Typesetting Karnaugh Maps with LaTeX and Ti*k*Z

Luis Paulo Laus
e-mail: laus@utfpr.edu.br

Version: 1.3, Version date: 2021-10-14

1 Abstract

Karnaugh maps are used to simplify logic equations leading to the most compact expression of two levels for a given truth table. The drawing of them used to be a boring, annoying and error-prone task. This set of macros intend to simplify the task. They can typeset Karnaugh maps with up to twelve variables[1], which is more than you might likely need[2]. You only have to provide a list of variable identifiers plus the truth table of your logic function. The macros also allow to highlight the simplifications of your logic function directly within a Karnaugh map. This package is based on `kvmacros.tex` from `karnaugh` package referred herein as "the original one". The modifications carried out intended to use Ti*k*Z instead of native LaTeX commands allowing easier customisation, easier extension when you need to draw other elements along with the map and leading to higher graph quality. The labels that indicate values of the variables across de rows and columns may be represented in both numerical and simplified form.

2 Introduction

Karnaugh maps [1] and Veitch charts are used to simplify logic equations. They are map representations of logic functions, and in that they are equivalent. Veitch charts are not supported by this package, but it should not be a big problem to port Andreas W. Wieland's `veitch` macro, available in `karnaugh` package, if you need it. Please note that this package, including its documentation, is based on Andreas W. Wieland's previous work and the author wishes to register his acknowledgment.

2.1 Comparison with Other Packages

If you ask yourself "why another Karnaugh map typesetting package?" the answer is easy: because I was not completely happy with the available packages I know and those are:

1. `karnaugh`: it is a great package that uses native LaTeX commands to draw the map. It supports Karnaugh maps and Veitch charts. It employs a recursive algorithm with no size limit[3] which leads to an interesting kind of symmetry. Remember, Karnaugh maps are all about symmetry. It is not customisable, for instance, one cannot change the distance between bars[4] (the marks showing around the map with variable identifiers on them) and if the variable identifier is long, it will overlap another bar. Also, I want to use Ti*k*Z to draw colourful semi-transparent figures on to top of the map to highlight groups (prime implicants) and, although it is possible, it is rather difficult and the result is not very good because they always look a bit off. I have a long-time experience with this package and I have also written a java program to draw the maps because, though typesetting simple maps is easy, highlighting the prime implicants is not.

2. `karnaughmap`: it uses Ti*k*Z so you got a lot of options for customisation. It is limited to eight variable which, to be honest, should be enough for anyone. The problem is that it only draws bars (those

[1] The actual limit may be different for you.
[2] A twelve variables map contains of 4096 cells in a 64×64 grid. They are simply too big to handle manually and you should consider to use a software.
[3] It works until you blow the memory out which will happen about ten to twelve variables.
[4] Those bars have been underappreciated along the history. Karnaugh [1] himself called them "simplified labels" and used them only to replace the Gray coded numbers showing around the map. Their true strength is the ease way they point out which variable belong to a prime implicant and which does not. An approach much easier than interpreting the Gray coded numbers.

marking mentioned above) up to four variables. Also, the order in which the variable list is inputted is different from the order employed by `karnaugh`.

3. `askmaps`: this package generates configurable American style Karnaugh maps for 2, 3, 4 and 5 variables. In America, instead of bars denoting the one value of variables, they use Gray coded binaries on the top and left side of the map. This behaviour is supported with `tikz-karnaugh` (see Section 7), though, in my twenty years of experience teaching the subject, I have found out that bars are much more intuitive. The `askmaps` contains four macros, one for each number of variables, and it can be used to highlight the prime implicants in the very same way that `karnaugh` does.

4. `karnaugh-map`: uses Ti*k*Z to draw up to four maps of four variables leading to a 3D six variables map. It contains commands for drawing implicants on top of the map. Like `askmaps`, this package uses Gray code instead of bars.

5. `kvmap`: this is a relatively new package (release on 16 September, 2020) that allows you to typeset a Karnaugh map similarly as you use a tabular environment. In Section 10 the same result is obtained using a java software, but in this case the implicants (bundles or groups) highlighting is done automatically. If you interested in seeing this future in `tikz-karnaugh` please let me know.

6. `cartonaugh`: This is also a relatively new package (release on 15 July, 2021). It is a fork of `karnaugh-map` package that draws Karnaugh maps with up to 6 variables. The documentation of both packages looks very similar; one notable difference is the position of the function and variable labels are equivalent as used `tikz-karnaugh` if `American style` is enabled.

With `tikz-karnaugh` you can typeset big (up to twelve variables or 4096 cells) good looking maps with Gray code labels (American style) or bars (simplified labels). Using a companion free java software (JQM, see Section 10), you can do it automatically, including highlighting the solution.

2.2 Introductory example

Let us start with an introduction on how to use these macros. The first thing you have to do is to load Ti*k*Z. For this type `\usepackage{tikz}` in the preamble of your document. Then, if the package is somewhere TEX can find it, load the library with the command `\usetikzlibrary{karnaugh}`. If it is not, you can use something like `\input tikzlibrarykarnaugh.code`. You may need to provide the full or relative path to file `tikzlibrarykarnaugh.code.tex`.

Suppose now you have a logic function f with the following truth table:

Index	a	b	c	d	f
0	0	0	0	0	1
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	0
8	1	0	0	0	0
9	1	0	0	1	1
10	1	0	1	0	1
11	1	0	1	1	0
12	1	1	0	0	0
13	1	1	0	1	1
14	1	1	1	0	1
15	1	1	1	1	0

This logic function can easily be put into a Karnaugh map by using the `\karnaughmap` macro in a Ti*k*Z environment (`\begin{tikzpicture}`) or inline command (`\tikz`). The `\karnaughmap` macro has five mandatory parameters:

1. the number of variables in the map;

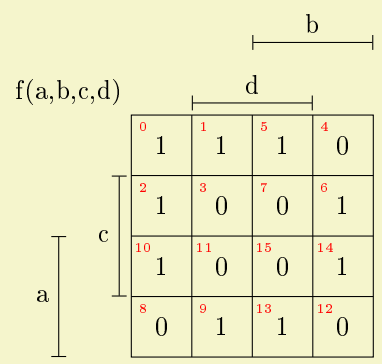
2. an identifier for the function;

3. a list of variable identifiers for the variables;

4. the list of values of f for each line in the truth table; and

5. a possibly empty set of TikZ commands that will be drown before the function values so the values will appear on top of them.

The variable identifiers in the third parameter are ordered from highest to lowest significance (the same way as in the truth table, with a having a significance of $2^3 = 8$ and d having a significance of $2^0 = 1$). The list of values of f was read from lowest to highest index. The fifth parameter remains empty in this example, it will be discussed further on:



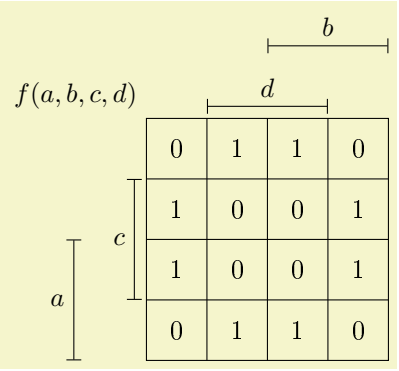

```
\tikz[karnaugh,enable indices]
  \karnaughmap{4}{f(a,b,c,d)}{abcd}{1110 0110 0110 0110}{};
```

The indices in the upper left corner of each cell corresponds to the indices in the truth table. The indices can easily be calculated from the variable value in the truth table, e.g., the index for row 11 is given by:

$$2^3\,a + 2^2\,b + 2^1\,c + 2^0\,d = 8\,a + 4\,b + 2\,c + 1\,d = 8 + 2 + 1 = 11.$$

The macros that read the variable list and the list of logic values (i.e., parameters #3 and #4) work recursively. This is why variables a and b are assorted with Karnaugh map rows and b and d with columns. If you need a desire a different variable arrangement, you will not only need to change parameter #3, but also #4 accordingly. This can be troublesome; see Section 10 for a java software that can help on this matter.

