Karnaugh Maps

Preparation

It is recommended that you read the **Conjunctive Normal Form** chapter in the logic notes.

1 Constraints

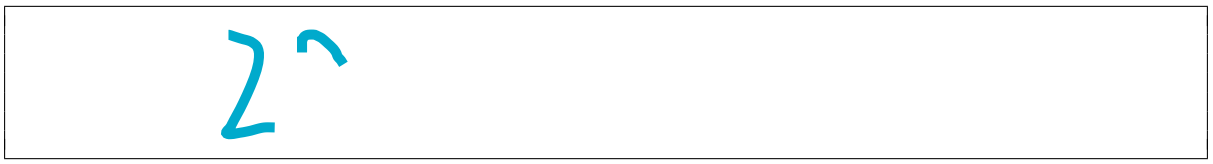
A **literal** is either a propositional letter or its negation. We call a disjunction of literals a **clause**, or **constraint**. Below is a conjunction of three clauses:

$$(\neg A \vee \neg C \vee D) \wedge (A \vee C) \wedge \neg D$$

We say that the propositional letter A occurs negatively in $\neg A \vee \neg C \vee D$ and positively in $A \vee C$.

Since $\vee$ is associative and commutative, two clauses are equivalent if they mention the same literals. Clauses 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 to the disjunction of a set of literals that is not trivial.

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 disjunctions of literals.

We often represent a CNF as a clausal form – a set of sets of literals – where the innersets are interpreted as disjunctions and the outer set as a conjunction. The inner disjunctive sets are called clauses, or constraints. For example, we can turn $(\neg A \vee \neg C \vee D) \wedge (A \vee C) \wedge \neg D$ into a set of sets:

$$\{\{\neg A, \neg C, D\}, \{A, C\}, \{\neg D\}\}$$

States A *state* of a set of atoms, is a set, Σ, of literals in which every atom occurs at most once, either positively or negatively. We say this is a **consistent** set of literals. A set of literals is **inconsistent** if it includes both A and $\neg A$ for some atom, A.

Since a state determines the value of every atom, two individuals in the same state are indistinguishable. So we can replace any universe by an equivalent universe in which the individuals are states. In fact, if we take all the states, we have a universal model for expressions using these atoms. Any universe in which these atoms are interpreted as predicates is equivalent to a sub-universe of the universal one.

In the universal model, each state simply lists the predicates for which it is true and those for which it is false, for example, `not isBig, isRed, isTriangle` , is red and is triangle, and is not big. So the state is just like a line in the truth table.

Any consistent set of literals, Γ, is a partial *valuation* it determines the values of some atoms.

Recall that $\Gamma \vDash b$ is valid in some universe iff $\vDash b$ is valid in the sub-universe of states that satisfy every predicate in Γ. If Σ is a state (a consistent set of literals), then $\Sigma \vDash b$ is valid iff $\vDash b$ is valid the sub-universe with a single state, Σ. This means that for any literal b, $\Sigma \vDash b$ in the universal model iff $b \in \Sigma$. Furthermore, if Γ is any consistent set of literals, then in the universal model, $\Gamma \vDash b$ iff $b \in \Gamma$.

Finally, if Δ is a set of literals, then in the universal model,
$\Gamma \vDash \Delta$ iff there is some $b \in \Delta$ such that $b \in \Gamma$;

if for every $d \in \Delta$, it is the case that $\neg d \in \Gamma$ then $\Gamma \vDash \neg \bigvee \Delta$; no consistent extension of Γ can include any $d \in \Delta$.

We say that a constraint, Δ, **eliminates** those valuations that make $\bigvee \Delta$ false, by making every $d \in \Delta$ false. The constraint, Δ, is **satisified** by those valuations that make $\bigvee \Delta$ true by making some $d \in \Delta$ true. The impossible constraint $\bot$ represented by the empty disjunctive clause eliminates every state.

A valuation satisfies some CNF iff it statisfies **every** constraint true. The trivial form `[]` with no constraints is always satisfied. A state is eliminated by some CNF iff it is eliminated by **at least one** of the conjoined constraints.

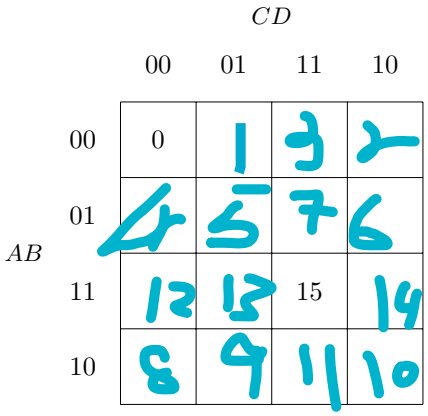
We give a Haskell implementation of these relationships at the end of this note.

