CS226 Quiz 1 (Spring 2016)

Feb 16, 2016 **Time: 45 mins**

- *Be brief, complete and stick to what has been asked.*
- *Unless asked for explicitly, you may cite results/proofs covered in class without reproducing them.*
- *If you need to make any assumptions, state them clearly.*
- ***Please start writing your answer to each sub-question on a fresh page. DO NOT write answers to multiple sub-questions on the same page.***
- ***The use of internet enabled devices is strictly prohibited. You will be debarred from taking the examination if you are found accessing the internet during the examination.***
- ***Please do not engage in unfair or dishonest practices during the examination. Anybody found indulging in such practices will be referred to the D-ADAC.***

1. *[10 + 10 marks]* Write as simple Boolean functions as you can from the following K-maps.

(a)

X,Y→ / Z,W↓	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	0	1	0	1
10	1	1	1	1

(b)

X,Y→ / Z↓	00	01	11	10
0	1	1	0	0
1	0	1	0	1

Grading scheme:

- 10 marks for a minimized sum-of-products/product-of-sums form
- -2 marks for every non-maximal cube (collection of K-map squares with 1) identified and used in obtaining the funciton.
- Some non-standard techniques (like using xors and ands) by some students. We've given full marks as long as the resulting function was correct.
- In some cases, students identified the right maximal cube, but made a mistake in determining the product of terms that corresponds to the cube. 5 marks has been deducted for such mistakes. Basically, the function is wrong, but since we noticed that the error was in identifying the right product of terms for only one cube, we gave 5 marks.

2. *20 + 5 marks* A Boolean function is often specified as a collection of partial specifications. Consider the table shown below that specifies a function $F(a, b, c, d, e, f, g)$ in this manner.

a	b	c	d	e	f	g	F
0	0	-	-	-	-	1	1
0	1	1	-	-	-	1	-
1	0	0	0	-	-	-	0
-	-	1	0	-	-	0	0
-	1	-	-	1	1	1	1
1	0	0	-	1	-	1	0

Note the following points carefully about the entries in the table.

- Each row of the table above is a partial specification of what value F should have for certain combinations of input values.
- An *input value* of $-$ means that the input is allowed to be either 0 or 1. For example, the last row of the table represents four input combinations, namely $abcdefg$ = 100$\underline{\mathbf{0}}$1$\underline{\mathbf{0}}$1, 100$\underline{\mathbf{0}}$1$\underline{\mathbf{1}}$1, 100$\underline{\mathbf{1}}$1$\underline{\mathbf{0}}$1, 100$\underline{\mathbf{1}}$1$\underline{\mathbf{1}}$1, where the bold-faced underlined values are replacements of the $-$ entries in this row with 0/1. The last row of the table therefore specifies that F evaluates to 0 for each of the combinations of input values $abcdefg = 1000101, 1000111, 1001101, 1001111$.
- An *output value* of $-$ in a row means that the value of the output for the corresponding combinations of inputs does not matter, and can be set to either 0 or 1. You can choose what value to assign to the output for these input combinations, such that it doesn't conflict with the partial specifications given by other rows of the table. For example, considering row 3 ($abcdef = 011 - - - 1$, $F = -$) and row 6 ($abcdef = -1 - -111$, $F = 1$) of the table, we can see that when the inputs are $abcdef = 011 - 011$, $011 - 101$ or $011 - 001$, the value of F can be set to either 0 or 1. However, when $abcdef = 011 - 111$, row 6 requires F to be 1. By choosing what value to assign to the output judiciously when there is a choice, one can often come up with simple implementations of the function.
- For all input combinations not specified in a row of the table, the value of the output can be assumed to be $-$.

(a) Construct an ROBDD for F with the variable order $g < a < b < c < d < e < f$, where $g < a$ means that g appears closer to the root than a in any path from the root to the leaves of the ROBDD (assuming both g and a appear along that path). You need not show the computed table or unique table. Actually, if you observe the table carefully, you can construct the ROBDD fairly easily without needing the help of any computed table or unique table.

Grading scheme:

- If the graph drawn is not an ROBDD, 0 marks was given.
- If the graph drawn is an ROBDD, then
  - For every row of the table that is correctly implemented by the ROBDD (except the row where F is $-$), we have given 4 marks. This makes the total marks $4 \times 5 = 20$.
  - Those who have been able to identify the smallest ROBDD for the given function (as discussed in class) have been given an additional (bonus) 5 marks.

(b) How many other variable orders will yield an ROBDD for F that has the same size (number of nodes) as the one obtained using $g < a < b < c < d < e < f$. Please list these orders, and give brief justification for your answer. You need not construct the ROBDDs for the other orders.

Grading scheme:

- We've been very lenient with grading this. As long as anybody has given a set of orders, which are all correct (i.e. result in the same no. of nodes as with $g < a < b < c < d < e < f$), and has provided justification why all of these orders would give rise to the same size, we've given 5.
- Orders without justification have been given no marks.
- If somebody has given a set of orders, and some of these orders are incorrect (i.e. result in no. of nodes that is different from that obtained using $g < a < b < c < d < e < f$), no marks have been given. Such answers appear to be guesswork, some of which may just turn out to be correct. We definitely don't want to encourage guesswork.