Each entry has to be one character long and spaces are allowed[5], otherwise – like a variable identifier enclosed in $s – you have to put it into curly brackets:

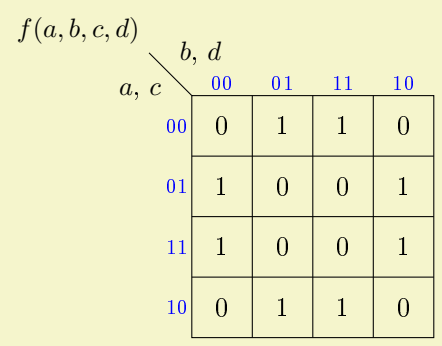



```
\begin{tikzpicture}[karnaugh]
  \karnaughmap{4}{$f(a,b,c,d)$}{{$a$}{$b$}{$c$}{$d$}}%
     {0110 0110 0110 0110}{}
\end{tikzpicture}
```

Observe that the labels are all in math mode in this example. Also, a TikZ environment was used so there is no semicolon (;) in the end of `\karnaughmap` macro. Moreover, the indices were omitted by removing `enable indices` from the options list.

If you prefer Gray coded labels, referred herein as *American style*, just type `American style` in the option list. A little adjustment in the space between the map and the number would be in order too:

[5] White spaces are really usable to make the string more readable leading to fast verification.




```
\begin{tikzpicture}[karnaugh,
                    American style,
                    kmbar top sep=0.2|kmunitlength,
                    kmbar left sep=-0.1|kmunitlength]
  \karnaughmap{4}{$f(a,b,c,d)$}{{$a$}{$b$}{$c$}{$d$}}%
     {0110 0110 0110 0110}{}
\end{tikzpicture}
```

See Section 7 for more details on American styled maps. Also, the switch `Gray index` might be of interest.

3 Karnaugh Map Library

Ti*k*Z Library karnaugh

```
\usepgflibrary{karnaugh}  % LaTeX and plain TeX and pure pgf
\usepgflibrary[karnaugh]  % ConTeXt and pure pgf
\usetikzlibrary{karnaugh} % LaTeX and plain TeX when using TikZ
\usetikzlibrary[karnaugh] % ConTeXt when using TikZ
```

This library provides TEX macros to typeset Karnaugh maps. This library defines the following key:

/tikz/karnaugh (no value)

This key should be passed as an option to a picture or a scope that contains a map, i.e., that calls `\karnaughmap` macro. It will do some internal setups.

`\karnaughmap{`⟨*num var*⟩`}{`⟨*function*⟩`}{`⟨*var list*⟩`}{`⟨*contents*⟩`}{`⟨*decoration*⟩`}`

This macro creates a Karnaugh map of ⟨*num var*⟩ variables for variable ⟨*function*⟩ as a function of the variables listed in ⟨*var list*⟩ for the values given in ⟨*content*⟩ and applying the specified ⟨*decoration*⟩. Any but the first parameter can be empty.

`\karnaughmapvert{`⟨*num var*⟩`}{`⟨*function*⟩`}{`⟨*var list*⟩`}{`⟨*contents*⟩`}{`⟨*decoration*⟩`}`

Similar to `\karnaughmap`, but map will be transposed (like in matrix transposition). See Section 8 for more details.

`\pgfmathdectoGray{`⟨*macro*⟩`}{`⟨*number*⟩`}`

Defines ⟨*macro*⟩ as the result of converting ⟨*number*⟩ from base 10 to Gray coded binary. The idea is to provide a new macro compatible with other base conversion macros described in Section 95.4 Base Conversion [2].

1001 `\pgfmathdectoGray\mynumber{14} \mynumber`

`\kmdectobin{`⟨*number*⟩`}`

Converts ⟨*number*⟩ from base 10 to binary with that number of digits equal to `\kmvarno`.

001001 `\kmvarno=6\relax\kmdectobin{9}`

`\kmdectoKG{`⟨*number*⟩`}`

Converts ⟨*number*⟩ (usually the cell index) to a binary code that resembles Gray code deinterleaving the variables with that number of digits equal to `\kmvarno`, the number of variables in the map. See `kmcell/.style` for an example of application.

010001 `\kmvarno=6\relax\kmdectoKG{9}`

/tikz/every karnaugh (style, initially empty)

The style automatically applied to every Karnaugh map. Can be globally set using `\tikzset`.

\kmindexcounter

A TEX counter for cell index. See `kmcell/.style` for an example of application.

\kmunitlength={⟨*length*⟩}

This length sets the size of an individual cell in the map. Must be set before `karnaugh` is used.

/tikz/disable bars=⟨*boolean*⟩ (default `true`, initially `false`)

Boolean switch that disables the typesetting of all bars and the function identifier. The original intention was to allow for manually crafted American style maps, but since version 1.3 this map style is fully supported so this switch lost its purpose. The initial value is `false` meaning that the bars will be typeset unless they are explicitly disabled.

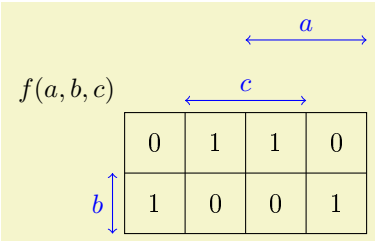
0	1	1	0
1	0	0	1

```
\begin{tikzpicture}[karnaugh,disable bars]
  \karnaughmap{3}{$f(a,b,c)$}{{$a$}{$b$}{$c$}}{0110 0110}{}
\end{tikzpicture}
```

Note that `$f(a,b,c)$` and `{$a$}{$b$}{$c$}` are not used and could be empty.

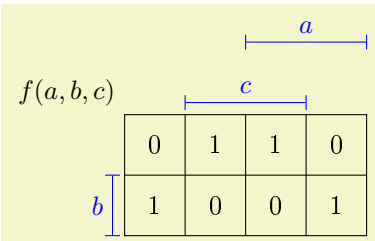
/tikz/kmbar (style, initially `|-|`)

The style used for the top and side bars related to the variables and denoting the rows and columns for which the respective variable is 1. The initial value is `|-|` meaning they all will be represented as a line with T shaped tips.




```
\begin{tikzpicture}[karnaugh,
                    kmbar/.style={blue,<->}]
  \karnaughmap{3}{$f(a,b,c)$}{{$a$}{$b$}{$c$}}{0110 0110}{}
\end{tikzpicture}
```

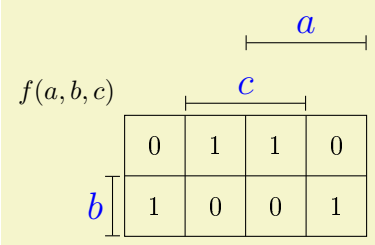
See Chapter 16 Arrows of reference [2] for more arrow options and proper control over arrow appearance. For instance, if you want to draw a bar that ends exactly aligned with the lines inside the map, you can change the style to `|-|,shorten >=-0.2pt,shorten <=-0.2pt`, assuming you are using the default thickness `thin` which implies `line width=0.4pt`. Otherwise, the very and end on the line tip will be aligned with the middle of the internal lines, but someone should have to be very careful to notice a difference of `0.2pt`.




```
\begin{tikzpicture}[karnaugh,
                    kmbar/.style={blue,
                                  |-|,
                                  shorten >=-0.2pt,
                                  shorten <=-0.2pt}]
  \karnaughmap{3}{$f(a,b,c)$}{{$a$}{$b$}{$c$}}{0110 0110}{}
\end{tikzpicture}
```

