node box, 98
node processor, 138
node size, 85
node size from, 86
nullify edge level shift, 235
options for family, 104
options for node, 95
options for subtree, 107
parent distance, 89
parent distance in child graph, 89
parent distance in parent graph, 88
phantom, 125
phantom*, 127
pivot, 97
pivot shift, 106
place text, 190
pref code, 184
proband level, 111
processing, 138
profession code, 196
remove child, 224
remove parent, 224
reset, 136
reset edge level shift, 234
set position, 111
show, 265
show family, 267
show id, 265
show level, 266
show number, 266
show type, 267
subtree, 108
subtree box, 99
subtree edges, 206
subtree label, 218
subtree surn, 184
switch edge level shift, 234
symbols record reset, 243
tcbset, 115
template, 269
template horizontal border, 304
template vertical border, 304	tikz, 103
tikzpicture, 114	tikzset, 114	timeflow, 80	tree offset, 111
turn, 100
use family database, 219
fit to family, 116
fit to subtree, 116
genealogytree edges scope, 207
genealogytree extra edges scope, 230
gtr set color colorwheel, 328
gtr set color series, 328
gtr set color wave, 328
gtrNodeDimensions, 152
gtrNodeDimensionsLandscape, 152
keysfrom key, 136
Killed key, 245
killed key, 192
killed value, 170
\l_getree_fanchart_angle_a_tl, 343
\l_getree_fanchart_angle_b_tl, 343
\l_getree_fanchart_id_tl, 344
\l_getree_fanchart_level_tl, 344
\l_getree_fanchart_line_width_tl, 344
\l_getree_fanchart_major_angle_tl, 343
\l_getree_fanchart_minor_angle_tl, 343
\l_getree_fanchart_offset_a_tl, 343
\l_getree_fanchart_offset_b_tl, 343
\l_getree_fanchart_radius_a_tl, 343
\l_getree_fanchart_radius_b_tl, 343
\l_getree_fanchart_ratio_tl, 344
label key, 217
label database options key, 219
label options key, 217
language key, 247
left value, 80, 100, 112, 113, 284, 287, 290, 292, 294
letterpaper 3 value, 299, 304
letterpaper 4 value, 299, 304
level key, 109
level distance key, 83
level n key, 110
level size key, 84
list separators key, 194
list separators hang key, 194
m-d-yyyy value, 187
m.d.yyyy value, 187
m/d/yyyy value, 187
Male key, 245
male key, 101, 165
male value, 165, 198
malefemale relation value, 334
malefemale sober value, 334
marriage key, 170, 191
marriage value, 174, 219
marriage+ key, 170
marriage- key, 170
Married key, 245
medium value, 174, 284, 287
medium marriage above value, 174
medium marriage below value, 174
medium no marriage value, 174, 289
mesh key, 210
mm-dd-yyyy value, 187
mm.dd.yyyy value, 187
mm/dd/yyyy value, 187
mon d yyyy value, 187
mon.d yyyy value, 187
month d yyyy value, 187
name key, 165, 184
name value, 174
name code key, 185
name font key, 185
Neuter key, 245
neuter key, 101, 165
neuter value, 165
\nick, 184
nick code key, 184
no background key, 214
no content interpreter key, 157
no foreground key, 213\node, 138, 152
node key, 96
node box key, 98
node processor key, 138
node size key, 85
node size from key, 86
none key, 212
none value, 97, 188, 263
nullify edge level shift key, 235
off value, 100
options for family key, 104
options for node key, 95
options for subtree key, 107
other key, 192
other value, 170
out of wedlock key, 191
out of wedlock value, 169
outwards value, 319
p grammar, 17, 19, 21, 32, 55, 59, 60, 64, 73, 77, 78, 117, 282, 290, 292, 294, 296–303, 348
parent value, 97, 225
parent distance key, 89
parent distance in child graph key, 89
parent distance in parent graph key, 88 Partnership key, 245
periphery value, 214
perpendicular key, 208
phantom key, 125
phantom* key, 127
pivot key, 97
pivot shift key, 106
place text key, 190
portuguese value, 247
\pref, 184
pref code key, 184
prefix date value, 192
prefix date place value, 192
proband level key, 111
processing key, 138
profession key, 166
profession code key, 196
radial malefemale value, 341
radial opulent value, 342
radial rich value, 341
radial serious value, 340
radial sober value, 340
relation key, 167
relationship key, 167
remove child key, 224
remove parent key, 224
reset key, 136
reset edge level shift key, 234
right value, 80, 100, 112, 113, 272
rounded key, 208
rounded value, 41

