Session Objectives

- Use De Morgan's laws to simplify logical expressions
- Define disjunctive and conjunctive normal forms
- Check if an expression is in CNF or DNF
- Rewrite an expression in DNF
- Define the Natural Deduction System
- Read and check a proof that uses Natural Deduction

De Morgan's Laws

$$\neg p \wedge \neg q \Leftrightarrow \neg(p \vee q) \tag{1}$$

$$\neg p \vee \neg q \Leftrightarrow \neg(p \wedge q) \tag{2}$$

De Morgan's laws for propositional logic can be used to prove the corresponding laws for set theory. This is done by letting each basic proposition represent $x \in S$ for each set S. If p represents $x \in A$ and q represents $x \in B$, then $p \wedge q$ is equivalent to $x \in (A \cap B)$, etc.

Applications of De Morgan's Laws

- Suppose you want to write a loop which inputs and processes pairs of data values until two zeros are found.

```
LOOP
  Get(First); Get(Second);
  EXIT WHEN First = 0 AND Second = 0
  -- Process the pair of values First and Second
END LOOP;
```

- An alternative way to write this loop would be using a `WHILE` construction. We need the loop to continue while the condition `First=0 AND Second=0` is false.

- In logical notation, we let the proposition p represent `First=0` and q represent `Second=0`.
- We require the loop to continue while $p \wedge q$ is false, or equivalently, while $\neg(p \wedge q)$ is true.
- By De Morgan's second law (2), this is equivalent to $\neg p \vee \neg q$ so we can write the loop in Ada as

```
Get(First); Get(Second);
WHILE First /= 0 OR Second /= 0
  -- Process the pair of values First and Second
  Get(First); Get(Second);
END LOOP;
```

Here we have written `First /= 0` for `NOT (First = 0)`.

Normal Forms

- Truth tables are very handy for checking the effect of combinations of logic gates when designing hardware.
- When the expressions become more complicated, the size of the truth tables grows rapidly: if there are n simple propositions, then we require 2^n combinations of truth values.
- This becomes unmanageable (at least by hand) and so more algebraic methods are needed: these involve the application of laws (amongst them logical equivalences like distributivity and De Morgan's laws) to make the process more automated; an alternative approach is resolution, which you studied in Introduction to AI.

Choice of Operators

- There are many possible choices of operators.
- We want to be able to model all possible truth functions.
- $p \wedge q$ is logically equivalent to $\neg(p \rightarrow (\neg q))$.
- $p \vee q$ is logically equivalent to $(\neg p) \rightarrow q$.
- We cannot reduce the set $\{\neg, \rightarrow\}$ to a single logical operator.

p	$\wedge$	q	$\neg$	$(p$	$\rightarrow$	$(\neg$	$q))$
T	T	T	T	T	F	F	T
T	F	F	F	T	T	T	F
F	F	T	F	F	T	F	T
F	F	F	F	F	T	T	F

Logical equivalence of $p \wedge q$ and $\neg(p \rightarrow (\neg q))$.

Choice of Operators II

- The operator NAND ('not and'), which I shall write $\overline{\wedge}$, it is true that any formula is logically equivalent to a formula using only this operator.
- Why then should we use four logical operators where one will do? The answer is in readability; we are used to expressing ourselves with the four operators, and hence it is easier to understand formulae written with them. For example $p \vee q$ is a much clearer expression than $(p \mathbin{\overline{\wedge}} p) \mathbin{\overline{\wedge}} (q \mathbin{\overline{\wedge}} q)$.
- For machinery, the story is quite different. An electronic logical network can always be constructed from a single kind of component, namely a NAND gate, rather than several types. This makes the construction of large scale integrated circuits easier and cheaper than it would be if several different types of component were used.

DNF and CNF

- A formula $p_1 \vee p_2 \vee \cdots \vee p_n$ is called the disjunction of p_1, p_2, $\ldots$, p_n.
- The formula $p_1 \wedge p_2 \wedge \cdots \wedge p_n$ is called the conjunction of p_1, p_2, $\ldots$, p_n.
- A fomula which is either a propositional variable or its negation is called a literal.
- A formula is in disjunctive normal form (DNF) if it is a disjunction of a conjunction of literals. A formula is in conjunctive normal form (CNF) it is a conjunction of disjunctions of literals.
- A truth function is one possible truth table. For a given set of propositional variables $\{p_1, \ldots, p_n\}$ the value T or F is listed against each of the 2^n possible assignments of truth values to the variables.

Examples

1. The following are literals: p, $\neg p$, p_7.
2. The following are in disjunctive normal form: $(p \wedge q) \vee (\neg p \wedge \neg q)$, $\neg p \wedge \neg q$.
3. The following are in conjunctive normal form: $(p \vee q) \wedge (\neg p \vee \neg q)$, $p \wedge \neg q$.
4. Which of the following formulas are literals, in DNF, or in CNF? (Hint: some may be more than one).

$$p$$

$$(\neg p \wedge q) \vee (p \wedge \neg r) \vee (\neg q \wedge \neg r \wedge \neg s)$$

$$(\neg p \vee q) \wedge (\neg q \vee \neg r \vee \neg s) \wedge (p \vee \neg q)$$

Solution

Which of the following formulas are literals, in DNF, or in CNF? (Hint: some may be more than one).

1. p
2. $(\neg p \wedge q) \vee (p \wedge \neg r) \vee (\neg q \wedge \neg r \wedge \neg s)$
3. $(\neg p \vee q) \wedge (\neg q \vee \neg r \vee \neg s) \wedge (p \vee \neg q)$

1. literal, CNF and DNF.
2. DNF, but not CNF.
3. CNF, but not DNF.