/tikz/kmbar label (style, initially empty)

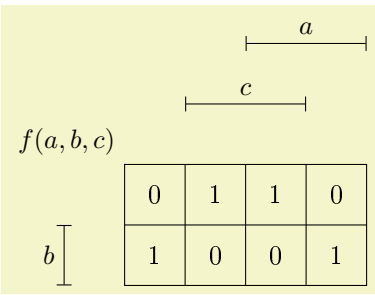
The style used for the variable identifiers on the bars. For American styled maps, it means the style for typesetting the lists of variables associated with columns and rows.




```
\begin{tikzpicture}[karnaugh,
                    kmbar label/.style={blue,font=|Large}]
  \karnaughmap{3}{$f(a,b,c)$}{{$a$}{$b$}{$c$}}{0110 0110}{}
\end{tikzpicture}
```

/tikz/kmbar sep=⟨*width*⟩ (no default, initially `0.2\kmunitlength`)

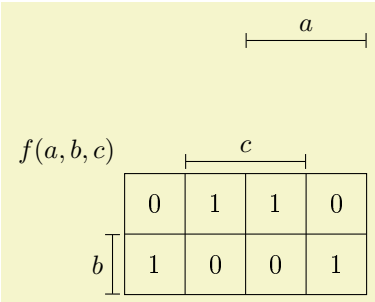
The distance between the bar closer to the map and the map itself. It depends mainly on the line tip used in `kmlines/.style`.




```
\begin{tikzpicture}[karnaugh,
                    kmbar sep=1|kmunitlength]
  \karnaughmap{3}{$f(a,b,c)$}{{$a$}{$b$}{$c$}}{0110 0110}{}
\end{tikzpicture}
```

/tikz/kmbar top sep=⟨*width*⟩ (no default, initially `1\kmunitlength`)

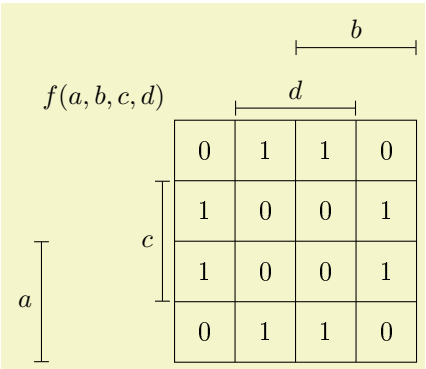
The distance between two bars on top of map. It depends mainly on the font height used in `kmbar label/.style`. For American styled maps, it means the distance between the label centre and the top of the map.




```
\begin{tikzpicture}[karnaugh,
                    kmbar top sep=2|kmunitlength]
  \karnaughmap{3}{$f(a,b,c)$}{{$a$}{$b$}{$c$}}{0110 0110}{}
\end{tikzpicture}
```

/tikz/kmbar left sep=⟨*width*⟩ (no default, initially `1\kmunitlength`)

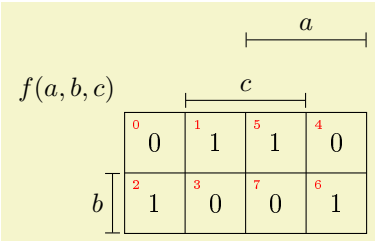
The distance between two bars at the left side of map. It depends mainly on the variable identifier width and the font size used in `kmbar label/.style`. For American styled maps, it means the distance between the label and the left most side of the map.




```
\begin{tikzpicture}[karnaugh,
                    kmbar left sep=2|kmunitlength]
  \karnaughmap{4}{$f(a,b,c,d)$}{{$a$}{$b$}{$c$}{$d$}}%
     {0110 0110 0110 0110}{}
\end{tikzpicture}
```

/tikz/enable indices=⟨*boolean*⟩ (default `true`, initially `false`)

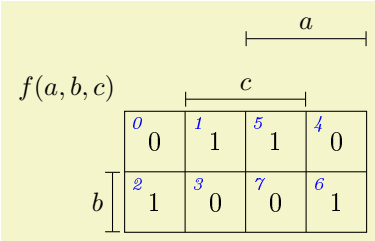
Boolean switch that enables the typesetting of all indices. The index corresponds to the row number in the truth table respective to the cell. The style `kmindex/.style` can modify how the indices are presented and the lengths `\kmindexposx` and `\kmindexposy` where they are placed. Base-10 is the default base for indices, but there are two Boolean switches, `binary index` and `Gray index`, to change the encoding. The initial value is `false` meaning that the indices will not be typeset unless they are explicitly enabled.




```
\begin{tikzpicture}[karnaugh,
                    enable indices]
  \karnaughmap{3}{$f(a,b,c)$}{{$a$}{$b$}{$c$}}{0110 0110}{}
\end{tikzpicture}
```

/tikz/kmindex (style, initially `red,font=\tiny`)

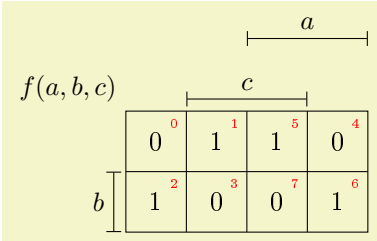
The style used for cell index if enable (see also `enable indices`).




```
\begin{tikzpicture}[karnaugh,
                    enable indices,
                    kmindex/.style={blue,
                                    font=\scriptsize\itshape}]
  \karnaughmap{3}{$f(a,b,c)$}{{$a$}{$b$}{$c$}}{0110 0110}{}
\end{tikzpicture}
```

/tikz/kmindexposx=⟨*dimension*⟩ (no default, initially `0.2\kmunitlength`)

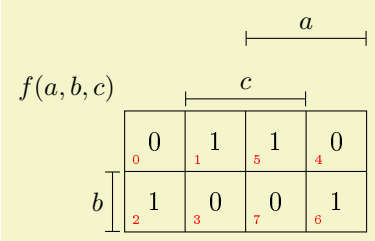
The horizontal distance from the cell left side to the index centre.




```
\begin{tikzpicture}[karnaugh,
                    enable indices,
                    kmindexposx=.8\kmunitlength]
  \karnaughmap{3}{$f(a,b,c)$}{{$a$}{$b$}{$c$}}{0110 0110}{}
\end{tikzpicture}
```

/tikz/kmindexposy=⟨*dimension*⟩ (no default, initially `0.8\kmunitlength`)

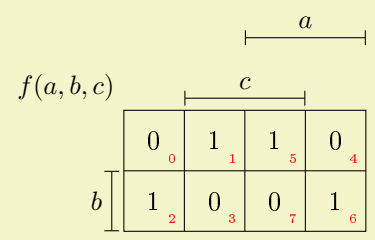
The vertical distance from the cell bottom to the index centre.




```
\begin{tikzpicture}[karnaugh,
                    enable indices,
                    kmindexposy=.2\kmunitlength]
  \karnaughmap{3}{$f(a,b,c)$}{{$a$}{$b$}{$c$}}{0110 0110}{}
\end{tikzpicture}
```