sandclock grammar, 34, 39, 55, 60, 64, 71, 106, 128, 130, 348
save value, 168
set position key, 111
sex key, 165
short value, 174, 184
short marriage above value, 174
short marriage below value, 174
short no marriage value, 174, 279, 282
shortname key, 165
show key, 265
show family key, 267
show id key, 265
show level key, 266
show number key, 266
show type key, 267
sibling key, 167
sibling value, 167
signpost value, 270
spanish value, 247
spartan value, 333
Stillborn key, 245
Stillborn key, 191
Stillborn value, 169
subtree key, 108
subtree box key, 99
subtree edges key, 206
subtree label key, 218
\surn, 184
surn code key, 184
swedish value, 247
swing key, 209
switch edge level shift key, 234
symbol value, 174
symbol nodes value, 270
symbols record reset key, 243
\tcbEXTERNALIZE, 49
\tcbox, 138, 146, 149
tcbox value, 138, 146
tcbox* value, 138, 149, 269–272
\tcboxfit, 138, 139, 161
tcbset key, 115
tcolorbox environment, 138, 143
tcolorbox value, 138, 143
template key, 269
template horizontal border key, 304
template vertical border key, 304
templates key, 13	ikz key, 103
tikznodel value, 103, 138, 152
tikzpicture environment, 55, 57, 310
tikzpicture key, 114
tikzset key, 114
timeflow key, 80	iny boxes value, 271
tiny circles value, 271
tree offset key, 111
turn key, 100
typical value, 186
union grammar, 26, 27, 55, 64, 67, 69, 106, 128, 225, 231, 261, 267, 348–350, 352
unrelated key, 167
unrelated value, 167
up value, 80
upsidedown value, 100
use family database key, 219
uuid key, 166