2 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, A, B, C, D, and the corresponding regions of the Karnaugh map.

We can assign each of the atoms A, B, C, D a binary value, with 1 representing $\top$ and 0 representing $\bot$. We will refer to the state using the decimal value of the binary string, 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.
We will use these numbers as names for the states.

1. Label each of the sixteen squares in the Karnaugh map to the right with the corresponding number.



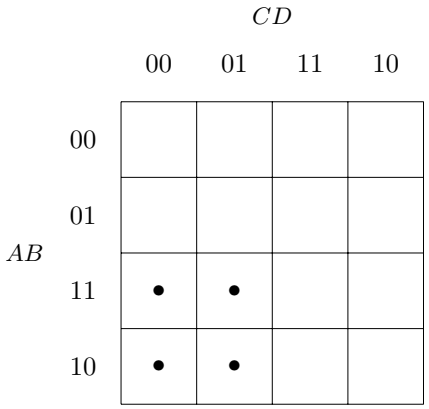
The Karnaugh map represents the univeral model for four atoms, A, B, C, D. Each square represents one of the 16 states. We can represent any one of the 64Ki[1] predicates on this model by labelling each square with 1 if the predicate is true for that state and 0 if it is false.

In this tutorial you will learn how to give a compact CNF for any one of these 64Ki predicates.

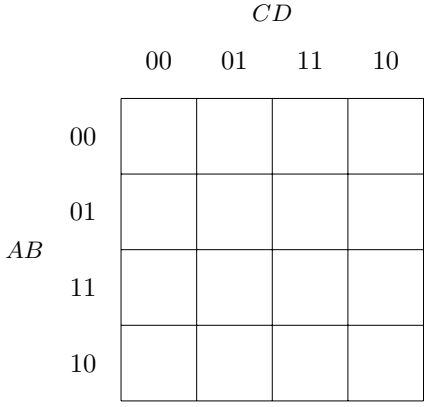
[1] 64×1024

2. Name the states **eliminated** by each disjunction and mark them on the map.

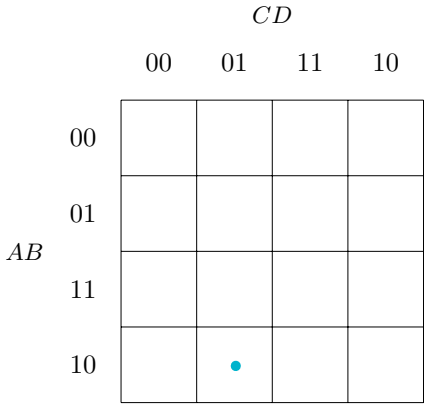
(a) $\neg A \vee C$
Eliminated states: 8,9,12,13



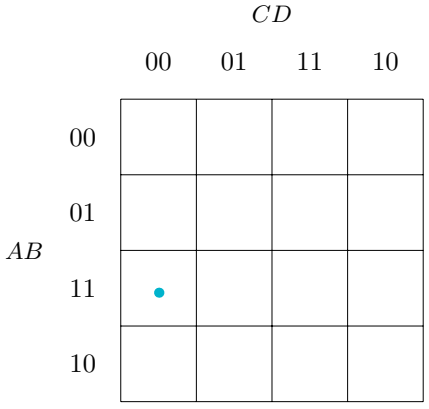
(b) $A \vee \neg D$
Eliminated states:



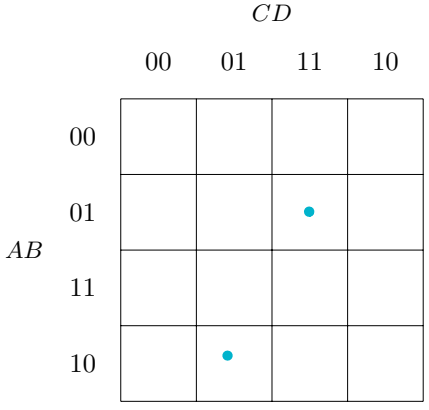
(c) $(\neg A \vee C) \wedge (A \vee \neg D)$
Eliminated states:



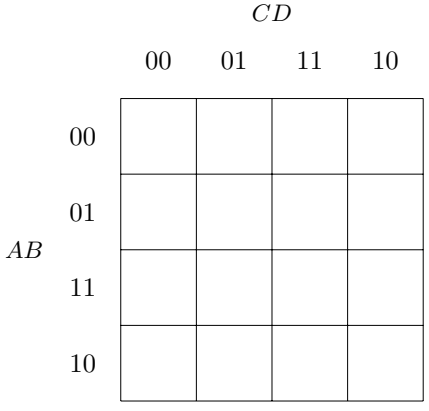
(d) $(\neg A \vee C) \wedge (A \vee \neg D)$
Eliminated states:



(e) $(\neg A \vee C) \wedge (A \vee \neg D)$
Eliminated states:



(f) $(\neg A \vee C) \wedge (A \vee \neg D)$
Eliminated states:



3. 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?