/tikz/kmindexpos={⟨*x coordinate*⟩}{⟨*y coordinate*⟩}

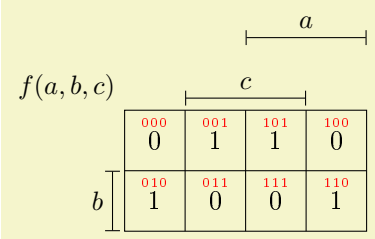
Sets `\kmindexposx` and `\kmindexposy` to `x` and `y` coordinates measured in `\kmunitlength` from the cell bottom left corner.




```
\begin{tikzpicture}[karnaugh,
                    enable indices,
                    kmindexpos={0.8}{0.2}]
  \karnaughmap{3}{$f(a,b,c)$}{{$a$}{$b$}{$c$}}{0110 0110}{}
\end{tikzpicture}
```

/tikz/binary index=⟨*boolean*⟩ (default `true, initially false`)

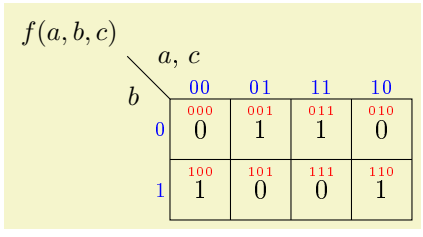
Boolean switch that sets the index presentation to binary code. It is convenient to also set the index coordinates. In the following example, the significance order is a, b and c, meaning, a is the most significant bit and c is the least significant bit. Therefore, the left most bit of the indices is one only in the two left columns below a bar, the middle bit is one in the bottom row as b bar extends and the right most bit is one in the central columns below c bar.




```
\begin{tikzpicture}[karnaugh,
                    enable indices,
                    binary index,
                    kmindexpos={0.5}{0.8}]
  \karnaughmap{3}{$f(a,b,c)$}{{$a$}{$b$}{$c$}}{0110 0110}{}
\end{tikzpicture}
```

/tikz/Gray index=⟨*boolean*⟩ (default `true, initially false`)

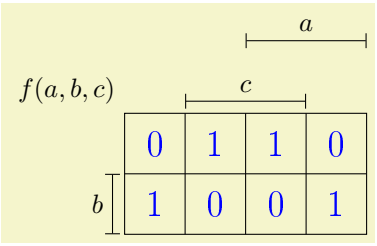
Boolean switch that sets the index presentation to binary code corresponding to the aggregate value of the variables for one cell and not the index of the truth table row respective to the cell. The difference between `Gray index` and `binary index` is that the latter is a binary coded representation of the cell position in the truth table and, because the recursive algorithm that builds the map scrambles (interleaves) the variables, it looks random at first sight. The former, descrambles (deinterleaves) the variables so it would look like a string of zeros and ones associated with row variables followed by one associated with columns in the order they are shown along the map and not in macro parameter #3. It looks like Gray code, that's where its name came from, but in fact it is not. Therefore, `Gray index` is pretty much useless to locate the corresponding row in the truth table. However, it can be used to visualize the stated of variables more easily than binary code. A secondary index can be set, see `kmcell/.style` for an example of how it can be done. Normally, enabling binary or Gray coded indices requires that the index position be adjusted. The initial value of this switch is `false` meaning that the indices will be typeset accordingly to another code, decimal or binary, depending on `binary index` switch.




```
\begin{tikzpicture}[karnaugh,
                    American style,
                    kmbar top sep=0.2\kmunitlength,
                    kmbar left sep=-0.1\kmunitlength,
                    enable indices,
                    Gray index,
                    kmindexpos={0.5}{0.8}]
  \karnaughmap{3}{$f(a,b,c)$}{{$a$}{$b$}{$c$}}{0110 0110}{}
\end{tikzpicture}
```

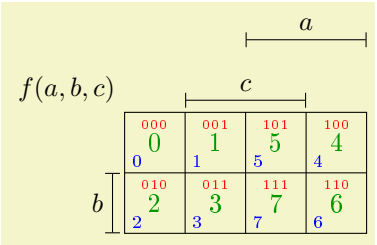
/tikz/kmcell (style, initially empty)

The style used for cell contents.



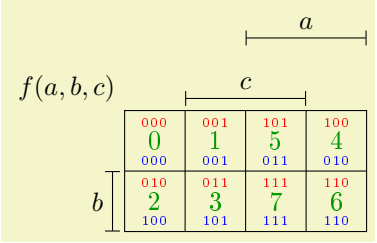

```
\begin{tikzpicture}[karnaugh,
                    kmcell/.style={blue,font=\Large}]
  \karnaughmap{3}{$f(a,b,c)$}{{$a$}{$b$}{$c$}}{0110 0110}{}
\end{tikzpicture}
```

Some interesting applications of `kmcell/.style` involves the cell index given by `\the\kmindexcounter`. You can name every cell for future use or place a label within the cell index just like `enable indices` does. In the following example, `kmcell/.style` is used to place a label within each cell with the decimal value of the cell index and `enable indices` is for the binary value. Moreover, the cell content is also a number that corresponds to the cell index (manually placed) just to show the correlation, namely, the cell index corresponds to the cell position it the parameter #4.




```
\begin{tikzpicture}[karnaugh,
                    enable indices,
                    binary index,
                    kmindexpos={0.5}{0.8},
                    kmcell/.style={green!60!black,
                        label={[font=\scriptsize,blue,
                            label distance=-0.3\kmunitlength]
                            below left:\the\kmindexcounter}}]
  \karnaughmap{3}{$f(a,b,c)$}{{$a$}{$b$}{$c$}}{0123 4567}{}
\end{tikzpicture}
```

So, let's compare the binary index, internally generated by `\kmdectobin`, with pseudo index generated by `\kmdectoKG` when `Gray index` is set:

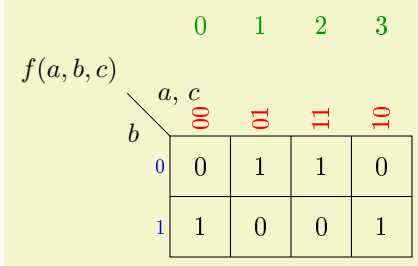



```
\begin{tikzpicture}[karnaugh,
                    enable indices,
                    binary index,
                    kmindexpos={0.5}{0.8},
                    kmcell/.style={green!60!black,
                        label={[font=\tiny,blue,
                            label distance=-0.2\kmunitlength]
                            below:\kmdectoKG{\kmindexcounter}}}]
  \karnaughmap{3}{$f(a,b,c)$}{{$a$}{$b$}{$c$}}{0123 4567}{}
\end{tikzpicture}
```

The binary index at the top in red is the representation of the cell position in binary so it contains the values of *abc*. Pseudo Gray code at the bottom in blue is the deinterleaved values of the cell position placing *b* in the leftmost position, in other worlds, it is just *bac*. This code is convenient to represent the code obtained placing the variables side by side as they appear in the left and top of the map. Curious readers are invited to relate the values indicated by the variable bars with the red and blue codes.