Values

\a3paper 5, 298, 304
\a4paper 3, 296, 304\a4paper 4, 297, 304
above, 112, 113
AD, 172
ahnentafel 3, 290
ahnentafel 4, 292
ahnentafel 5, 294
all, 188
all but AD, 172, 188
ancestor, 167
auto, 319
BC, 172
below, 112, 113
both, 97, 225, 263
car, 172
cased, 172
cased, 172
catalan, 247
center, 214
choice, 97, 225
chinese, 247
clockwise, 319
colorwheel malefemale, 336
colorwheel opulent, 337
colorwheel rich, 336
colorwheel serious, 335
colorwheel sober, 335
counterclockwise, 319
cremated, 171
d M yyyy, 186
d mon yyyy, 186
d mon.yyyy, 186
d month yyyy, 186
d-M-yyyy, 187
d-m-yyyy, 187
d-mon-yyyy, 187
d-month-yyyy, 187
d. M yyyy, 186
d.m yyyy, 186
d.mon yyyy, 186
d.month yyyy, 186
d m yyyy, 186
d/M yyyy, 186
d/mon yyyy, 186
d/month yyyy, 186
danish, 247
database pole, 273, 275, 276
database pole reduced, 275
database poleportrait, 276, 278
database poleportrait reduced, 278
database portrait, 279, 281
database portrait reduced, 278
database relationship, 289
database sideways, 284, 286, 287
database sideways reduced, 286
database sidewaysportrait, 287, 288
database sidewaysportrait reduced, 288
database traditional, 282, 283
database traditional reduced, 283
date, 192
dd mm yyyy, 186
dd mon yyyy, 186
dd mon.yyyy, 186
dd-mm yyyy, 186
dd-mon yyyy, 187
dd-month yyyy, 187
dd.mm yyyy, 186
dd.mon yyyy, 186
dd/mm yyyy, 186
dd/mon yyyy, 186
dd/month yyyy, 186
ddmonyyyy, 187
descendant, 167
died, 169
directory, 272
display 16:10 3, 303, 304
display 16:10 4, 303, 304
display 16:10 5, 303, 304
display 16:9 3, 300, 304
display 16:9 4, 301, 304
display 16:9 5, 302, 304
down, 80, 211, 282, 289
dutch, 247
east, 263
eempty, 174
english, 244, 247
error, 168
female, 165, 198
fit, 138, 139, 270
formal graph, 269
french, 247
full, 174, 184
full marriage above, 174, 273, 276
full marriage below, 174, 296–303
full no marriage, 174
german, 245, 247
german-austrian, 247
german-german, 247
GR, 172
ignore, 168
inwards, 319
italian, 247
JU, 172
killed, 170
left, 80, 100, 112, 113, 284, 287, 290, 292, 294, 296–303
letterpaper 3, 299, 304
letterpaper 4, 299, 304
m-d yyyy, 187
m.d yyyy, 187
m/dd yyyy, 187
male, 165, 198
malefemale relation, 334
malefemale sober, 334
marriage, 174, 219
medium, 174, 284, 287
medium marriage above, 174
medium marriage below, 174
medium no marriage, 174, 289
mm/dd yyyy, 187
mon d yyyy, 187
mon.d yyyy, 187
month d yyyy, 187
name, 174
neuter, 165
none, 97, 188, 263
off, 100
other, 170
out of wedlock, 169
outwards, 319
parent, 97, 225
periphery, 214
portuguese, 247
prefix date, 192
prefix date place, 192
radial malefemale, 341
radial opulent, 342
radial rich, 341
radial serious, 340
radial sober, 340
right, 80, 100, 112, 113, 272
rounded, 41
save, 168
short, 174, 184
short marriage above, 174
short marriage below, 174
short no marriage, 174, 279, 282
sibling, 167
signpost, 270
spanish, 247
spartan, 333
stillborn, 169
swedish, 247
symbol, 174
symbol nodes, 270
tcbox, 138, 146
tcbox*, 138, 149, 269–272
tcolorbox, 138, 143
tikznode, 103, 138, 152
tiny boxes, 271
tiny circles, 271
typical, 186
unrelated, 167
up, 80
upside down, 100
warn, 168
wave malefemale, 338
wave opulent, 339
wave rich, 339
wave serious, 338
wave sober, 337
west, 263
yyyy, 187
yyyy d/m, 187
yyyy mm dd, 187
yyyy mon dd, 187
yyyy month d, 187
yyyy-mm-dd, 187
yyyy-mon-d, 187
yyyy-mon-dd, 187
yyyy.m.d, 187
yyyy.m.d. value, 187
yyyy.mm.dd value, 187
yyyy.month d. value, 187
yyyy-month d., 187
yyyy.m/d value, 187
yyyy/mm/dd value, 187
yyyymmdd value, 187
yyyymondd value, 187
viewport key, 166
warn value, 168
wave malefemale value, 338
wave opulent value, 339
wave rich value, 339

xshift key, 215

yshift key, 216

yyy value, 187

xxxx d/m value, 187

yyyy mm mm value, 187

yyyy mon mm value, 187

yyyy month d value, 187

yyyy-mm-mm value, 187

yyyy-mon-m value, 187

yyyy-month m value, 187

yyyy.m.m value, 187

yyyy/mm/mm value, 187

yyyymmmd value, 187

yyyymonmd value, 187

viewports key, 166

warn value, 168

wave malefemale value, 338

wave opulent value, 339

wave rich value, 339