# Informatics 1 - Computation & Logic: Tutorial 2

## Propositional Logic: Karnaugh Maps [1]

Week 4: 9 – 13 October 2017

Please attempt the entire worksheet in advance of the tutorial, and bring all work with you. Tutorials cannot function properly unless you study the material in advance. Attendance at tutorials is **obligatory**; please let the ITO know if you cannot attend.

You may work with others, indeed you should do so; but you must develop your own understanding; you can't phone a friend during the exam. If you do not master the coursework you are unlikely to pass the exams.

We will use propositional logic in many ways. One important use is to specify and reason about finite state systems. We use a propositional logic with $n$ propositional letters to describe a system with $n$ state bits that can be used to encode up to $2^n$ states.

Often we will want to specify some subset of the set of states. For example, we may want to specify the set legal states of a traffic light. For a system of cars and traffic lights we may want to specify the set of safe states – in which accidents will not happen.

It is natural to specify safety using a number of clauses, each of which addresses a particular hazard. A safety clause for a crossing with pedestrian and traffic lights might specify that at least one of the lights must be red, $R_P \vee R_T$. Later, we will use the tools developed from ideas suggested by some of the examples in this tutorial to specify and reason about such systems.

In this tutorial we focus on Karnaugh Maps a tool used the design of combinational logic circuits.

[1] *This tutorial exercise sheet was written by Michael Fourman.*
*Send comments to* `Michael.Fourman@ed.ac.uk`

## Constraints

A **literal** is either a propositional letter or its negation.

$$(\neg A \vee \neg G \vee R) \wedge (A \vee G) \wedge \neg R \tag{1}$$

We call a disjunction of literals a **clause**, or **constraint**. In this definition, we include the empty disjunction, equivalent to $\bot$.

Equation (1) is a conjunction of three clauses. We say that the propositional letter $A$ occurs negatively in $\neg A \vee \neg G \vee R$ and positively in $A \vee G$.

Since $\vee$ is associative and commutative, two clauses are equivalent if they mention the same literals. Any in which some letter occurs both positively and negatively is equivalent to $\top$ (for example, $A \vee \neg A \vee X$ and $B \vee \neg B$ are both equivalent to $\top$). We say such clauses are **trivial**. So every clause is either trivial – equivalent to $\top$, or is equivalent the disjunction of a (finite) set of literals that is not trivial.

We say that a constraint eliminates those states that make the corresponding disjunction false. The **constraint is satisified** by those states that make at least one of the literals in the constraint true. *The empty clause can never be satisfied* – there is no literal we can make true – it corresponds to $\bot$.

1. How many non-equivalent clauses are there for a system with $n$ propositional letters?

   *Hint:* In any non-trivial clause, each letter occurs either positively or negatively, or not at all.

An expression is in conjunctive normal form (CNF) if it is a *conjunction of constraints*, where each constraint is a disjunction of literals.

We say that a non trivial constraint eliminates those states that make the corresponding disjunction false. The constraint is satisified by those states that make at least one of the literals in the constraint true. The trivial constraint $\top$ is satisfied by every state; the impossible constraint $\bot$ eliminates every state.

A state satisfies some CNF iff it statisfies all of the conjoined constraints.

A state is eliminated by some CNF iff it is eliminated by at least one of the conjoined constraints.

Given expressions $\varphi, \theta, \psi$, we say $\varphi$ and $\theta$ **entail** $\psi$ (for which we write $\varphi, \theta \models \psi$) iff every state that satisfies both $\varphi$ and $\theta$ also satisfies $\psi$. When working with an entailment $\varphi, \theta \models \psi$, we refer to $\varphi, \theta$ as its **premises** and $\psi$ as its **conclusion**.

## Karnaugh Maps

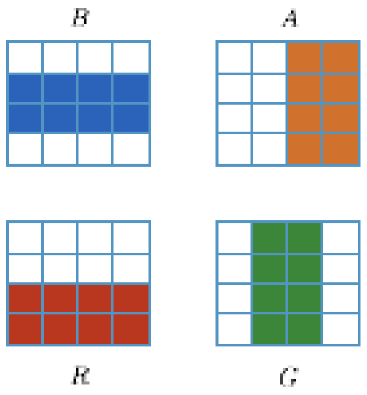
For this tutorial you will use Karnaugh Maps. It will be helpful to have a shorthand code for referring to the sixteen states represented by four boolean values assigned to the propositional letters $R, B, A, G$, and the corresponding regions of the Karnaugh map.

