Logical reasoning

- Can we formalise our thought processes?

Why study (mathematical) logic?

*Logic:* the science or method of reasoning

*Logical:* able to reason correctly

- Human reasoning poorly understood, except for *LOGICAL DEDUCTION*
  - reaching conclusion from general rules by reasoning

- Intelligence is about coming to the right conclusion

- Mathematical logic formalises the syntax (structures) and semantics (meaning) used in deduction

- *PROOF METHODS* - determine if a given structure is true

- *PROPOSITIONAL LOGIC* - the simplest level of logic

- *PREDICATE LOGIC* - more advanced level of logic
Formalising reasoning by using logic

Example:

*If the temperature exceeds 40 degrees then children will be sent home from school*  (1)

*The temperature is currently 42 degrees*  (2)

If these two assertions are correct, then we conclude that:

*There are no children at school*

The argument has the form

| If $P$ is true then $Q$ is true.  
$P$ is true.  
Therefore $Q$ is true. |

The conclusion is -

valid by virtue of the form of the argument, and independent of the assertions associated with $P$ and $Q$. 
**Propositional logic**

- A *proposition* is a statement that can be either **true** or **false**.
- We use *propositional symbols* (also called variables) such as \( p, q, r \), etc. to represent propositions.
- Propositions represented by a single propositional symbol are called *atomic* propositions.
- We can represent more complicated propositions by *compound* formulas.
- Compound formulas built up from simpler formulas (propositions) using *logical connectives*.

<table>
<thead>
<tr>
<th>CONNECTIVE</th>
<th>FORMULA</th>
<th>PROPOSITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>negation</td>
<td>( \neg P )</td>
<td>(not P)</td>
</tr>
<tr>
<td>disjunction</td>
<td>( P \lor Q )</td>
<td>(P or Q)</td>
</tr>
<tr>
<td>conjunction</td>
<td>( P \land Q )</td>
<td>(P and Q)</td>
</tr>
<tr>
<td>implication</td>
<td>( P \rightarrow Q )</td>
<td>(if P then Q)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Q if P)</td>
</tr>
</tbody>
</table>

In \( P \rightarrow Q \), P is the *Antecedent*,
Q the *Consequent*.
Evaluating propositions using truth tables

A truth table represents the values (true or false) of a proposition for all possible values of the atomic propositions making it up.

NEGATION

<table>
<thead>
<tr>
<th>P</th>
<th>¬ P</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
</tr>
</tbody>
</table>

DISJUNCTION (Boolean OR) and CONJUNCTION (Boolean AND)

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>P \lor Q</th>
<th>P \land Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>False</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>true</td>
<td>False</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>true</td>
<td>False</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>True</td>
</tr>
</tbody>
</table>
IMPLICATION

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>P → Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
</tbody>
</table>

Note: ‘P implies Q’ is not true when P is true but Q is false (3rd row above), so P implies Q is assumed true for all other cases.

Also, causality is not necessarily assumed. In other words, Q may or may not be caused by P, Eg,

If it rained, then the ground was wet       (causal implication)

If the ground is wet, then the road is wet (no causality)

BICONDITIONAL

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>P ↔ Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
</tbody>
</table>
Inference  
- reaching a conclusion

Problem: Deciding whether or not a particular CONCLUSION follows from a given set of PREMISES.

Definition:

A formula (proposition) $Q$ is said to be a LOGICAL CONSEQUENCE of a set of formulas $S = \{P_1, ..., P_n\}$ if it is the case that, whenever EVERY formula in $S$ is true, $Q$ must also be true.

$Q$ is a VALID CONCLUSION (or INFERENCE) from the premises $S$ also expressed as

$$
P_1
P_2
\cdot
\cdot
\cdot
P_n
\phantom{--------}
Q
$$
The truth table method of proof

The above definition gives a method for determining validity of a given inference:

List (in a truth table) all possible combinations of truth values of all propositional symbols appearing in the premises and the conclusion.

For each instance in which all the premises are true, check that the conclusion is also true.

Example 1: The following inference is valid

\[
\begin{array}{c}
P \\
\hline
P \\
P
\end{array} \rightarrow Q
\]

\[
\begin{array}{c|c|c}
P & Q & P \rightarrow Q \\
\hline
false & false & true \\
false & true & true \\
true & false & false \\
true & true & true \\
\end{array}
\]
Example 2: The following is NOT a valid inference.

\[ R \rightarrow (P \lor Q) \]
\[ Q \]

\[ \hline \]
\[ R \rightarrow P \]

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>R</th>
<th>P \lor Q</th>
<th>R\rightarrow(P \lor Q)</th>
<th>R\rightarrow P</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
</tbody>
</table>

The truth table method of proof becomes impractical as the number of distinct propositional symbols appearing in the premises and conclusion increases.

Is there a better way?
Clausal notation

A propositional formula of the form

\[ Q \leftarrow (P_1 \land P_2 \land \ldots \land P_n) \]

is called a \textit{CLAUSE}.

