# Informatics 1 - Computation & Logic: Tutorial 4

## Propositional Logic: Conjunctive Normal Form (CNF) [1]

Week 5: 17–21 October 2016

Please attempt the entire worksheet in advance of the tutorial, and bring with you all work, including (if a computer is involved) print-outs of code and test results. Tutorials cannot function properly unless you do the work in advance.

You may work with others, but you must understand the work; you can't phone a friend during the exam.

Assessment is formative, meaning that marks from coursework do not contribute to the final mark. But coursework is not optional. If you do not do the coursework you are unlikely to pass the exams.

Attendance at tutorials is **obligatory**; please let your tutor know if you cannot attend.

## CNF

An expression is in conjunctive normal form (CNF) if it is a *conjunction of disjunctions of literals*, where a literal is either an atomic propositional symbol or a negated atomic propositional symbol.

For example,

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

We call a disjunction of literals a **clause**, or **constraint**. Since $\vee$ is associative and commutative, we can view a clause as the disjunction of a (finite) set of literals.

There are three clauses in (1), with three elements, two elements, and one element, respectively:

$$\{\neg A, \neg G, R\} \quad \{A, G\} \quad \{\neg R\} \tag{2}$$

[1] *This tutorial exercise sheet was originally written by Paolo Besana, and extended by Thomas French, Areti Manataki, Michael Fourman, and Dave Cochran. Send comments to* `Michael.Fourman@ed.ac.uk`

Since $\wedge$ is also associative and commutative we can view a CNF as a set of clauses. So we can represent each CNF as a (finite) set of finite sets of literals which we call its **clausal form**.

The clausal form for Equation 1 is the set whose elements are the three clauses given in (2):

$$\big\{\{\neg A, \neg G, R\}, \{A, G\}, \{\neg R\}\big\}$$

We treat $\top$ as the empty conjunction (of no clauses), and $\bot$ as the empty disjunction (of no literals). So, a clause is $\bot$, or a literal, or a disjunction of clauses; and a conjunctive normal form is a conjunction of zero or more clauses.

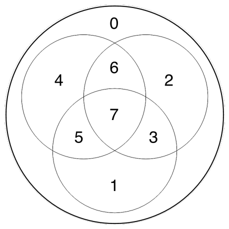
We will often call the individual clauses *constraints*. A state **satisfies** a given constraint iff it makes the corresponding disjunction true, that is, if it makes at least one of the literals in the constraint true. *The empty clause can never be satisfied* – there is no literal we can make true – it corresponds to $\bot$..

A state **satisfies** a given clausal form if it makes the corresponding conjunction true; that is, iff it satisfies every clause in the conjunction. The empty clausal form is always satisfied; it corresponds to $\top$.

1. For the purposes of this tutorial, it will be helpful to have a shorthand code for referring to the eight states represented by three boolean values assigned to the propositional letters $R, A, G$, and the corresponding regions of the Venn diagram.

   If $R, A, G$ have binary values $r, a, g$ with 1 representing $\top$ and 0 representing $\bot$, we will refer to the state using the decimal value of the binary string $rag$. Thus 0 represents the state 000 in which all three atoms are false, while 7 represents the state 111 in which they are all true.

   (a) Label each of the eight atomic regions in the Venn diagram with the corresponding number.

   (b) Say which states satisfy each of the following:

   i. the constraint $\{A, \neg R\}$, 0,1,2,3,6,7
   ii. the constraint $\{\neg G, R\}$, 0,2,4,5,6,7
   iii. the clausal form $\big\{\{A, \neg R\}, \{\neg G, R\}\big\}$, 0,2,6,7
   iv. the constraint $\{\neg G, A\}$0,2,3,4,6,7

   Note that adding the final constraint (given by resolution) would not eliminate any further states. The first constraint eliminates states $4, 5$, where $\neg A \wedge R$. The second eliminates states $1, 3$, where $G \wedge \neg R$. The state given by resolution eliminates states $1, 5$ where $G \wedge \neg A$ – both of which have already been eliminated.

