Typesetting Karnaugh Maps with LaTeX and Ti*k*Z

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

Version 1.1 of 10 January 2018

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. 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.

2 Introduction

Karnaugh maps 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 packages available I know and those are:

[1] The actual limit may be different for you.

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[2] 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[3] (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 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 can be mimic 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.

With `tikz-karnaugh` you can typeset big (up to twelve variables or 4096 cells) good looking maps. Using a java software, 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

[2] It works until you blow the memory out which will happen about ten to twelve variables.

[3] Those bars have been underappreciated along the history. Karnaugh (1953) 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.

of you 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

Index	a	b	c	d	f
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:

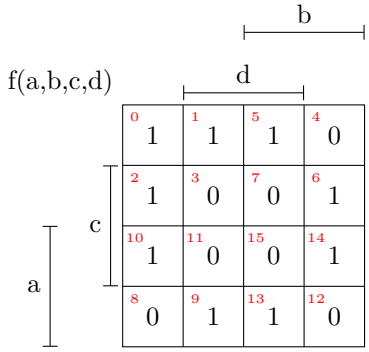
1. the number of variables in the map;
2. an identifier for the function;
3. a list of variable identifiers for the variables;
4. the list of values of f for each line in the truth table; and
5. a possibly empty set of Ti*k*Z commands that will be drown before the function values so the values will appear 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}{};
```

This produces the following Karnaugh map, where the indices in the upper left corner of each cell correspond to the indices in the truth table:[4]

[4] The indices can easily be calculated from the variable value in the truth table, e.g., row 11: the index equals $2^3\,a + 2^2\,b + 2^1\,c + 2^0\,d = 8\,a + 4\,b + 2\,c + 1\,d = 8 + 2 + 1 = 11$.

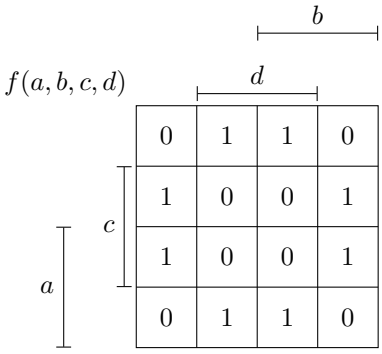


The macros that read the variable list and the list of logic values (i.e., parameters #3 and #4) work recursively.

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}
```

Here, a Ti*k*Z environment was used so there is no semicolon (;) in the end of `\karnaughmap` macro. Also, the indices were omitted by removing `,enable indices` from the options list. This produces the following Karnaugh map (observe that the labels are all in math mode):



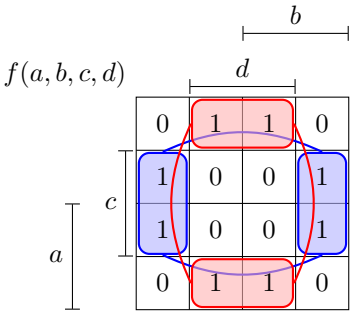
3 Marking simplifications

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:

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

```
\kmunitlength=2em
\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}
```

The corresponding Karnaugh map looks like this:



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[6] 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`).

[6] As in the original package.