`/tikz/kmtoplabel` (style, initially `blue,font=\footnotesize`)

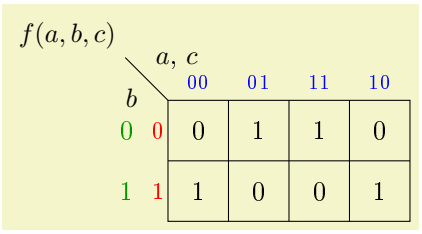
The style used for Gray coded labels at the top of American styled maps. Variable `\x` can be used to express the column position from 0 to `\kmxsize`−1.




```
\begin{tikzpicture}[karnaugh,
                    American style,
                    kmtoplabel/.style={red,font=\small,
                        rotate=90,centered,
                        label={[green!60!black,
                            label distance=2em]right:\x}},
                    kmbar top sep=0.3\kmunitlength,
                    kmbar left sep=-0.1\kmunitlength]
  \karnaughmap{3}{$f(a,b,c)$}{{$a$}{$b$}{$c$}}%
    {0110 0110}{}
\end{tikzpicture}
```

`/tikz/kmleftlabel` (style, initially `blue,font=\footnotesize`)

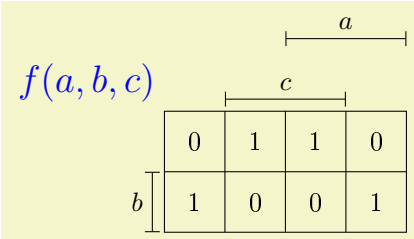
The style used for Gray coded labels at the left side of American styled maps. Variable \y can be used to express the row position from 0 to \kmysize−1.




```
\begin{tikzpicture}[karnaugh,
                    American style,
                    kmleftlabel/.style={red,font=\small,
                        label={[green!60!black]left:\y}},
                    kmbar top sep=0.3\kmunitlength,
                    kmbar left sep=-0.1\kmunitlength]
  \karnaughmap{3}{$f(a,b,c)$}{{$a$}{$b$}{$c$}}%
     {0110 0110}{}
\end{tikzpicture}
```

/tikz/kmvar (style, initially empty)

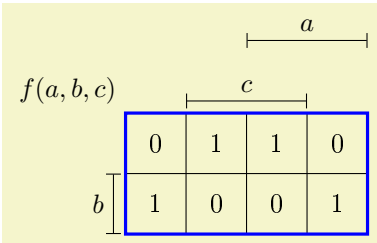
The style used for the variable name (function) of the map.




```
\begin{tikzpicture}[karnaugh,
                    kmvar/.style={blue,font=\Large}]
  \karnaughmap{3}{$f(a,b,c)$}{{$a$}{$b$}{$c$}}{0110 0110}{}
\end{tikzpicture}
```

/tikz/kmbox (style, initially empty)

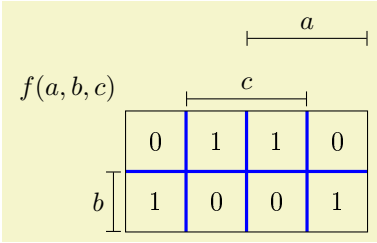
The style used for the box surrounding the map.




```
\begin{tikzpicture}[karnaugh,
                    kmbox/.style={blue,very thick}]
  \karnaughmap{3}{$f(a,b,c)$}{{$a$}{$b$}{$c$}}{0110 0110}{}
\end{tikzpicture}
```

/tikz/kmlines (style, initially empty)

The style used for the lines separating adjacent rows and columns inside the map.

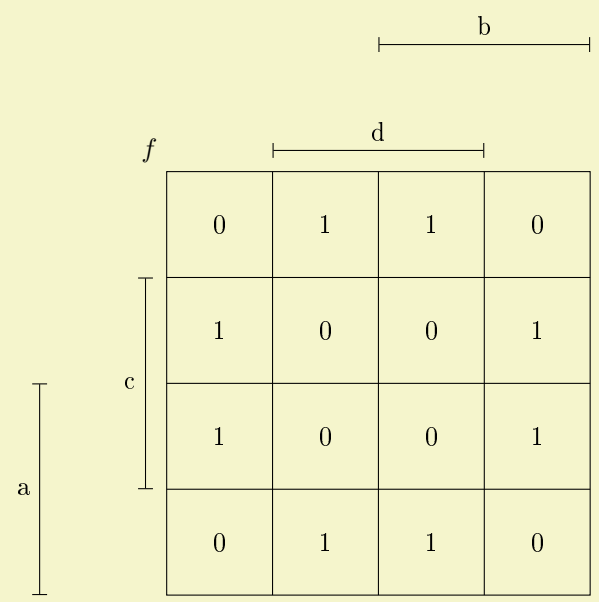



```
\begin{tikzpicture}[karnaugh,
                    kmlines/.style={blue,very thick}]
  \karnaughmap{3}{$f(a,b,c)$}{{$a$}{$b$}{$c$}}{0110 0110}{}
\end{tikzpicture}
```

You can add options to the graphics by setting the `every karnaugh` style which is automatically applied.

4 Adjusting the map size

Possibly the most important feature that you can change is the size of the diagrams and it is done by changing the size of the cells within the map, simply by typing:

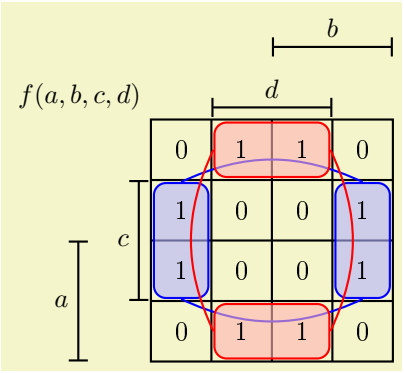



```
\kmunitlength=14mm
\begin{tikzpicture}[karnaugh]
  \karnaughmap{4}{$f$}{abcd}{0110 0110 0110 0110}{}
\end{tikzpicture}
```

The setting of the `\kmunitlength` remains active until you change it again;[6] the default `\kmunitlength` is 8 mm:

5 Marking simplifications

The already mentioned fifth parameter can be used if you want to draw something inside the Karnaugh map. For example, this is useful if you want to show how you simplified a logic function highlighting the prime implicants:




```
\begin{tikzpicture}[karnaugh,x=1|kmunitlength,y=1|kmunitlength,
                    thick,
                    grp/.style n args={3}{#1,fill=#1!30,
                      minimum width=#2|kmunitlength,
                      minimum height=#3|kmunitlength,
                      rounded corners=0.2|kmunitlength,
                      fill opacity=0.6,
                      rectangle,draw}]
  \karnaughmap{4}{$f(a,b,c,d)$}{{$a$}{$b$}{$c$}{$d$}}%
    {0110 0110 0110 0110}%
    {
      \node[grp={blue}{0.9}{1.9}](n000) at (0.5,2.0) {};
      \node[grp={blue}{0.9}{1.9}](n001) at (3.5,2.0) {};
      \draw[blue] (n000.north) to [bend left=25]  (n001.north)
                  (n000.south) to [bend right=25] (n001.south);
      \node[grp={red}{1.9}{0.9}](n100) at (2.0,3.5) {};
      \node[grp={red}{1.9}{0.9}](n110) at (2.0,0.5) {};
      \draw[red]  (n100.west) to [bend right=25] (n110.west)
                  (n100.east) to [bend left=25] (n110.east);
    }
\end{tikzpicture}
```

and the corresponding expression is:

$$f(a,b,c,d) = c\,\bar{d} + \bar{c}\,d$$

where colours were used to relate the subexpression with the prime implicant highlighted on the map.

Instead of LaTeX's graphics macros[7] you can use Ti*k*Z for this purpose. In this example, a new style `grp` was defined in order to draw semi-transparent rectangles with a specified colour, width and height (both given in `\kmunitlength`). The Karnaugh map has its datum at the lower left point *exactly*. The centre point coordinates of those rectangles are specified using the `at` command. The length of a single cell within the Karnaugh map is equal to `\kmunitlength`. Thus, the `x` and `y` units are set to `1\kmunitlength` so the coordinates can be written without the unit and the rectangles will fall in the precise position even if one changes the map size by changing the `\kmunitlength`.

[6] Or, of course, until you leave the group in which you redefined the value.
[7] As in the original package.