2. The following expressions in CNF, on the left, are represented by the clausal forms shewn on the right:

$$\neg A \;\sim\; \{\{\neg A\}\} \tag{3}$$

$$A \vee \neg G \vee \neg R \;\sim\; \{\{A, \neg G, \neg R\}\} \tag{4}$$

$$A \wedge (\neg G \vee \neg R) \;\sim\; \{\{A\}, \{\neg G, \neg R\}\} \tag{5}$$

$$A \wedge \neg G \wedge \neg R \;\sim\; \{\{A\}, \{\neg G\}, \{\neg R\}\} \tag{6}$$

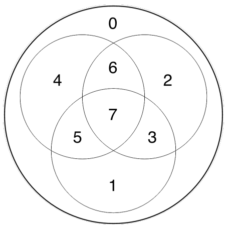
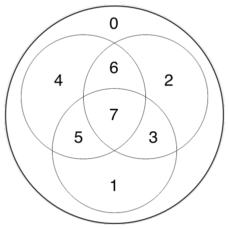
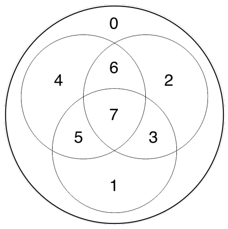
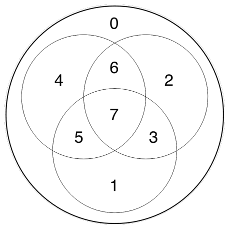
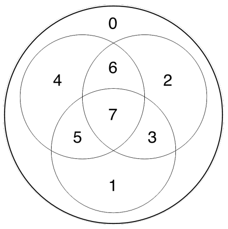
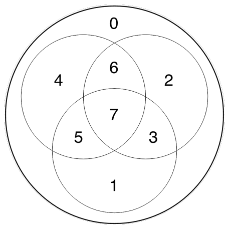
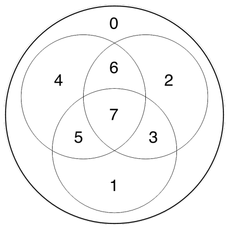
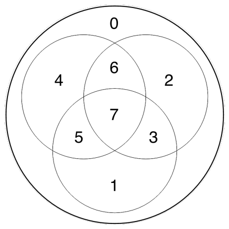
$$(A \vee \neg G) \wedge (\neg R \vee A) \;\sim\; \{\{A, \neg G\}, \{\neg R, A\}\} \tag{7}$$

$$\begin{array}{r}(A \vee G \vee \neg R) \wedge (\neg G \vee \neg R)\\ \wedge\, A \wedge (\neg G \vee \neg A)\end{array} \;\sim\; \left\{\begin{array}{r}\{A, G, \neg R\},\\ \{\neg G, \neg R\}, \{A\},\\ \{\neg G, \neg A\}\end{array}\right\} \tag{8}$$

$$, , , , \bot \;\sim\; \{\{\,\}\} \tag{9}$$

$$\top \;\sim\; \{\,\} \tag{10}$$

For each of these expressions use the coding introduced in Question 1 to identify the atomic regions of the Venn diagram *excluded* by each of its clauses.

| | | | |
|---|---|---|---|
| $\neg A$<br>2,3,6,7<br> | $A \vee \neg G \vee \neg R$<br>5<br> | $A \wedge (\neg G \vee \neg R)$<br>0,1,4,5; 5,7<br> | $A \wedge \neg G \wedge \neg R$<br>0,1,4,5; 1,3,5,7; 4,5,6,7<br> |
| $(A \vee \neg G)$<br>$\wedge\ (\neg R \vee A)$<br>1, 5; 4, 5<br> | $(A \vee G \vee \neg R)$<br>$\wedge\ (\neg G \vee \neg R)$<br>$\wedge\ A \wedge (\neg G \vee \neg A)$<br>4; 5, 7; 0, 1, 4, 5; 3, 7<br> | $\bot$<br>0, 1, 2, 3, 4, 5, 6, 7<br> | $\top$<br>*none*<br> |

## Conversion to CNF

To turn a CNF expression into clausal form, we simply turn each conjunct into a *set* of literals, and then convert the whole conjunction into a set of constraints, that is, a *set of sets* of literals. However, many expressions, for example,

$$(\neg A \lor G) \to R \tag{11}$$

are *not* in CNF.