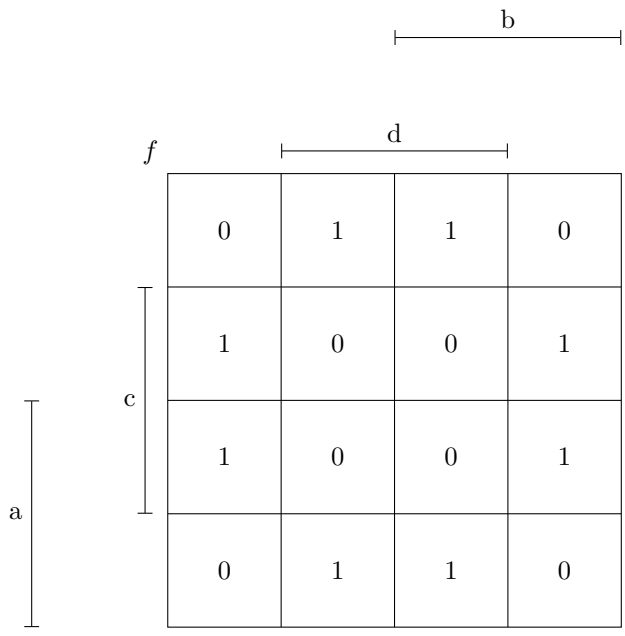
The Karnaugh map has its datum at the lower left point *exactly*. The centre point coordinates of those rectangles are specified using the `at` command. The length of a single cell within the Karnaugh map is equal to `\kmunitlength`. Thus, the `x` and `y` units are set to `1\kmunitlength` so the coordinates 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`.