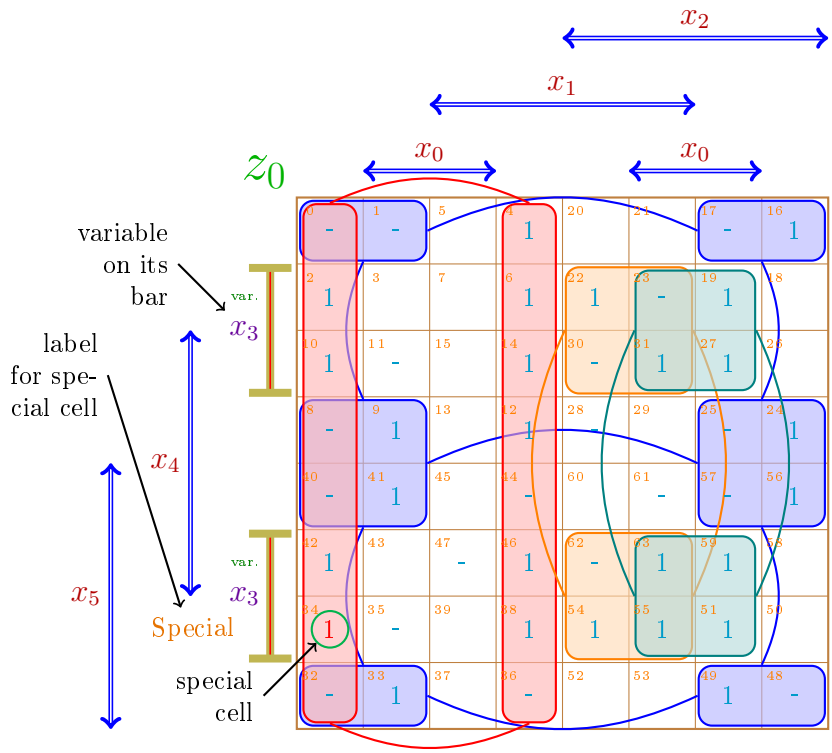
Highlighting the solution can be an arduous and error-prone task. See Section 10 for JQM, a java software that can generate Karnaugh maps with the implicants automatically highlighted and the simplified equation colour coded accordingly to the highlights. You will still need `tikz-karnaugh` package to render the map. By the way, the highlights seem in this section and the next two sections were generated using JQM.

6 Complete example

In this and in the next sections, examples of how individual variables and cell contents can be format are presented. The syntax relies on square brackets (`[]`) to enclose Ti*k*Z features that change the appearance and add more graphs to an individual variable or cell content. Let us see a more interesting and colourful example:

```
\kmunitlength=2.5em
\begin{tikzpicture}[karnaugh,x=1|kmunitlength,y=1|kmunitlength,
    thick,
    grp/.style n args={3}{#1,fill=#1!30,
      minimum width=#2|kmunitlength,
      minimum height=#3|kmunitlength,
      rounded corners=0.2|kmunitlength,
      fill opacity=0.6,
      rectangle,draw},
    kmbar/.style={blue,<->,double=white,semithick},
    kmbar left sep=1.2|kmunitlength,
    kmbar sep=0.4|kmunitlength,
    kmbar label/.style={red!70!black,font=|large},
    kmindex/.style={orange,font=|tiny},
    enable indices,
    kmcell/.style={cyan!80!black},
    kmbox/.style={brown,thick},
    kmlines/.style={brown,thin},
    kmvar/.style={green!70!black,font=|huge},
    lbl/.style={left,align=right,text width=1.5|kmunitlength}]
  \karnaughmap{6}{$z_{0}$}{%
    {$x_{5}$}{$x_{2}$}{$x_{4}$}{$x_{1}$}%
      {[yellow!70!black,name=Nv,|-|,double=red,very thick,
       label={[font=|tiny,green!50!black]above:var.},
       text=blue!60!red]$x_{3}$}%
      {$x_{0}$}}%
    {--1{}1{}1{}-11-1{}1{}1-{}1{}{}1-1-{}1-{}-1-1%
     {[red,name=Nc,label={[name=Nl,orange!90!black,
       label distance=1|kmunitlength]left:Special},
       circle,inner sep=2pt,draw=green!70!blue]1}%
     --{}1{}-11{}-{}1--1{}1{}{}111-{}1{}--1}%
    {
      \node[grp={blue}{1.9}{0.9}](n000) at (1.0,7.5) {};
      \node[grp={blue}{1.9}{0.9}](n002) at (7.0,7.5) {};
      \node[grp={blue}{1.9}{1.9}](n010) at (1.0,4.0) {};
      \node[grp={blue}{1.9}{1.9}](n012) at (7.0,4.0) {};
      \node[grp={blue}{1.9}{0.9}](n030) at (1.0,0.5) {};
      \node[grp={blue}{1.9}{0.9}](n032) at (7.0,0.5) {};
      \draw[blue] (n000.east)  to [bend left=25]  (n002.west)
                  (n010.east)  to [bend left=25]  (n012.west)
                  (n030.east)  to [bend right=25] (n032.west)
                  (n000.south) to [bend right=25] (n010.north)
                  (n002.south) to [bend left=25]  (n012.north)
                  (n010.south) to [bend right=25] (n030.north)
                  (n012.south) to [bend left=25]  (n032.north);
      \node[grp={red}{0.8}{7.8}](n100) at (0.5,4.0) {};
      \node[grp={red}{0.8}{7.8}](n101) at (3.5,4.0) {};
      \draw[red] (n100.north) to [bend left=25]  (n101.north)
                 (n100.south) to [bend right=25] (n101.south);
      \node[grp={orange}{1.9}{1.9}](n200) at (5.0,6.0) {};
      \node[grp={orange}{1.9}{1.9}](n220) at (5.0,2.0) {};
      \draw[orange] (n200.west) to [bend right=25] (n220.west)
                    (n200.east) to [bend left=25]  (n220.east);
      \node[grp={teal}{1.8}{1.8}](n300) at (6.0,6.0) {};
      \node[grp={teal}{1.8}{1.8}](n320) at (6.0,2.0) {};
      \draw[teal] (n300.west) to [bend right=25] (n320.west)
                  (n300.east) to [bend left=25]  (n320.east);
 }
  \draw[<-] (Nv) -- +(-1,1)  node[lbl]{variable on its bar};
  \draw[<-] (Nc) -- +(-1,-1) node[lbl]{special cell};
  \draw[<-] (Nl.120) -- +(-1.1,3.5)  node[lbl]{label for special cell};
\end{tikzpicture}
```

The corresponding Karnaugh map looks like this:



end the logic expression[8] is

$$z = \bar{x}_3\,\bar{x}_1 + \bar{x}_2\,\bar{x}_0 + x_3\,x_2\,x_1 + x_3\,x_2\,x_0.$$

You may notice that the zeros were omitted (replaced by `{}` in the list). Also, the cell 34 is special because `{[red,name=Nc, label={[name=Nl,orange!90!black, label distance=1\kmunitlength ]left: Special}, circle, inner sep=2pt, draw=green!70!blue]1}`. You can put almost anything inside a cell using curly brackets and you can customize the cell style using square brackets. The format is: `{[opt]string}` where `opt` is an optional set of styles (among other Ti*k*Z parameters) which will be passed as the last option of Ti*k*Z command `\node` and `string` will be written inside the cell by that command. To use this syntax, it is imperative that the very first character after the opening curly brackets ({) be the opening square brackets ([). Matching pairs of square brackets are allowed inside the optional sequence provided that they are protected inside a pair of curly brackets. In this case, the proper content of cell 34 is just the number 1 near the end, all the rest is the style applied to this single 1, therefore coded between square brackets. The style uses Ti*k*Z syntax in order to change colour, font size, add a label, add figure, add decoration and name it for future reference. In this case, two nodes are named `Nc` and `Nl` for future reference. Near the Ti*k*Z environment end, those names are used to place arrows pointing to the nodes with a description. The `\draw` command that draws those arrows cannot be placed inside the fifth argument of macro `\karnaughmap` because the fifth argument is typeset before the cell's contents (the fourth argument), therefore no name would be created at the time the fifth argument is typeset.