$R, B, A, G$ have binary values $r, b, a, g$ with 1 representing $\top$ and 0 representing $\bot$, we will refer to the state using the decimal value of the binary string $rbag$. Thus 0 represents the state 0000 in which all four atoms are false, while 15 represents the state 1111 in which they are all true.

2. Label each of the sixteen squares in the Karnaugh map with the corresponding number – as described in the preceding paragraph.

| | | AG | | | |
|---|---|---|---|---|---|
| | | 00 | 01 | 11 | 10 |
| RB | 00 | | | | |
| | 01 | | | | |
| | 11 | | | | |
| | 10 | | | | |

## K-map encoding



3. Name the states **eliminated** by each expression; mark them on the map.

(a) $A \vee \neg R$

| | | AG | | | |
|---|---|---|---|---|---|
| | | 00 | 01 | 11 | 10 |
| | 00 | | | | |
| RB | 01 | | | | |
| | 11 | | | | |
| | 10 | | | | |

(b) $\neg G \vee R$

| | | AG | | | |
|---|---|---|---|---|---|
| | | 00 | 01 | 11 | 10 |
| | 00 | | | | |
| RB | 01 | | | | |
| | 11 | | | | |
| | 10 | | | | |

(c) $(A \vee \neg R) \wedge (\neg G \vee R)$,

| | | AG | | | |
|---|---|---|---|---|---|
| | | 00 | 01 | 11 | 10 |
| | 00 | | | | |
| RB | 01 | | | | |
| | 11 | | | | |
| | 10 | | | | |

(d) $\neg G \vee A$

| | | AG | | | |
|---|---|---|---|---|---|
| | | 00 | 01 | 11 | 10 |
| | 00 | | | | |
| RB | 01 | | | | |
| | 11 | | | | |
| | 10 | | | | |

(e) $G \vee B$

| | | AG | | | |
|---|---|---|---|---|---|
| | | 00 | 01 | 11 | 10 |
| | 00 | | | | |
| RB | 01 | | | | |
| | 11 | | | | |
| | 10 | | | | |

(f) $A \vee B$

| | | AG | | | |
|---|---|---|---|---|---|
| | | 00 | 01 | 11 | 10 |
| | 00 | | | | |
| RB | 01 | | | | |
| | 11 | | | | |
| | 10 | | | | |

(g) The eliminated states in one of these examples include all of the eliminated states in one other example. Which examples are these? What does this tell you about the states that satisfy these two expressions?

4. For each Karnaugh map, give a constraint that eliminates, and an expression that is satisfied by, (exactly) the marked states.
   The marked states

   (a) are excluded by:
   satisfy:

| | | AG | | | |
|---|---|---|---|---|---|
| | | 00 | 01 | 11 | 10 |
| RB | 00 | | | | |
| | 01 | ● | ● | ● | ● |
| | 11 | | | | |
| | 10 | | | | |

   (b) are excluded by:
   satisfy:

| | | AG | | | |
|---|---|---|---|---|---|
| | | 00 | 01 | 11 | 10 |
| RB | 00 | | | | |
| | 01 | | | | |
| | 11 | | | ● | ● |
| | 10 | | | ● | ● |

   (c) are excluded by:
   satisfy:

| | | AG | | | |
|---|---|---|---|---|---|
| | | 00 | 01 | 11 | 10 |
| RB | 00 | | ● | | |
| | 01 | | ● | | |
| | 11 | | ● | | |
| | 10 | | ● | | |

   (d) are excluded by:
   satisfy:

| | | AG | | | |
|---|---|---|---|---|---|
| | | 00 | 01 | 11 | 10 |
| RB | 00 | | | | |
| | 01 | | | ● | ● |
| | 11 | | | ● | ● |
| | 10 | | | | |

   (e) are excluded by:
   satisfy:

| | | AG | | | |
|---|---|---|---|---|---|
| | | 00 | 01 | 11 | 10 |
| RB | 00 | | | | |
| | 01 | | | | |
| | 11 | | ● | ● | |
| | 10 | | ● | ● | |

   (f) are excluded by:
   satisfy:

| | | AG | | | |
|---|---|---|---|---|---|
| | | 00 | 01 | 11 | 10 |
| RB | 00 | ● | ● | | |
| | 01 | | | | |
| | 11 | | | | |
| | 10 | | | | |

   (g) What trivial answers could you have given had the word *exactly* been omitted above?