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=15mm
\begin{tikzpicture}[karnaugh]
   \karnaughmap{4}{$f$}{abcd}{0110 0110 0110 0110}{}
\end{tikzpicture}
```

This results in the following Karnaugh map. The setting of the `\kmunitlength` remains active until you change it again;[7] the default `\kmunitlength` is 8 mm:



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

5 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.

6 Dimensions, styles and switches

The appearance of the Karnaugh map is controlled by some styles, dimensions and switches as follows:

kmbar/.style 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 default value is `black,|-|,thin` meaning they all will be represented as a black thin line with T chapped tips.

kmbar label/.style style used for the variable identifiers on the bars. The default value is `black`.

kmvar/.style style used for the variable name (function) of the map. The default value is `black`.

kmindex/.style style used for cell index. The default value is `red,font=\tiny` meaning they all will typeset in red tiny font if enable (see `enable indices`).

kmcell/.style style used for cell contents. The default value is `black`.

kmbox/.style style used for the box surrounding the map. The default value is `thin`.

kmlines/.style style used for the lines separating adjacent rows and columns. The default value is `thin`.

bar sep distance between the bar closer to the map and the map itself. It depends mainly on the line tip used in `kmlines/.style`. The default value is `0.2\kmunitlength`.

kmbar top sep distance between two bars on top of map. It depends mainly on the font height used in `kmbar label/.style`. The default value is `1\kmunitlength`.

kmbar left sep 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`. The default value is `1\kmunitlength`.

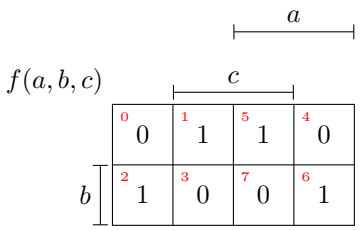
enable indices boolean switch that enables the typesetting of all indices. The default is `false` meaning that the indices will not be typeset unless they are enabled.

disable bars boolean switch that disables the typesetting of all bars and the function identifier. Usable when you want an American style map. The default is `false` meaning that the bars will be typeset unless they are disabled.

For more on styles, have a look in the Ti*k*Z documentation.

One feature that you can switch on is the indices inside the map (like in the first example) by typing:

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



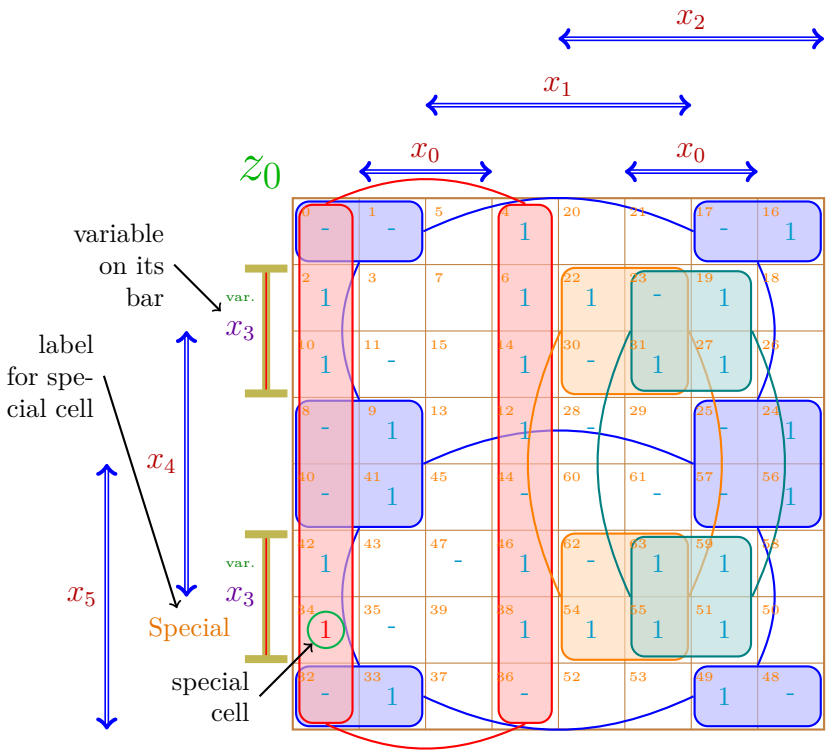
The font size of the map's contents and indices should be set to suitable values (usually `\tiny` for `kmindex/.style` and `\normalsize` for `kmcell/.style`). Those sizes, of course, can be adjusted as needed in agreement with the cell size controlled by `\kmunitlength`.

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},
    bar left sep=1.2\kmunitlength,
    bar 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 beause `{[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 de 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

[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 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 cells 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. If you want an American style map you can use `bar sep` to leave space for the Gray coded numbers.

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 9 for a java software that can help on this matter.

7 American style

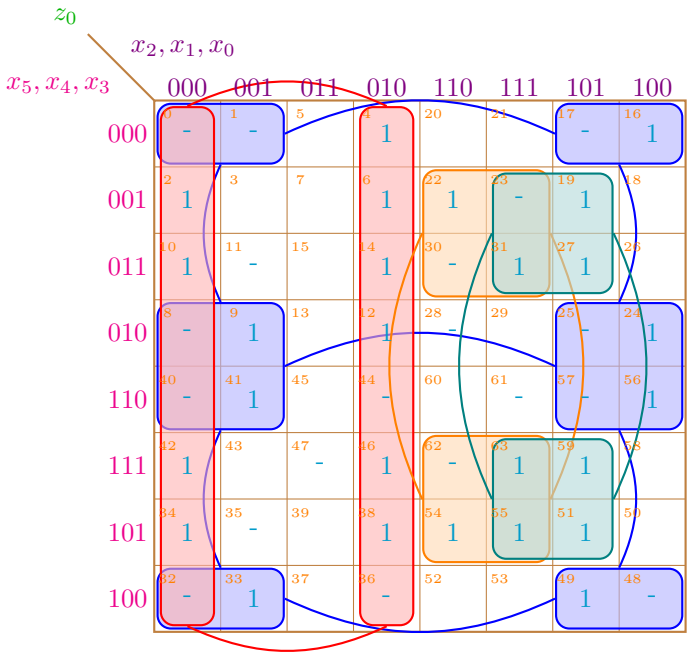
If you really want an American style map and you are not afraid of admitting it publicly, you can still use this package to typeset it. The first thing to do is to disable the bars and the function identifiers. Therefore, this option has to be included in the Ti*k*Z environment: `disable bars`.

Then you will need rows and columns labels in Gray code and a caption for the map and variables identifiers. In the last example, these can be achieved by appending the following code in the fifth argument of the `karnaughmap` macro:

```
\draw[kmbox] (0,8) --
```

```
      node[below left,magenta]{$x_5,x_4,x_3$}
      node[above right,violet]{$x_2,x_1,x_0$} +(-1,1)
      node[above left,green!70!black] {$z_0$};
 \foreach \x/\l in %
  {0/000,1/001,2/011,3/010,4/110,5/111,6/101,7/100} {
   \node[violet] at (\x+0.5,8.2) {\l};
   \node[magenta] at (-0.4,7.5-\x) {\l};
}
```

The result should be:



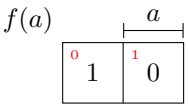
Note, however, that the index inside a cell does not match the Gray code value of the respective row and column[9]. The indices can still be computed by

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

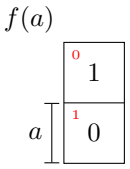
8 Vertical mode

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:

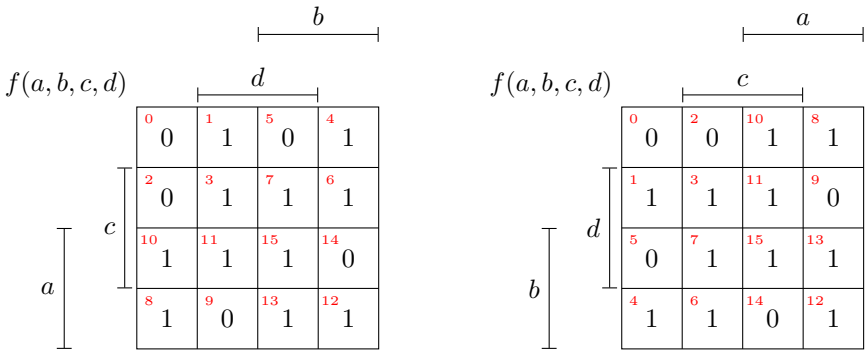
[9] Do not use a Gray code table to compute the index.



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[10]:



This is called, for lack of a better name, vertical mode[11] 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.

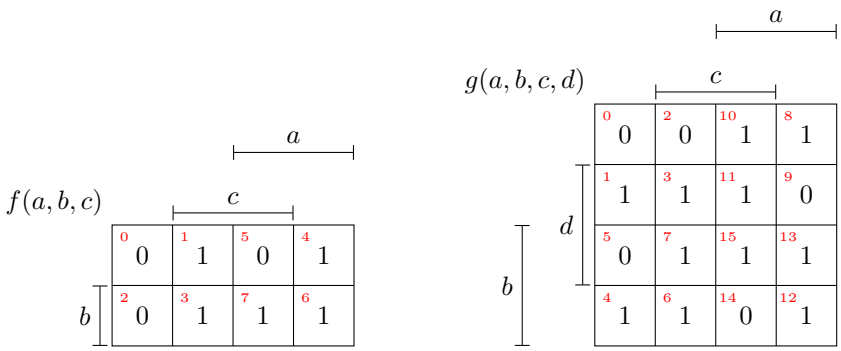


The indices are calculated in the same way, but their position inside the map are different because the variables positions are different. It is like one map is mirrored and then rotated 90^o (mirrored horizontally and rotated clockwise or mirrored vertically and rotated counterclockwise.) Exactly like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

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:

[10] Or you can use landscape.

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



A more general approach is to use the java software described in Section 9 to create maps with arbitrary variables positioning. Suppose that you desire the most significant variable a to appear at the left side of the 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 9 allows a to be placed at the left in a normal (horizontal) mode map, but it changes the indices because it reorder the truth tablet such that a will no longer be the most significant variable, but without changing the logic function.

9 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.

If you need help to typeset Karnaugh maps with or without the prime implicants highlighted, you can try `JQM - Java Quine McCluskey` for minimization of Boolean functions available on `https://sourceforge.net/projects/jqm-java-quine-mccluskey/`. It can generate the solution and create the corresponding map based on a given truth table. One very useful feature of this software is that you can reorder de variables on the map to suite your particular application instead of rely exclusively on the macro to scatter your variables around.