The variables identifiers (the third argument) can also be formatted individually using style, but note that the custom style will be applied to both the bar line and the node for the variable identifier. If a bar gets segmented, just like x_3 bar, the named node will be the top most if the bar is vertical or the right most if the bar is horizontal. The x_3 bar is different from the other bars because `[yellow!70!black,name=Nv,|-|,double=red, very thick, label={[font=\tiny,green!50!black]above:var.}, text=blue!60!red]` changes its appearance. The node name `Nv` is also not available at the time the fifth argument is typeset. So, any command that makes use of it will need to be placed after the end of macro `\karnaughmap`.

The distance between bars on the left side was set to `1.2\kmunitlength` to prevent overlapping between x_3 (the label) and x_4 bar and x_4 and x_5 bar, but the distance between the bars on top was left unchanged. The distance between the map and the bars closest to it was set to `0.4\kmunitlength` to prevent overlapping between the bar tip ($\Longrightarrow$) and the map itself.

[8] This is not of any importance here, but I couldn't hold myself back. By the way, if you are curious, there are another two minimal solutions.

The indices can be computed by

$$32\,x_5 + 8\,x_4 + 2\,x_3 + 16\,x_2 + 4\,x_1 + 1\,x_0$$

which is a bit bizarre. The truth table values ought to be arranged according to this index order. This bizarreness is the price we pay to have the variables placed in positions which are more intuitive. See Section 10 for a java software that can help on this matter.

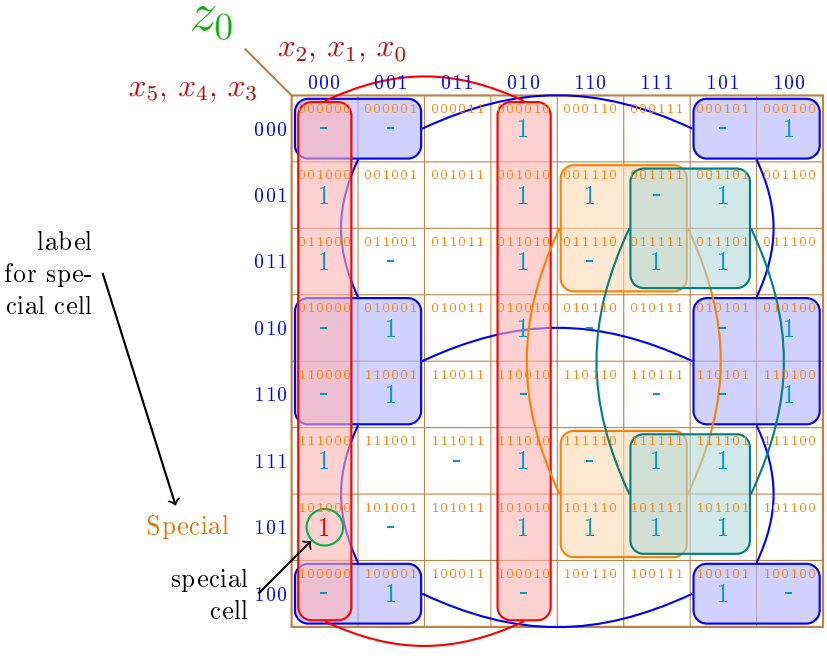
7 American style

Support for American styled map was completed in version 1.3. Although the default style is what Karnaugh calls "simplified labels" [1], it is sufficient to enable the `American style` to obtain a map that resembles the maps found in many books on Digital Electronics and Digital System Design. Besides setting map style to American, you would need to adjust the vertical and horizontal distance between the map itself and the Gray coded labels. This is done through setting `kmbar top sep` and `kmbar left sep`. Using `\tikzset`, all the settings can be done globally, for instance:

```
\tikzset{every karnaugh/.style={
    American style,
    kmbar top sep=0.2\kmunitlength,
    kmbar left sep=-0.1\kmunitlength,
    kmGrayCode/.style={font=\small,purple},
    enable indices, Gray index,
    kmindexpos={0.5}{0.8}}}
```

enables the `American style` and, consequently, disables bars; sets the vertical distance in 20% of the cell size (remember, it is measured from the centre of the node that contains the label to the map boarder); approaches the labels on the left of the map by 10% of the cell size; sets style for the labels; enables indices in each cell; sets the index to in Gray code and sets the index position.

The example of the last section, just deleting some useless code, renders as:



One possible problem is that, depending on the number of variables, one would need to reduce the labels on top of the map to an unreasonable size to prevent them from overlapping, or to enlarger the cell size to make them fit. The same is true for indices inside the cells.

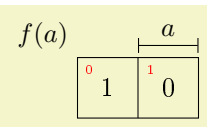
If, instead of `Gray index`, you use `binary index` you would notice they are quite different. This is because binary indices reflect the position of the cell as it was taken from a row of a linear truth table. Since the algorithm that builds the map distributes the variables among the rows and columns of the map in a non-trivial way, the indices depend on the cell position and its relation to the variable input order. Gray coded indices are fix. The decimal indices can still be computed by

$$32\,x_5 + 8\,x_4 + 2\,x_3 + 16\,x_2 + 4\,x_1 + 1\,x_0.$$

If you are struggling with the variable position and the order in which variables and values should appear in order to result in the map you want, see Section 10 for a java software that can help.

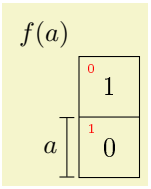
8 Vertical mode

For an odd number of variables, the Karnaugh map is rectangular and macro `karnaughmap` will typeset it twice as wide as it is high (not taking into account the bars). Like this single variable map:



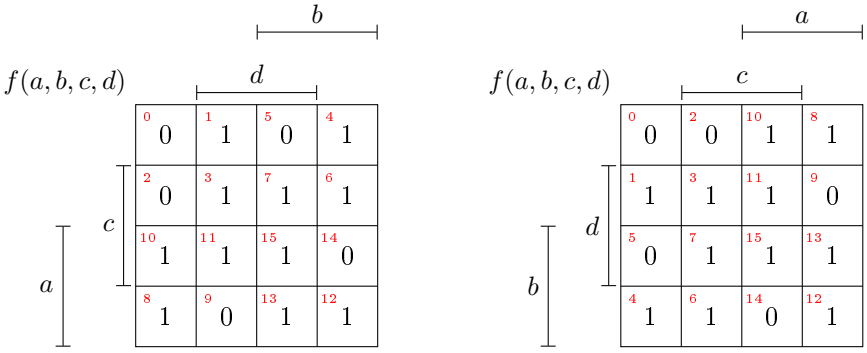

```
\tikz[karnaugh,enable indices]%
  \karnaughmap{1}{$f(a)$}{{$a$}}{10}{};
```

This layout is good for presentations because the projection area is usually wider than higher. Paper sheets, on the other hand, are usually higher than wider, so for a big map you may need something like[9]:




```
\tikz[karnaugh,enable indices]%
  \karnaughmapvert{1}{$f(a)$}{{$a$}}{10}{};
```

This is called, for lack of a better name, vertical mode[10] and it is done by the `karnaughmapvert` macro. Note that `karnaughmapvert` macro arranges the variables in a different order. Compare the two square (four variables) maps below in the normal (on the left) and vertical mode (on the right) paying attention to the indices and variables identifier.