To convert an arbitrary expression of propositional logic into CNF, and hence into a set of constraints, we apply the following equivalences:

$$\begin{aligned}
X \leftrightarrow Y &\equiv (X \to Y) \land (Y \to X) && (\leftrightarrow)\\
X \to Y &\equiv \neg X \lor Y && (\to)\\
X \oplus Y &\equiv (X \lor Y) \land (\neg X \lor \neg Y) && (\oplus)\\
\neg(X \land Y) &\equiv \neg X \lor \neg Y && (\neg\land)\\
\neg(X \lor Y) &\equiv \neg X \land \neg Y && (\neg\lor)\\
\neg\neg X &\equiv X && (\neg\neg)\\
X \lor (Y \land Z) &\equiv (X \lor Y) \land (X \lor Z) && (\text{dist-}\lor)
\end{aligned}$$

We make liberal use of the associativity and commutativity of conjunction and disjunction. So, for example,

$$\begin{aligned}
X \land (Z \land Y) &\equiv (X \land Y) \land Z && (\text{for which we write } X \land Y \land Z)\\
(Y \land Z) \lor X &\equiv X \lor (Y \land Z) && (\text{equivalent, by dist-}\lor\text{, to } (X \lor Y) \land (X \lor Z))
\end{aligned}$$

To convert an arbitrary expression to CNF:

- eliminate implications etc. by rewriting in terms of $\land, \lor, \neg$;
- push negations inside $\land$ and *vee*, using de Morgan ($\neg\lor, \neg\land$ and eliminate $\neg\neg$;
- push $\lor$ inside $\land$ using dist-$\lor$.

Thus we can convert the expression in (11) into CNF as follows:

$$\begin{aligned}
&\underline{(\neg A \lor G) \to R} && (\to)\\
\Rightarrow\ &\underline{\neg(\neg A \lor G)} \lor R && (\neg\lor)\\
\Rightarrow\ &(\underline{\neg\neg A} \land \neg G) \lor R && (\neg\neg)\\
\Rightarrow\ &\underline{(A \land \neg G) \lor R} && (\text{dist-}\lor)\\
\Rightarrow\ &(A \lor R) \land (\neg G \lor R) && ()
\end{aligned}$$

In other words, as you can verify this using a truth table, the expression in (11) is logically equivalent to the following CNF expression:

$$(A \lor R) \land (\neg G \lor R) \tag{12}$$

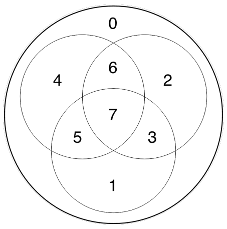
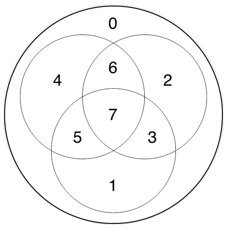
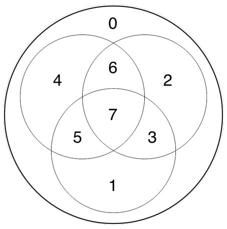
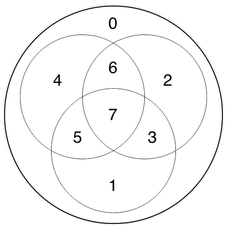
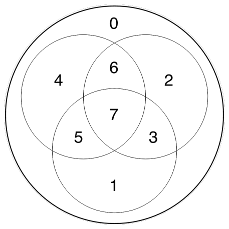
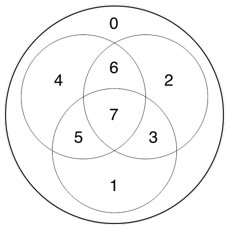
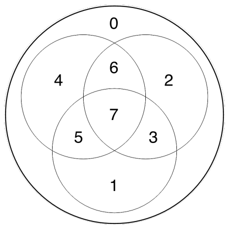
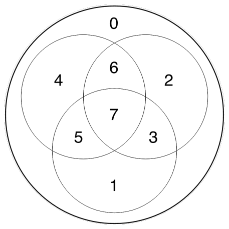
To turn a CNF expression into clausal form, simply turn each conjunct into a *set* of literals, and then convert the whole conjunction into a *set of sets* of literals.