5. (a) If $\varphi, \theta \models \psi$ what can we say about every state excluded by $\psi$?

(b) Look back at your answers to question 4. Can you use them identify an example of a valid entailment? (If not, look again once you have completed the rest of this question.)

For each of the following pairs of constraints (taken from question 4), mark the states excluded by one or both of of the constraints and use the Karnaugh map to identify a clause entailed by these constraints. Write down the entailment and highlight the states excluded by the conclusion.

(c) 4(a), 4(b)

| | | AG | | | |
|---|---|---|---|---|---|
| | | 00 | 01 | 11 | 10 |
| | 00 | | | | |
| RB | 01 | | | | |
| | 11 | | | | |
| | 10 | | | | |

(e) 4(a), 4(e)

| | | AG | | | |
|---|---|---|---|---|---|
| | | 00 | 01 | 11 | 10 |
| | 00 | | | | |
| RB | 01 | | | | |
| | 11 | | | | |
| | 10 | | | | |

(d) 4(b), 4(c)

| | | AG | | | |
|---|---|---|---|---|---|
| | | 00 | 01 | 11 | 10 |
| | 00 | | | | |
| RB | 01 | | | | |
| | 11 | | | | |
| | 10 | | | | |

(f) 4(a), 4(f)

| | | AG | | | |
|---|---|---|---|---|---|
| | | 00 | 01 | 11 | 10 |
| | 00 | | | | |
| RB | 01 | | | | |
| | 11 | | | | |
| | 10 | | | | |

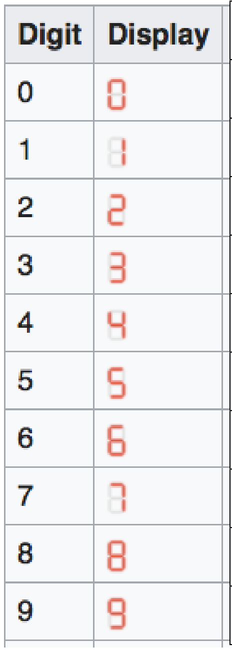
(g) Examine the entailments you have identified in this question. Can you find a common pattern?

(h) Can you use this pattern to guess a conclusion that is entailed by the following two premises?

$$A \vee B \vee C \vee D \vee R, \neg R \vee W \vee X \vee Y \vee Z \models$$

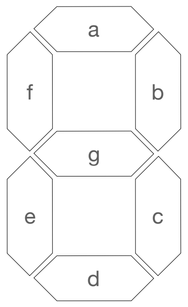
Karnaugh maps are routinely used in the design of logic circuits. In the tutorial. You will work as a group to design the logic to drive a seven-segment display – set of LED (or LCD) segments that render numerals 0 through 9 depending on a four-bit input, as shown below.

6. (a) Complete the table to show which segments should be on (1) or off (0), for each combination of inputs. Column (a) has been filled in to show that the (a) segment should be on except when the input represents 1 or 4.



| Digit | Display |
|---|---|
| 0 | 0 |
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |
| 4 | 4 |
| 5 | 5 |
| 6 | 6 |
| 7 | 7 |
| 8 | 8 |
| 9 | 9 |

| digit | RBAG | a | b | c | d | e | f | g |
|---|---|---|---|---|---|---|---|---|
| 0 | 0000 | 1 | | | | | | |
| 1 | 0001 | 0 | | | | | | |
| 2 | 0010 | 1 | | | | | | |
| 3 | 0011 | 1 | | | | | | |
| 4 | 0100 | 0 | | | | | | |
| 5 | 0101 | 1 | | | | | | |
| 6 | 0110 | 1 | | | | | | |
| 7 | 0111 | 1 | | | | | | |
| 8 | 1000 | 1 | | | | | | |
| 9 | 1001 | 1 | | | | | | |



| | | AG | | | |
|---|---|---|---|---|---|
| | a | 00 | 01 | 11 | 10 |
| RB | 00 | 1 | 0 | 1 | 1 |
| | 01 | 0 | 1 | 1 | 1 |
| | 11 | X | X | X | X |
| | 10 | 1 | 1 | X | X |