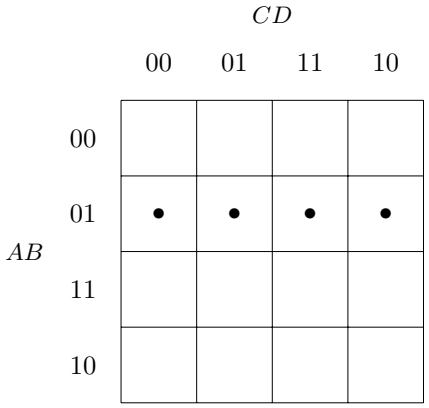
c d ef

not eliminated

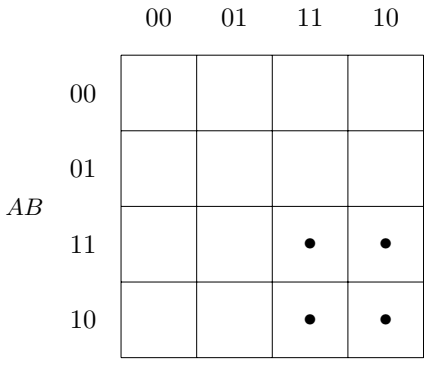
4. For each Karnaugh map, give a constraint that eliminates, and an expression that is satisfied by, (exactly) the marked states.

The marked states

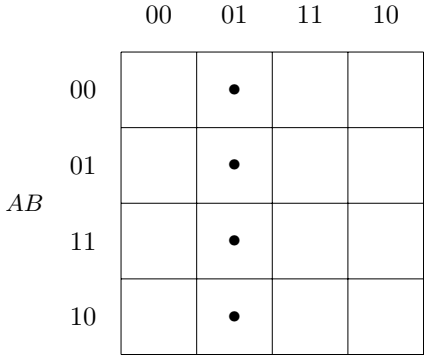
(a) are excluded exactly by: $A \vee \neg B$
satisfy exactly: $\neg A \wedge B$



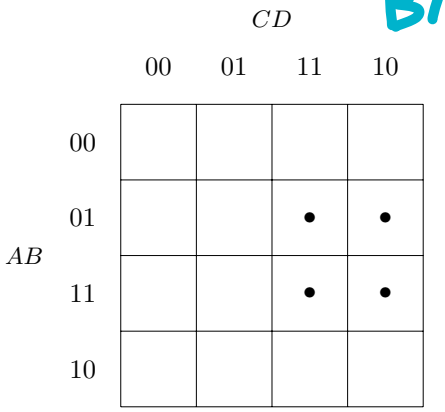
(b) are excluded exactly by: $\neg A \vee \neg C$
satisfy exactly: $A \wedge C$



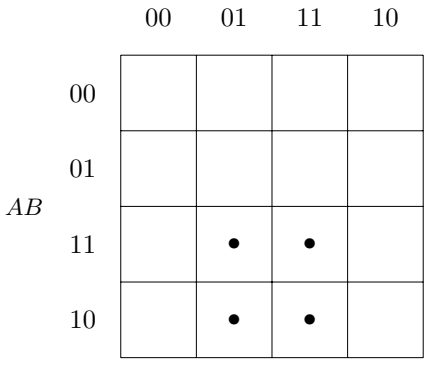
(c) are excluded exactly by: $C \vee \neg D$
satisfy exactly: $\neg C \wedge D$



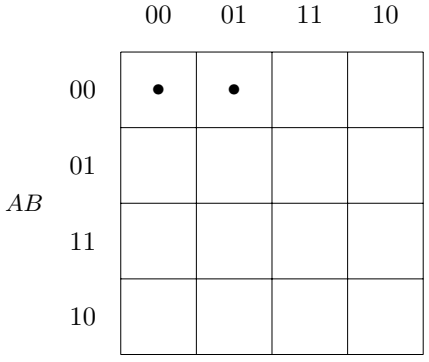
(d) are excluded exactly by: $\neg B \vee \neg C$
satisfy exactly: $B \wedge C$



