Typesetting Karnaugh Maps with LaTeX and Ti*k*Z

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

Version 1.0 of 22 December 2017

1 Abstract

Karnaugh maps are used to simplify logic equations leading to the most compact expression of two level 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, 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(s). The macros also allow to mark the simplifications of your logic function directly within a Karnaugh map. These 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 high 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 software, including its documentation, is based on Andreas W. Wieland's previous work and the author wished 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. `karnaugh`: it is 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 (well, it works until you blow the memory out which will happen about ten to twelve variables) 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 (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 point out groups (prime implicants) and, although it is possible, it is rather difficult and the results is not very good because they always look a bit of. 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, to point out 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 for four variables. Also, the order in with 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 code on the top and left side of the map. This behaviour can be mimic with `tikz-karnaugh`, 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 point out 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.

2.2 Introductory example

Let us start first with an introduction on how to use these macros: The first thing you have to do is to put the macro file `tikz-karnaugh.tex` into a directory where TeX will find it, i.e., you have to put it into a directory where your `TEXINPUTS` environment variable points to. You can then load them by typing `\input tikz-karnaugh` somewhere in the preamble of your document. Also, you are going to need Ti*k*Z. For this type `\usepackage{tikz}`.

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

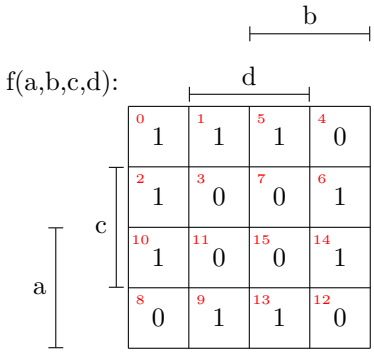
This logic function can easily be put into a Karnaugh map by using the `\karnaughmap` macro in a Ti*k*Z environment (`\begin{tikzpicture}`) or inline command (`\tikz`). It has five parameters:

1. the number of variables in the map;
2. an identifier for the function;
3. a list of variable identifiers for the variables; and
4. the list of values of f for each line in the truth table; and
5. a 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,kmindex/.style={red,font=\tiny}]%
  \karnaughmap{4}{f(a,b,c,d):}{abcd}{1110011001100110}{};
```

This produces the following Karnaugh map, where the indices in the upper left corner of each box correspond to the indices in the truth table:[1]



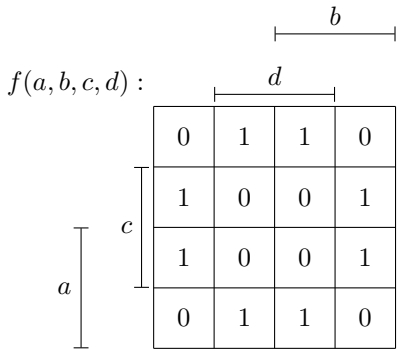
[1] The indices can easily be calculated from the variable values in the truth table, i.e. line 11: the index equals $2^3\,a + 2^2\,b + 2^1\,c + 2^0\,d = 8\,a + 4\,b + 2\,d + 1\,d = 8 + 2 + 1 = 11$.

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, otherwise – like a variable identifier enclosed in $s – you have to put it into braces:

```
\begin{tikzpicture}[karnaugh]
   \karnaughmap{4}{$f(a,b,c,d):$}{{$a$}{$b$}{$c$}{$d$}}%
     {0110011001100110}{}
\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 `,kmindex/.style={red,font=\tiny}` from the options list. This produces the following Karnaugh map:



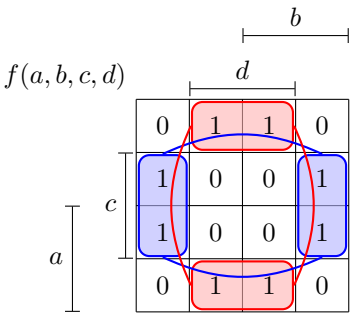
3 Marking simplifications

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 pointing out the prime implicants:

```
\kmunitlength=2em
\begin{tikzpicture}[karnaugh,x=1\kmunitlength,y=1\kmunitlength,
  thick,grp/.style={rounded corners=0.2\kmunitlength,
  fill opacity=0.6,rectangle,draw}]
  \karnaughmap{4}{$f(a,b,c,d)$}{{$a$}{$b$}{$c$}{$d$}}%
    {0110011001100110}%
    {
      \node[grp,color=blue,fill=blue!30,
          minimum width=0.9\kmunitlength,
          minimum height=1.9\kmunitlength] (n000) at (0.5,2.0) {};
      \node[grp,color=blue,fill=blue!30,
          minimum width=0.9\kmunitlength,
          minimum height=1.9\kmunitlength] (n001) at (3.5,2.0) {};
```

```
      \node[grp,color=red,fill=red!30,
          minimum width=1.9\kmunitlength,
          minimum height=0.9\kmunitlength] (n100) at (2.0,3.5) {};
      \node[grp,color=red,fill=red!30,
          minimum width=1.9\kmunitlength,
          minimum height=0.9\kmunitlength] (n110) at (2.0,0.5) {};
      \draw[color=blue] (n000.north) to [bend left=25] (n001.north)
                        (n000.south) to [bend right=25] (n001.south);
      \draw[color=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:

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

Instead of LaTeX's graphics macros[2] you can use Ti*k*Z for this purpose. The Karnaugh map has its datum at the lower left point exactly. The length of a single cell within the Karnaugh map is equal to `\kmunitlength`.

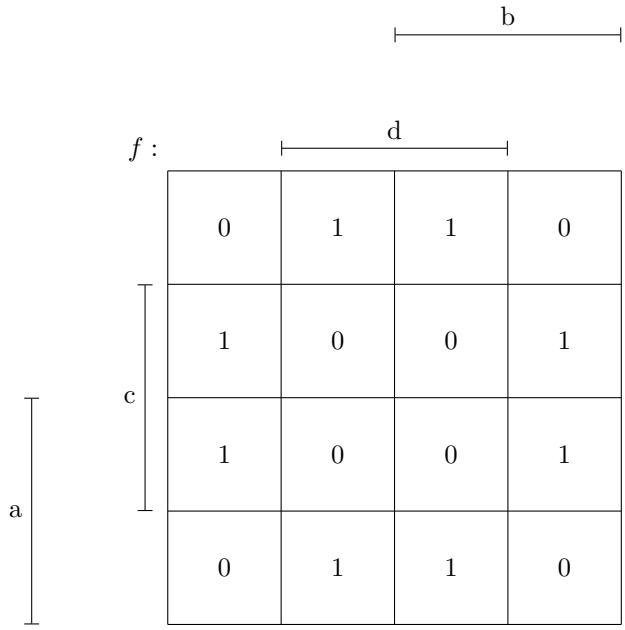
4 Other features

There are some more features that you can use. Possibly the most important is that you can change the size of the diagrams by changing the size of the boxes within the map, simply by typing:

```
\kmunitlength=15mm
\begin{tikzpicture}[karnaugh]
   \karnaughmap{4}{$f:$}{abcd}{0110011001100110}{}
\end{tikzpicture}
```

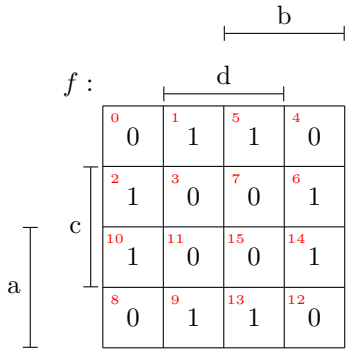
[2] As in the original software.

This results in the following Karnaugh map. The setting of the `\kmunitlength` remains active until you change it again;[3] the default `\kmunitlength` is 8 mm:



Another feature is that you can switch on the indices inside the map like in the fisrt example by typing:

```
\begin{tikzpicture}[karnaugh,kmindex/.style={red,font=\tiny}]
  \karnaughmap{4}{$f:$}{abcd}{0110011001100110}{}
\end{tikzpicture}
```



Usually, the font size of the map's contents and indices should be suitable (`\tiny` for `kmindex/.style` and `\normalsize` for `kmcell/.style`). Those sizes, of course, can be adjusted.

[3] 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 will 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.

6 Dimensions and styles

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

kmbar/.style 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 think line with T chapped tips.

kmbar label/.style style used for the variable name 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 `transparent` meaning they all will be omitted (invisible to be precise).

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

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

bar sep distance between the bar closer to the map and 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 the 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 on the left of map. It depends mainly on the variable name which and the font size used in `kmbar label/.style`. The default value is `1\kmunitlength`.

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

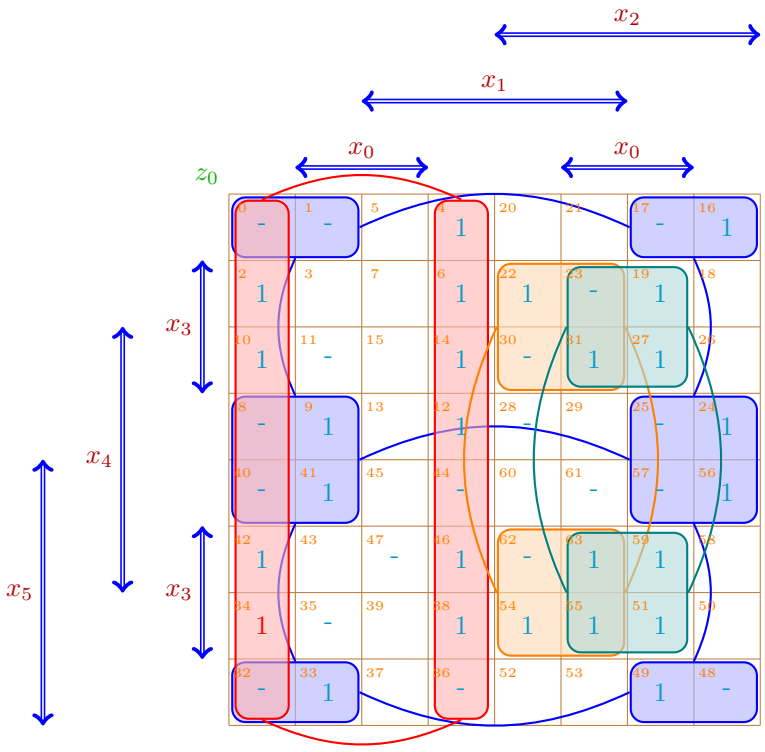
Let us see a more interesting and colourful example.

```
\kmunitlength=2.5em
\begin{tikzpicture}[karnaugh,x=1\kmunitlength,y=1\kmunitlength,
    thick,grp/.style={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},
  kmindex/.style={orange,font=\tiny},
  kmcell/.style={cyan!80!black},
  kmlines/.style={brown,thin},
  kmvar/.style={green!70!black}]
\karnaughmap{6}{$z_{0}$}{%
  {$x_{5}$}{$x_{2}$}{$x_{4}$}{$x_{1}$}{$x_{3}$}{$x_{0}$}}%
  {--1{}1{}1{}-11-1{}1{}1-{}1{}{}1-1-{}1-{}-1-1{\color{red}1}%
   --{}1{}-11{}-{}1--1{}1{}{}111-{}1{}--1}%
  {
    \node[grp,color=blue,fill=blue!30,
      minimum width=1.9\kmunitlength,
      minimum height=0.9\kmunitlength](n000) at (1.0,7.5) {};
    \node[grp,color=blue,fill=blue!30,
      minimum width=1.9\kmunitlength,
      minimum height=0.9\kmunitlength](n002) at (7.0,7.5) {};
    \node[grp,color=blue,fill=blue!30,
      minimum width=1.9\kmunitlength,
      minimum height=1.9\kmunitlength](n010) at (1.0,4.0) {};
    \node[grp,color=blue,fill=blue!30,
      minimum width=1.9\kmunitlength,
      minimum height=1.9\kmunitlength](n012) at (7.0,4.0) {};
    \node[grp,color=blue,fill=blue!30,
      minimum width=1.9\kmunitlength,
      minimum height=0.9\kmunitlength](n030) at (1.0,0.5) {};
    \node[grp,color=blue,fill=blue!30,
      minimum width=1.9\kmunitlength,
      minimum height=0.9\kmunitlength](n032) at (7.0,0.5) {};
    \draw[color=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,color=red,fill=red!30,
      minimum width=0.8\kmunitlength,
      minimum height=7.8\kmunitlength](n100) at (0.5,4.0) {};
    \node[grp,color=red,fill=red!30,
      minimum width=0.8\kmunitlength,
      minimum height=7.8\kmunitlength](n101) at (3.5,4.0) {};
    \draw[color=red] (n100.north) to [bend left=25] (n101.north)
```