Using a slightly different notation

\[ Q \leftarrow P_1, \ldots, P_n \]

(ie. conjunction is denoted by a \texttt{,} instead of a \texttt{\land})

We also adopt the notational convention that a proposition \( P \) can be denoted by the \textit{clause}

\[ P \leftarrow \]

Resolution

The following is a valid inference

\[
\begin{align*}
Q &\leftarrow P, S_1, \ldots, S_n \\
P &\leftarrow R_1, \ldots, R_m \\
\hline \\
Q &\leftarrow R_1, \ldots, R_m, S_1, \ldots, S_n
\end{align*}
\]
The above *INFEERENCE RULE* is called *RESOLUTION*

The conclusion is called the *RESOLVENT* of the two premises.

**Proof using resolution**

Example: Suppose we have the following set of formulas (clauses)

\[ p \] \[ q \] \[ u \leftarrow p, q \] \[ v \leftarrow u, s \] \[ w \leftarrow v, r \] \[ s \leftarrow u \]

If these clauses are all assumed to be true, is it the case that \( v \) is also true?

In other words, is \( v \) a logical consequence of the above set of clauses?
We may proceed as follows:

\[ \text{v} \leftarrow \text{u}, \text{s} \]  \quad \text{[clause 4]}

\[ \text{u} \leftarrow \text{p}, \text{q} \]  \quad \text{[clause 3]}

\[ \text{v} \leftarrow \text{p}, \text{q}, \text{s} \]  \quad \text{[resolvent]}

\[ \text{p} \leftarrow \]  \quad \text{[clause 1]}

\[ \text{v} \leftarrow \text{q}, \text{s} \]  \quad \text{[resolvent]}

\[ \text{q} \leftarrow \]  \quad \text{[clause 2]}

\[ \text{v} \leftarrow \text{s} \]  \quad \text{[resolvent]}

\[ \text{s} \leftarrow \text{u} \]  \quad \text{[clause 6]}

\[ \text{v} \leftarrow \text{u} \]  \quad \text{[resolvent]}

\[ \text{u} \leftarrow \text{p}, \text{q} \]  \quad \text{[clause 3]}

\[ \text{v} \leftarrow \text{p}, \text{q} \]  \quad \text{[resolvent]}

\[ \text{p} \leftarrow \]  \quad \text{[clause 1]}

\[ \text{v} \leftarrow \text{q} \]  \quad \text{[resolvent]}

\[ \text{q} \leftarrow \]  \quad \text{[clause 2]}

\[ \text{v} \leftarrow \]  \quad \text{[resolvent]}
Note:

(i) There are 7 propositional symbols in the above set of clauses.

The appropriate truth table would have to contain \(2^7 = 128\) lines!

(ii) Use of the resolution inference rule gives a relatively efficient method for determining the logical consequences of an arbitrary set of *CLAUSES*.

This is, more or less, how the programming language *PROLOG* does it!
Proof using Resolution

First year requirements for a maths major:
1. Calculus Stream, and
2. Algebra Stream

1. Calculus Stream made up of
   Basic Calculus (equivalent to TEE calculus)
   Advanced Calculus

   Basic Calculus can also be completed by doing
   Calculus A, and
   Calculus B

2. Algebra Stream consists of
   Group Theory
   Linear Algebra
   Discrete Structures

Joanne has completed TEE Calculus, Advanced
Calculus and Linear Algebra

Has Joanne met the first year requirements for a
maths major?
Database of clauses

\[
\text{mathmajor } \leftarrow \text{ calculus, algebra} \\
\quad [\text{clause 1}]
\]

\[
\text{calculus } \leftarrow \text{ bascalc, advcalc} \\
\quad [\text{clause 2}]
\]

\[
\text{bascalc } \leftarrow \text{ teecalc} \\
\quad [\text{clause 3}]
\]

\[
\text{bascalc } \leftarrow \text{ calcA, calcB} \\
\quad [\text{clause 4}]
\]

\[
\text{algebra } \leftarrow \text{ group} \\
\quad [\text{clause 5}]
\]

\[
\text{algebra } \leftarrow \text{ linear} \\
\quad [\text{clause 6}]
\]

\[
\text{algebra } \leftarrow \text{ discrete} \\
\quad [\text{clause 7}]
\]

\[
\text{teecalc } \leftarrow \\
\quad [\text{clause 8}]
\]

\[
\text{advcalc } \leftarrow \\
\quad [\text{clause 9}]
\]

\[
\text{linear } \leftarrow \\
\quad [\text{clause 10}]
\]

To determine if \textbf{mathmajor} is a logical consequence of the above database, we proceed as follows:

\[
\text{mathmajor } \leftarrow \text{ calculus, algebra} \\
\quad [\text{clause 1}]
\]

\[
\text{calculus } \leftarrow \text{ bascalc, advcalc} \\
\quad [\text{clause 2}]
\]

\[
\text{mathmajor } \leftarrow \text{ bascalc, advcalc, algebra} \\
\quad [\text{resolvent}]
\]
mathmajor ← bascalc, advcalc, algebra  
[resolvent]

*bascalc ← teecalc  
[clause 3]

mathmajor ← teecalc, advcalc, algebra  
[resolvent]

teecalc ←  
[clause 8]

mathmajor ← advcalc, algebra  
[resolvent]

advcalc ←  
[clause 9]

mathmajor ← algebra  
[resolvent]

*algebra ← group  
[clause 5]

mathmajor ← group  
[resolvent]

FAIL.