(e) are excluded exactly by: $\neg A \vee \neg D$
satisfy exactly: $A \wedge D$



(f) are excluded exactly by: $A \vee B \vee C$
satisfy exactly: $\neg A \wedge \neg B \wedge \neg C$



5. What trivial answers could you have given had the word **exactly** been omitted above?

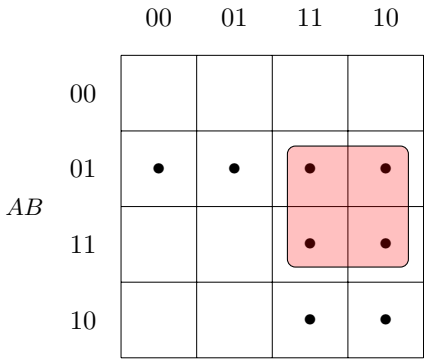
3 Rules and Karnaugh Maps

6. If $\phi, \theta \vDash \psi$ is valid in the universal model, what can we say about every state excluded by ψ? Drawing a diagram may help.

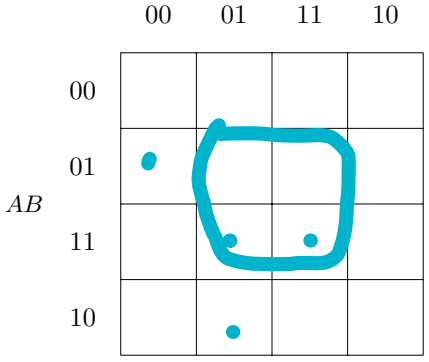
states excluded by ψ are either be excluded by ϕ or θ

7. For each of the following pairs of clauses (taken from question 4), mark the states excluded by one or both of the constraints and use the Karnaugh map to identify a clause entailed by these constraints, the identified clause should contain two distinct literals and it should not be the same as either of the premises. Write down the entailment and highlight the states excluded by the conclusion.

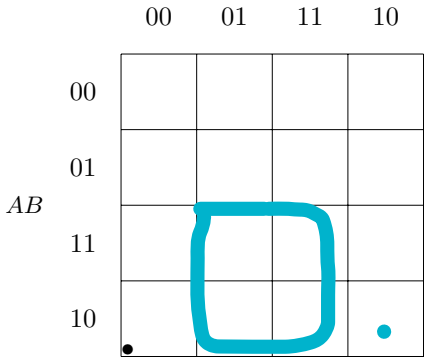
(a) 4(a), 4(b)
$A \vee \neg B, \neg A \vee \neg C \vDash \neg B \vee \neg C$



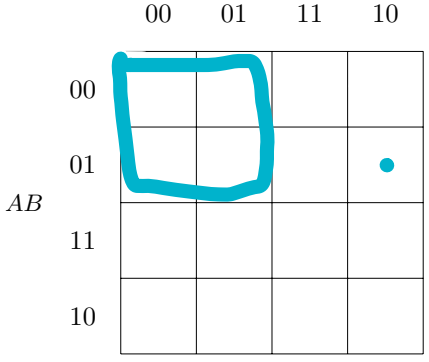
(c) 4(a), 4(e)
$A \vee \neg B, \neg A \vee \neg D \vDash \neg B \vee \neg D$



(b) 4(b), 4(c)
$\neg A \vee \neg C, C \vee \neg D \vDash \neg A \vee \neg D$



(d) 4(a), 4(f) text
$A \vee \neg B, A \vee B \vee C \vDash A \vee C$



Eliminate opposite literals

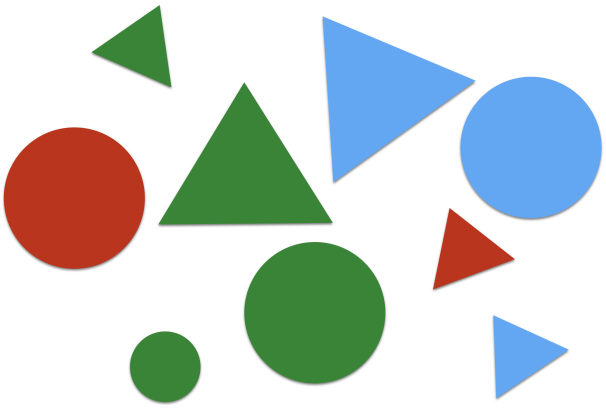
8. Examine the entailments you have identified in this question. Can you find a common pattern?

9. Can you use this pattern to guess a conclusion that is entailed by the following two premises?

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

$A \vee B \vee C \vee D \vee W \vee X \vee Y \vee Z$

4 Back to Basics



AB \ CD	00	01	11	10
00	1	1	1	1
01	1	1	0	1
11	0	0	0	0
10	0	1	1	0

10.
Uninhabited states:

In this exercise you will find a CNF to describe the universe of things from the tutorial 1.