1. Suppose that the propositional variables are p_1, p_2, . . . ,p_n. Consider the truth table of ϕ.
2. If every row has the value F, then we can use the formula $\bot$ which is in DNF.
3. Otherwise there is a row v where the truth table has the value T. For each assignment in row v of T and F to the propositional variables, let ϕ_v be the conjunction of literals, one for each propositional variable, which are p_i for the variables assigned T and $\neg p_i$ for the variables assigned F. (e.g. if $n = 4$ and $v(p_1) = v(p_4) = T$ and $v(p_2) = v(p_3) = F$, then $\phi_v = p_1 \wedge \neg p_2 \wedge \neg p_3 \wedge p_4$).
4. By definition, ϕ_v is true only in row v of the truth table.
5. List the rows of the truth table where ϕ has the value T as v_1, . . . , v_m. Then let δ be the formula $\phi_{v_1} \vee \phi_{v_2} \vee \cdots \vee \phi_{v_m}$.

DNF and CNF

- Hence every wff is logically equivalent to a wff in DNF.
- Using De Morgan's laws, we can use this result to prove that every formula is logically equivalent to one in CNF.

  1. If ϕ denotes the formula, put $\psi = \neg\phi$ into DNF.
  2. $\phi = \neg\psi$: applying negation switches round all the $\wedge$ and $\vee$ operators. This turns conjunctions into disjunctions and disjunctions into conjunctions. DNF becomes CNF.

Inference Systems

- So far we have used an informal method of deriving one proposition from others using truth tables.
- Now we shall consider a more formal approach to logical inference using a formal inference system.
- The propositional logic together with its inference system is called the propositional calculus.

Inference Systems and Proof

An inference system has two components.

Axioms: propositions which are asserted to be true.

Inference Rules: rules by which may derive some propositions as logical consequences of other propositions.

- A proposition p derived from the axioms by repeatedly using the inference rules is called a theorem of the propositional calculus.
- Note that we do not consider truth tables at all; we are simply applying rules.
- The process of applying the inference rules to the axioms to derive the theorem is called a formal proof. In principle, this can be automated.

Proof and Logical Consequence

If f is a formula and there is a proof of f from a set of axioms S, then we write $S \vdash f$. This is a different statement from $S \models f$.

$S \models f$ means that f is a logical consequence of S; whenever propositional variables are assigned values so that all expressions in S are true, then f is also true.

$S \vdash f$ means that f can be inferred from S; by applying inference rules we can derive f using S as our set of assumptions.

A proposition p that can be proved without any axioms is written as $\vdash p$ and can be used as an axiom in subsequent proofs.

Natural Deduction System

Negation.

Introduction

$$\frac{P, a \vdash b \quad P, a \vdash \neg b}{P \vdash \neg a}. \tag{3}$$

This is also known as proof by contradiction.

Elimination

$$\frac{\neg\neg p}{p} \tag{4}$$

Implication.

Introduction

$$\frac{P, a \vdash b}{P \vdash a \to b} \tag{5}$$

Elimination

$$\frac{a, a \to b}{b} \tag{6}$$

This is known as modus ponens, a Latin term dating back to the study of logic in the Middle Ages.

Conjunction.

Introduction

$$\frac{p, q}{p \wedge q} \quad (7)$$

Elimination

$$\frac{p \wedge q}{p} \quad \frac{p \wedge q}{q} \quad (8)$$

Disjunction.

Introduction

$$\frac{p}{p \vee q} \quad (9)$$

Given p we can deduce $p \vee q$ for any proposition q.

Elimination

$$\frac{P, a \vdash c \quad P, b \vdash c \quad a \vee b}{P \vdash c} \quad (10)$$

This is also known as disjunctive syllogism.

Assumptions

- The three rules (3), (10) and (5) make use of the notion of an assumption, that is a proposition a which we assert temporarily for the sake of argument.
- Assumptions cannot be viewed as established propositions, and nor can anything we derive from them, until the assumption is discharged by use of one of these three rules.
- The scope of an assumption starts from the line where the assumption is made until just before the line where it is discharged.

Formal Proof I

$p \wedge q \vdash p \vee q$

1. $p \wedge q$ Premise
2. p From 1 by $\wedge$-elimination
3. $p \vee q$ From 2 by $\vee$-introduction

Formal Proof II

$p \rightarrow q \vdash \neg(p \wedge \neg q)$

The strategy is proof by contradiction, so we will assume $p \wedge \neg q$ (i.e. the negation of the statement we are trying to prove). In the proof, the vertical line denotes the scope of the assumption and $\wedge$-E stands for $\wedge$-elimination.

1.	$p \rightarrow q$	Premise
2.	$p \wedge \neg q$	Assumption
3.	p	From 2 by $\wedge$-E
4.	q	From 1 and 3 by modus ponens ($\rightarrow$-E)
5.	$\neg q$	From 2 by $\wedge$-E
6.	$\neg(p \wedge \neg q)$	From 2, 4 and 5 by proof by contradiction ($\neg$-I)

Properties of Propositional Calculus

Consistency We want the inference rules to be sound: that is, any theorem that we prove should be true.

Completeness We want the inference rules to be complete so that there is a proof for every true statement.

We can summarise this by saying that for any set S of propositions, we want $S \vdash p$ if and only if $S \models p$. It can be proved (though the proof is beyond the scope of this module) that this equivalence does hold for propositional logic. So in the propositional logic we may use either truth table methods of formal inference to prove a statement.

Session Objectives

- Use De Morgan's laws to simplify logical expressions
- Define disjunctive and conjunctive normal forms
- Check if an expression is in CNF or DNF
- Rewrite an expression in DNF
- Define the Natural Deduction System
- Read and check a proof that uses Natural Deduction