```
      (n100.south) to [bend right=25] (n101.south);
  \node[grp,color=orange,fill=orange!30,
    minimum width=1.9\kmunitlength,
    minimum height=1.9\kmunitlength](n200) at (5.0,6.0) {};
  \node[grp,color=orange,fill=orange!30,
    minimum width=1.9\kmunitlength,
    minimum height=1.9\kmunitlength](n220) at (5.0,2.0) {};
  \draw[color=orange] (n200.west) to [bend right=25] (n220.west)
     (n200.east) to [bend left=25] (n220.east);
  \node[grp,color=teal,fill=teal!30,
    minimum width=1.8\kmunitlength,
    minimum height=1.8\kmunitlength](n300) at (6.0,6.0) {};
  \node[grp,color=teal,fill=teal!30,
    minimum width=1.8\kmunitlength,
    minimum height=1.8\kmunitlength](n320) at (6.0,2.0) {};
  \draw[color=teal] (n300.west) to [bend right=25] (n320.west)
     (n300.east) to [bend left=25] (n320.east);
 }
\end{tikzpicture}
```

The corresponding Karnaugh map looks like this:



You may notice that the zeros were omitted (replaced by `{}` in the list). Also, in the cell 34 the one is red beause `{\color{red}1}`. You can put almost anything inside a cell using curly brackets. 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 8 for a software that can help on this matter.

7 American style

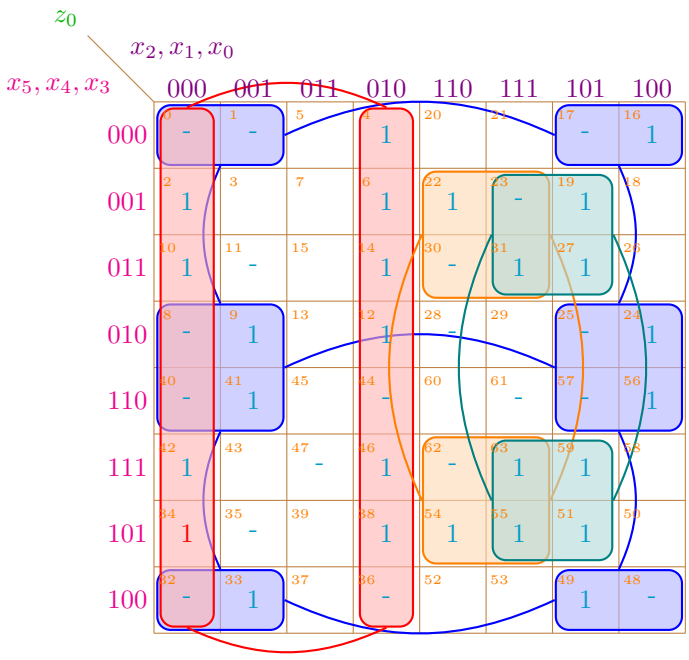
If you really want an American style map and you are not afraid of admitting it publicly, you can still use this package to typeset it. The first thing to do is to disable the bars and the function identifiers by making them transparent. Therefore, these options have to be included in the Ti*k*Z command:

```
kmbar/.style={transparent},
kmvar/.style={transparent}
```

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

```
\draw[brown,thin] (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 math the Gray code value of the respective row and column. 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 Final remarks

This is not nearly all you need to know about the usage of these macros. 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 the marking for the solution on it 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.