Using the predicates $A =$ `isRed`, $B =$ `isBlue`, $C =$ `isSmall`, $D =$ `isDisc`, fill in the Karnaugh map to show which states are uninhabited in this universe.

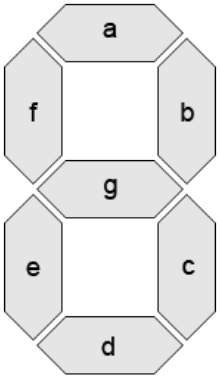
11. Use the Karnaugh map to derive a CNF that characterises our first universe.
Translate this back to the original predicates.

CNF:

$$(\neg A \vee \neg B) \wedge (\neg A \vee C \vee D) \wedge (\neg B \vee \neg C \vee \neg D) \wedge (\neg A \vee \neg C \vee D)$$

5 Seven-segment Display

A seven-segment display is a set of seven LED segments that render numerals 0 through 9 depending on a four-bit input, as shown below:



12. 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. Row 0 is also filled in to show that only segment (g) does not light up for digit 0.

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

digit	ABCD	a	b	c	d	e	f	g
0	0000	1	1	1	1	1	1	0
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						

13. The Karnaugh map bellow 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?

(a)

AB \ CD	00	01	11	10
00	1	0	1	1
01	0	1	1	1
11	X	X	X	X
10	1	1	X	X

14. Fill in a Karnaugh for each of the segments (some are filled for you) and extract a CNF from every map.

(b) CNF:

AB \ CD	00	01	11	10
00	1	1	1	1
01	1	0	1	0
11	X	X	X	X
10	1	1	X	X

(c) CNF:

AB \ CD	00	01	11	10
00	1	1	1	0
01	1	1	1	1
11	X	X	X	X
10	1	1	X	X

(d) CNF:

AB \ CD	00	01	11	10
00	1	0	1	1
01	0	1	0	1
11	X	X	X	X
10	1	1	X	X

(e) CNF:

AB \ CD	00	01	11	10
00	1	0	0	1
01	0	0	0	1
11	X	X	X	X
10	1	0	X	X

(f) CNF:

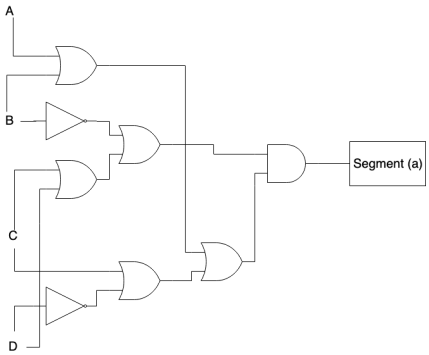
AB \ CD	00	01	11	10
00	1	0	0	0
01	1	1	0	1
11	X	X	X	X
10	1	1	X	X

(g) CNF:

AB \ CD	00	01	11	10
00	0	0	1	1
01	1	1	0	1
11	X	X	X	X
10	1	1	X	X

Circuits for Seven-segment Display

With CNFs for all the segments, we can construct circuits that make the segments light up. The circuit for segment (a) can look like this:



Doing it in Haskell

This code uses some Haskell constructs that you will encounter quite soon.

```
data Clause a  = Or[ Literal a ]
type Form a    = [ Clause a ]

neg :: Literal a -> Literal a
neg (P a) = N a
neg (N a) = P a

data Atom = A|B|C|D|W|X|Y|Z deriving Eq

eg = [ Or[N A, N C, P D], Or[P A, P C], Or[N D] ]

data Val a = And [ Literal a ]
vals :: [a] -> [Val a]
vals atoms =
  let
    vs []      =  [[]]
    vs (a:as)  =  [ P a: v | v <- vs as ] ++
                  [ N a: v | v <- vs as ]
  in map And (vs atoms)

v :: Eq a => Literal a -> (Val a -> Bool)
v lit (And gamma) = lit `elem` gamma

-- This is like our previous definition for |= but this speialised to the universal model
-- gamma makes some delta true (entails)
(|-) :: Eq a => Val a -> Clause a -> Bool
And gamma |- Or delta =
  or [ d `elem` gamma | d <- delta ]

-- This is no the same as (|/-). It is (entails not), not (not entails)
-- gamma makes every delta false
(|-/) :: Eq a => Val a -> Clause a -> Bool
And gamma |-/ Or delta =
  and [neg d `elem` gamma | d <- delta ]
```