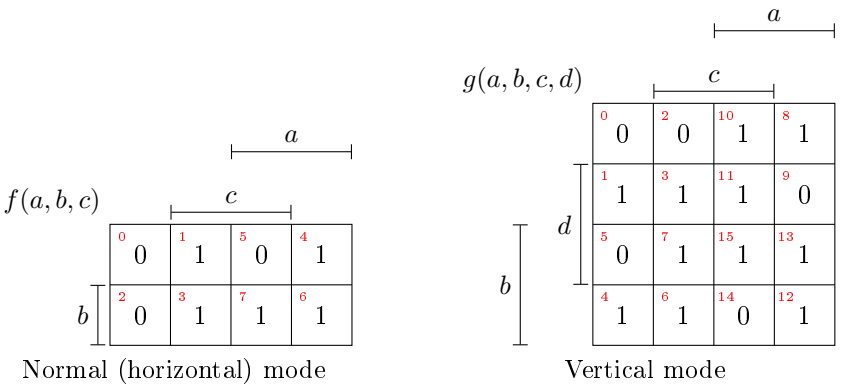
Normal (horizontal) mode Vertical mode

The indices are calculated in the same way, but their position inside the map is different because the variables positions are different. It is like one map is mirrored and then rotated 90^o (mirrored horizontally and rotated clockwise or mirrored vertically and rotated anticlockwise.) Exactly like matrix transposition.

One interesting application of vertical mode is when you want to keep consistency in variable identifier position among maps with odd and even number of variables. For example, if you want the most significant variable a appearing on top of the maps you can use normal (horizontal) mode for maps of odd number of variables and vertical mode for even amounts, like this:

[9] Or you can use landscape.

[10] Not to be confused with TEX vertical mode.



A more general approach is to use the java software described in Section 10 to create maps with arbitrary variables positioning. Suppose that you desire the most significant variable a to appear at the left side of a three variables map. You can do the opposite of what was done in the last example, but you will end up with a vertical map of three variables and maybe it is not what you want. Using the software described in Section 10 allows a to be placed at the left in a normal (horizontal) mode map. This will change the indices because it reorders the truth tablet such that a will no longer be the most significant variable, but without changing the logic function.

9 If you use the original version of the macros...

... you certainly have noticed a number of changes. The most important one is that now you control the appearance of cell, index, etc. by changing the style and not through macros. Also, you need a Ti*k*Z picture environment or inline command.

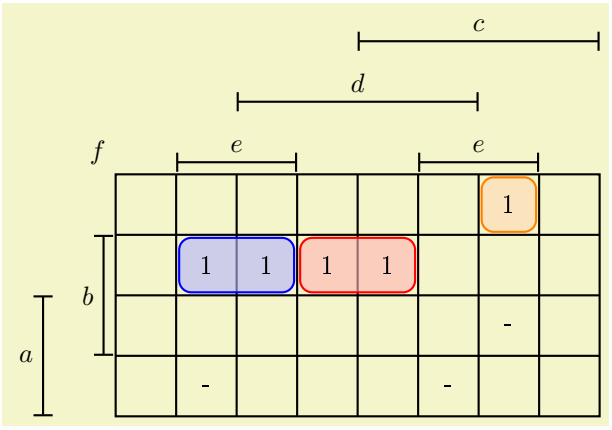
10 Java Quine McCluskey – JQM

Java Quine McCluskey (JQM) implements the Quine McCluskey algorithm with Petrick's Method (or the method of prime implicants) for minimization of Boolean functions. It is available on https://sourceforge.net/projects/jqm-java-quine-mccluskey/. Up to sixteen functions of sixteen variables can be minimized. A graphical interface is provided for entering and editing the truth table that can be saved and loaded.

One very useful feature of this software is that you can reorder the variables on the map to suite your particular application instead of rely exclusively on the macro to scatter your variables around. You can also input a map in text mode and obtain the Ti*k*Z code for creating the corresponding map, and the correspondence would be evident. For instance, a text file like:

```
.k
a,b,c,d,e,,f
0000 0010
0111 1000
0000 00-0
0-00 0-00
```

would render:




```
\tikzset{
  grp/.style n args={3}{
    draw=#1,fill=#1!30,
    minimum width=#2|kmunitlength,
    minimum height=#3|kmunitlength,
    rounded corners=0.2|kmunitlength,
    fill opacity=0.6,
    rectangle}
}

\begin{tikzpicture}[karnaugh,x=1|kmunitlength,
                    y=1|kmunitlength,thick]
  \karnaughmap{5}{$f$}{{$c$}{$a$}{$d$}{$b$}{$e$}}%
    {{}{}{}1{}{}11{}-{}{}{}{}{}%
     {}1{}{}{}{}1{}{}{}{}-{}-{}{}}%
    {
      \node[grp={blue}{1.9}{0.9}] at (2.0,2.5) {};
      \node[grp={red}{1.9}{0.9}] at (4.0,2.5) {};
      \node[grp={orange}{0.9}{0.9}] at (6.5,3.5) {};
    }
\end{tikzpicture}
```

Note that the original text file represents the obtained Karnaugh map very faithfully, but the command that creates the map above, in particular parameters #3 and #4, has no evident relation with the original text or the generated map. This, again, is due to the recursive algorithm that scrambles the variables. For a small map, meaning a small number of variables, it would not be too difficult to manually sort the parameters in order to obtain the desire result, but it would be simply too problematic for a bigger map. That's where JQM comes in. You can manipulate the map assigning the position of the variables in any order and the software automatically sorts the truth table accordingly. The order in which the variables will appear in the final expression is also independent on the order they appear in the map.

JQM main features:

- Up to 16 input variables.
- Up to 16 functions (output variables).
- Petrick's Method used to find solutions covered by non-essential prime implicant.
- Comfortable graphical interface allows variable renaming and column reorder.
- Truth table can be saved in CSV file for external editing or reuse. Then it can be loaded again. Also, one can generate the truth table using other software and "import" (open) for edition and minimization.
- Besides using the graphical interface, truth table can be written in a text file then load in the software. Several formats are available including: list decimal representing implicants and don't care, Karnaugh map and CSV with wildcards.
- Results can be expressed in several formats like: conventional Boolean expression, LaTeX, Structured Text (ST) and Ladder Diagram (LD).
- Results can be exported to HTML file and open in an internet browser.
- Because this software aims PLC programming, solutions are independent (non-simultaneous). It would not be too difficult to modify the algorithm to support simultaneous solution.
- Generates Karnaugh maps in HTML and LaTeX.
- Not only solves the problem, but also shows how the solution was obtained.
- To use the software just download the `zip` file, `unzip` it and double click on "JQM-QuineMcCluskey.jar". Please see ReadMe.txt (or LeiaMe.txt if you prefer Portuguese) for more details and examples.

11 Final remarks

This is not even nearly all you need to know about the usage of these macros, but it is a good start. In case you find a bug, or if you have comments or suggestions, please send me an e-mail.

The maximum size map I could produce was a Karnaugh map with 12 variables; with bigger maps I only exceeded TeX's main memory. This is due to the macros' recursive algorithm. Quite likely you will exceed TeX's capacity with even smaller maps if they occur in large documents.

References

[1] **Karnaugh**, Maurice. The map method for synthesis of combinational logic circuits, *Transactions of the American Institute of Electrical Engineers, Part I: Communication and Electronics* 72(5), 593–599, 1953.

[2] **Tantau**, Till. *The TikZ and PGF Packages* : Manual for version 3.1.9a-34-ga0d9cada, https://github.com/pgf-tikz/pgf, 2021.