Thus the clausal form of the CNF expression in (12) is the following:

$$\{\{A, R\}, \{\neg G, R\}\} \tag{13}$$

The constraint $\{A, R\}$ excludes states of the form $00x$, these are states 0 (000) and 1 (001); the constraint $\{\neg G, R\}$ excludes the states of the form $0x1$, these are states 3 (011) and 1 (001).

3. For each of the following expressions compute a CNF and list the states *excluded* by each clause of your CNF.

| | | | |
|---|---|---|---|
| $R \to A$<br><br>4, 5<br><br> | $A \to G$<br><br>2, 6<br><br> | $G \to R$<br><br>1, 3<br><br> | $(R \to A)$<br>$\wedge(A \to G)$<br>$\wedge(G \to R)$<br><br>4, 5; 2, 6; 1, 3<br><br> |
| $(R \wedge A) \to G$<br><br>$\neg R \vee \neg A \vee G$<br><br>6<br><br> | $G \to (A \vee R)$<br><br>$\neg G \vee A \vee R$<br><br>1<br><br> | $(G \vee A) \to R$<br><br>$(\neg G \vee R) \wedge (\neg A \vee R)$<br><br>1, 3; 2, 3<br><br> | $((R \wedge A) \to G)$<br>$\wedge(G \to (A \vee R))$<br>$\wedge((G \vee A) \to R)$<br>$(\neg R \vee \neg A \vee G)$<br>$\wedge(\neg G \vee A \vee R)$<br>$\wedge(\neg G \vee R) \wedge (\neg A \vee R)$<br><br>6; 1; 1, 3; 2, 3<br><br> |

4. Bonus question!

   - For our language with three atoms, $R, A, G$ how many different clauses are there?

     Hint: count the clauses that mention 0, 1, 2, or 3 of the atoms.

   - For each example in the right-hand column of Question 3, you have produced a set of clauses to give a CNF.

     Which other clauses, if any, can be added to your CNF without excluding any additional states? Is there a pattern?

   When we can add a new constraint without excluding any additional states, we say the new constraint is **entailed** by the existing constraints.

   The point of this 'bonus question' is to introduce the notion of entailment.

   We use the 'turnstile' ($\vdash$) notation for entailment: if $\Gamma$ is a set of constraints (a clausal form), and $\varphi$ is a constraint we write

$$\Gamma \vdash \varphi$$

   to indicate that the constraints in $\Gamma$ entail $\varphi$.

   A constraint $X$ is entailed by a set of constraints $\Gamma$ iff the set of states it excludes is covered by the union of the sets of states excluded by constraints in $\Gamma$.

   We can always weaken a constraint by adding further disjuncts to the clause – the exclusions associated with such a weakening are covered by the exclusions of the original clause – but we are primarily interested in the non-trivial entailments.

   In the top line, the constraints $R \to A$, $A \to G$, $G \to R$, taken together, eliminate the states $1, 2, 3, 4, 5, 6$. Observe that this can be achieved equally well by the (equivalent) set of contraints $A \to R$, $G \to A$, $R \to G$.

   So, for $\begin{array}{l}(R \to A)\\ \wedge(A \to G)\\ \wedge(G \to R)\end{array}$, the non-trivial entailments are $\begin{array}{l}(A \to R)\\ \wedge(G \to A)\\ \wedge(R \to G)\end{array}$

   On the bottom line, our constraints, taken together eliminate four states, $1, 2, 3, 6$. We can achieve the same effect with two simple constraints, $A \to G$ (which eliminates $6, 2$), and $G \to R$ (which eliminates $1, 3$).

   So, for $\begin{array}{l}((R \wedge A) \to G)\\ \wedge(G \to (A \vee R))\\ \wedge((G \vee A) \to R)\end{array}$, the non-trivial entailment is $A \to G$ ($G \to R$ arises in the normal form of the third conjunct).