(b) The Karnaugh map is filled in from the (a) column.
X represents an unspecified output – your logic may produce 0 or 1.
What is the CNF required to drive the (a) segment?

## Tutorial Activities

1. (15m) Compare your answers to questions 1-5 with a buddy, then check that your group agrees on the answers.

   How would you check whether an answer to question 5(h) is correct?

   Ask one of the tutors if you have questions.

2. Logic for a seven-segment decoder.

   (a) First check that you all agree on the truth table, and the logic for the (a) segment you constructed in exercise 6.

   (b) As a group, you will need to construct a Karnaugh map for the six remaining segments. Individually, you should each construct maps for two or three different segments, using the templates provided below. Make sure you allocate the segments so that each is tackled by at least two people independently – this will allow you as a group to cross-check your answers. Each Karnaugh map is labelled in the top-left corner with the letter of a segment. It will help you to cross-check your work if you use the right map for each segment.

   We have not specified which segments shoule be lit when the four-bit binary input represents a number in the range $10 - -15$. For these inputs you should enter an X in the Karnaugh map to represent a *don't care* output. When you come to design the logic, you can treat these don't cares as either 0s or 1s – whichever makes life easier.

   (c) For each of your Karnaugh maps you should identify a CNF for the logic driving that segment of the display. Each clause of the CNF corresponds to a block of squares that includes no 1 entries on the Karnaugh map. Taken together the blocks must cover all of the 0 entries. We don't care whether the X entries fall within or outwith the blocks.

   Use the space below the KM to list the clauses – use a pencil!

   Cross-check your answer for each of your segments with those of the others in your group who have also worked on that segment.

   (d) Although each segment requires its own logic, some clauses may be shared between different segments. When this happens we can simplify the decoder.

   Look again at your four segments. Can you share clauses between different segments? What is the smallest number of clauses you can find that can be combined to drive all four segments?

   (e) Now work as a group to find the smallest number of clauses you can find to drive all seven segments.

|  |  | AG |  |  |  |
|---|---|---|---|---|---|
|  | b | 00 | 01 | 11 | 10 |
| RB | 00 |  |  |  |  |
|  | 01 |  |  |  |  |
|  | 11 |  |  |  |  |
|  | 10 |  |  |  |  |

|  |  | AG |  |  |  |
|---|---|---|---|---|---|
|  | d | 00 | 01 | 11 | 10 |
| RB | 00 |  |  |  |  |
|  | 01 |  |  |  |  |
|  | 11 |  |  |  |  |
|  | 10 |  |  |  |  |

|  |  | AG |  |  |  |
|---|---|---|---|---|---|
|  | c | 00 | 01 | 11 | 10 |
| RB | 00 |  |  |  |  |
|  | 01 |  |  |  |  |
|  | 11 |  |  |  |  |
|  | 10 |  |  |  |  |

|  |  | AG |  |  |  |
|---|---|---|---|---|---|
|  | e | 00 | 01 | 11 | 10 |
| RB | 00 |  |  |  |  |
|  | 01 |  |  |  |  |
|  | 11 |  |  |  |  |
|  | 10 |  |  |  |  |

| | | AG | | | |
|---|---|---|---|---|---|
| | f | 00 | 01 | 11 | 10 |
| RB | 00 | | | | |
| | 01 | | | | |
| | 11 | | | | |
| | 10 | | | | |

| | | AG | | | |
|---|---|---|---|---|---|
| | g | 00 | 01 | 11 | 10 |
| RB | 00 | | | | |
| | 01 | | | | |
| | 11 | | | | |
| | 10 | | | | |

## K-map encoding