COMS10015 quick(ish) reference hand-out: Karnaugh map

An algorithm

Given the truth table for some n-input, 1-output Boolean function as input, application of a Karnaugh map proceeds as follows

1. Draw a rectangular ($p \times q$)-element grid, st.

   (a) $p \equiv q \equiv 0 \pmod 2$, and
   (b) $p \cdot q = 2^n$

   and each row and column represents one input combination; order rows and columns according to a **Gray code**.
2. Fill the grid elements with the output corresponding to inputs for that row and column.
3. Cover rectangular groups of adjacent 1 elements which are of total size 2^m for some m; groups can "wrap around" edges of the grid and overlap.
4. Translate each group into one term of an SoP form Boolean expression e where

   (a) *bigger* groups, and
   (b) *less* groups

   mean a simpler expression.

noting that

- We *only* ever have to consider an n-input, 1-output Boolean function, because we can decompose more complicated examples (i.e., n-input, m-output).
- For $n \geq 5$, we must alter this basic algorithm so it will produce correct output. Partly as a result, and partly since those functions a) get more complex but b) we see few (if any) cases where they are required, in COMS10015 you can assume $1 < n < 5$.
- Use of a Gray code simply means we reorder the intuitive labelling of rows and columns, i.e., instead of the left-hand side below we use the right-hand side:

<table>
<tr><td colspan="2" align="center">integer
sequence</td><td colspan="2" align="center">Gray code
sequence</td></tr>
<tr><td>$0_{(10)}$</td><td>$\mapsto \quad 000_{(2)}$</td><td>$0_{(10)}$</td><td>$\mapsto \quad 000_{(2)}$</td></tr>
<tr><td>$1_{(10)}$</td><td>$\mapsto \quad 001_{(2)}$</td><td>$1_{(10)}$</td><td>$\mapsto \quad 001_{(2)}$</td></tr>
<tr><td>$2_{(10)}$</td><td>$\mapsto \quad 010_{(2)}$</td><td>$3_{(10)}$</td><td>$\mapsto \quad 011_{(2)}$</td></tr>
<tr><td>$3_{(10)}$</td><td>$\mapsto \quad 011_{(2)}$</td><td>$2_{(10)}$</td><td>$\mapsto \quad 010_{(2)}$</td></tr>
<tr><td>$4_{(10)}$</td><td>$\mapsto \quad 100_{(2)}$</td><td>$6_{(10)}$</td><td>$\mapsto \quad 110_{(2)}$</td></tr>
<tr><td>$5_{(10)}$</td><td>$\mapsto \quad 101_{(2)}$</td><td>$7_{(10)}$</td><td>$\mapsto \quad 111_{(2)}$</td></tr>
<tr><td>$6_{(10)}$</td><td>$\mapsto \quad 110_{(2)}$</td><td>$5_{(10)}$</td><td>$\mapsto \quad 101_{(2)}$</td></tr>
<tr><td>$7_{(10)}$</td><td>$\mapsto \quad 111_{(2)}$</td><td>$4_{(10)}$</td><td>$\mapsto \quad 100_{(2)}$</td></tr>
<tr><td align="center">⋮</td><td></td><td align="center">⋮</td><td></td></tr>
</table>

  The reason for this is to ensure the label for adjacent rows (resp. columns) only differs in *one* bit; this allows the Karnaugh map to deliver an optimised expression, by allowing certain forms of group to be formed which would otherwise be disallowed.
- Rather than write the binary values themselves, we used a short-hand to avoid the font becoming too small and/or the diagram becoming too cluttered: a bar above (resp. to the left of) a column and (resp. a row) indicates where the associated variable is 1, with the absence of a bar indicating it is 0.
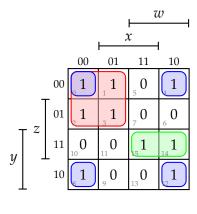
An example

Consider the 4-input, 1-output Boolean function

$$r = f(w, x, y, z)$$

---

described by the truth table below:

w	x	y	z	r
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

By applying the algorithm, the corresponding Karnaugh map results in an expression

$$
\begin{aligned}
r = (\quad \neg w \quad\quad\quad \wedge \quad \neg y \quad\quad\quad) \vee \\
(\quad\quad w \quad\quad\quad \wedge \quad y \quad \wedge \quad z \quad) \vee \\
(\quad\quad\quad\quad \neg x \quad\quad\quad \wedge \quad \neg z \quad)
\end{aligned}
$$

because:

- The red group spans columns 0 and 1 and rows 0 and 1; provided $w = 0$ and $y = 0$ we specify *just* those cells, so the expression is $\neg w \wedge \neg y$. That is, $w = 0$ restricts us to columns 0 and 1 (columns 2 and 3 have $w = 1$) and $y = 0$ restricts us to rows 0 and 1 (rows 2 and 3 have $y = 1$). Note that the values of x and z don't matter: cells in the group hold the value 1 regardless of x and y.
- The green group spans columns 2 and 3 in row 2; provided $w = 1$, $y = 1$ and $z = 1$ we specify *just* those cells, so the expression is $w \wedge y \wedge z$. That is, $w = 1$ restricts us to columns 2 and 3 (columns 0 and 1 have $w = 0$) and $y = 1$ and $z = 1$ restricts us to row 2 (rows 0, 1 and 3 have at least one of $y = 0$ or $z = 0$). Note that the value of x doesn't matter: cells in the group hold the value 1 regardless of x.
- The blue group spans columns 0 and 3 and rows 0 and 3; provided $x = 0$ and $z = 0$ we specify *just* those cells, so the expression is $\neg x \wedge \neg z$. That is, $x = 0$ restricts us to columns 0 and 3 (columns 1 and 2 have $x = 1$) and $z = 0$ restricts us to rows 0 and 3 (rows 1 and 2 have $z = 1$). Note that the values of w and y don't matter: cells in the group hold the value 1 regardless of w and y.