TUTORIAL 6 · [COMPUTATION AND LOGIC]

OBJECTIVES

In this tutorial, you will:

- learn more about Karnaugh maps;
- convert logical expressions to CNF using Karnaugh maps;
- learn about minimal CNFs.

TASKS

Exercises 1–4 are mandatory. Exercises 3 and 4 have some optional tasks. Exercise 5 is optional.

SUBMIT a file `cl-tutorial-6` with your answers (image or pdf).

DEADLINE Saturday, 31st of October, 4 PM UK time

Good Scholarly Practice

Please remember the good scholarly practice requirements of the University regarding work for credit.

You can find guidance at the School page

> *http://web.inf.ed.ac.uk/infweb/admin/policies/academic-misconduct.*

This also has links to the relevant University pages. Please do not publish solutions to these exercises on the internet or elsewhere, to avoid others copying your solutions.

EXERCISE 1

MANDATORY | BEFORE TUTORIAL SESSION

Watch the Week 6 videos and read Chapter 12 (*Karnaugh Maps*) of the textbook.

For each of the following expressions, say how many clauses there are in the expression, fill in a Karnaugh map like the one below and indicate the blocks of zeros corresponding to each clause:

1. $B \vee D$
2. $(\neg A \vee C) \wedge (A \vee \neg D)$

		CD			
		00	01	11	10
AB	00				
	01				
	11				
	10				

EXERCISE 2

MANDATORY | BEFORE TUTORIAL SESSION

For each of the following Karnaugh maps, give an expression that is true for exactly the states marked with x, and another expression that is true for exactly the states marked with y.

		CD			
		00	01	11	10
AB	00	x	x	x	x
	01	x	x	x	x
	11	x	y	y	x
	10	x	y	y	x

		CD			
		00	01	11	10
AB	00	y	y	x	x
	01	x	x	x	x
	11	x	x	x	x
	10	x	x	x	x

How would your answers change if the word “exactly” were omitted from the question?

EXERCISE 3

MANDATORY | 🕒 BEFORE TUTORIAL SESSION

✎ Give a Karnaugh map for each of the following expressions and use it to derive an equivalent CNF:

3. $r \leftrightarrow (a \wedge b)$
4. $r \leftrightarrow (a \rightarrow b)$

These are two of the expressions for which you derived CNFs in Exercise 5 from Tutorial 5.

🧠 How do the CNFs derived by the two methods compare?

🏆 Can you use your KM to find a minimal CNF for each expression? Where, by minimal we mean a CNF with as few occurrences of literals as possible – pay attention to the distinction between the number of literals in an expression and the number of occurrences of literals, as the same literal may occur several times.

EXERCISE 4

MANDATORY | 🕒 BEFORE TUTORIAL SESSION

✎ For each of the following pairs of clauses, draw a Karnaugh map and show the two blocks of zero states corresponding to the two clauses:

5. $A \vee \neg B,\ \neg A \vee \neg D$
6. $A \vee \neg B,\ A \vee B \vee C$

✎ Use these Karnaugh maps to identify new clauses Δ_1 and Δ_2, different from both premises, such that the following sequents are valid:

7. $A \vee \neg B,\ \neg A \vee \neg D \vDash \Delta_1$
8. $A \vee \neg B,\ A \vee B \vee C \vDash \Delta_2$

✎ How many different solutions can you find for clauses Δ_1 and Δ_2?

EXERCISE 4 (CONT.)

OPTIONAL | 🕓 BEFORE TUTORIAL SESSION

🏆 Read about race conditions on the Wikipedia Karnaugh maps page and on how race conditions can be avoided by adding clauses.

For each of the Karnaugh Maps you have drawn (i.e. for 5 and 6), identify any race conditions and derive both a minimal CNF and a minimal race-free CNF.

EXERCISE 5

Discuss with your table colleagues and solve together this exercise.

Using an example, explain why Karnaugh maps use the order 00, 01, 11, 10 for the values of A, B and C, D rather than truth table order.

Hint: Think of which groups of cells you can/can't identify using conjunctions of literals.

Use the following A-B-C criteria to fine-tune your answer:

Accessibility – your explanation should be accessible to someone who has not taken INF1;
Brevity – shorter explanations are better;
Clarity – the explanation should be clear and unambiguous.

At the end of the session you will name a representative to share your answer with the class.

EXERCISE 5 (CONT.)

OPTIONAL | DURING TUTORIAL SESSION

Gray codes are useful in dealing with electromechanical switches, error correction in digital communications, and measuring angles, among many other things.

You can read more about their use in rotary encoders for angle-measuring devices here.

Discuss with your colleagues how to write in Haskell a function

```haskell
gray :: Int -> [[Int]]
```

that returns, for any input n, an n-bit Gray code.