There are no clauses in the database that matches the last clause.

Back up to the most recent choice point (marked by an asterisk) to look for an alternative choice
This process is called \textit{BACKTRACKING}

\begin{itemize}
  \item \texttt{mathmajor ← algebra} \quad \text{[resolvent]}
  \item \texttt{algebra → group} \quad \text{[clause 5, \textit{Fail and backtrack}]}
  \item \texttt{algebra ← linear} \quad \text{[clause 6]}
  \item \texttt{mathmajor ← linear} \quad \text{[resolvent]}
  \item \texttt{linear ←} \quad \text{[clause 10]}
  \item \texttt{mathmajor ←} \quad \text{[resolvent]}
\end{itemize}

YES.

What would happen if \textit{Linear Algebra} had not been taken?

Note:

\begin{itemize}
  \item If all clauses in the database are true, then all logical consequences of the database will also be true
  \item Anything that is not a logical consequence of the database is assumed to be false
\end{itemize}
Propositions vs. Predicates

Propositions are inflexible and propositional symbols do not provide insight into their structures for further analysis.

Predicates are a better way of representing knowledge.
Example:
Representing the fact that both Socrates and Plato were men.

In propositional logic
SOCRATESMAN
PLATOMAN

are two totally different assertions.

In predicate logic
man(socrates)
man(plato)

displays the structure of the knowledge represented, and brings out similarities between Socrates and Plato.
More on predicates

A *predicate* is a function (of n arguments) which can have the value *true* or *false*
Example:
- dead(socrates, now)
- alive(plato, now)

Predicates typically express a *relationship* between their arguments.

The names *dead*, *alive* etc. are referred to as the predicate’s functors.
The *arity* of a predicate is the number of arguments that it has.
Example:  
- hastaken(X,Y)

This single predicate enables us to define a multitude of propositions as follows
- hastaken(joanne, advcalc) is *true*
- hastaken(joanne, linear) is *true*
- hastaken(joanne, group) is *false*

The terms *advcalc*, *linear*, *group*, etc,
  - no longer represent propositions
  - are now simply constants referring to a particular subject

Propositions are now represented by predicates.
Predicate Calculus

In *propositional calculus* -
the atomic propositions are propositional symbols.

In *predicate calculus* -
the atomic propositions are predicates.

We can form compound propositions using the
logical connectives exactly as before:

Example:

\[
\text{hastaken(joanne,bascalc)} \land
\text{hastaken(john,advcalc)}
\]

We can rewrite our original example using
predicates as follows:

\[
\text{hastaken(joanne,bascalc)} \iff \text{hastaken(joanne,teecalc)}
\]
\[
\text{hastaken(joanne,algebra)} \iff \text{hastaken(joanne,linear)}
\]

etc.
Why we need variables

Clauses of the form

$$\text{hastaken(joanne,bascalc)} \leftarrow \text{hastaken(joanne,teecalc)}$$

will be true, not just for Joanne, but for every other individual in the database as well.

Thus the following clauses are also true:

$$\text{hastaken(john,bascalc)} \leftarrow \text{hastaken(john,teecalc)}$$

$$\text{hastaken(john,algebra)} \leftarrow \text{hastaken(john,linear)}$$

and so on for every individual in the database.

It would be more efficient to introduce a variable $X$ and write the clauses once and for all as

$$\text{hastaken(X,bascalc)} \leftarrow \text{hastaken(X,teecalc)}$$

$$\text{hastaken(X,algebra)} \leftarrow \text{hastaken(X,linear)}$$

If the above clauses are true, this would mean that they are to be true for all possible values of $X$. 
Inference in Predicate Logic

Our *resolution* inference rule for Propositional Calculus is still valid in Predicate Calculus.

However we now have variables

Consider the following:

\[ p(X,Y) \leftarrow q(X), r(Y) \]

This is really a whole family of clauses - one for each different value that the variable(s) can take.

Suppose, we attempt to resolve the following two clauses:

\[ p(X,Y) \leftarrow q(X), s(Y) \]
\[ q(b) \leftarrow r(b) \]

In order to apply the resolution rule, the term \( q(X) \) in the first clause must match the term \( q(b) \) in the second clause.

This will be true if we *INSTANTIATE* the variable \( X \) in the first clause to the value \( b \).
Unification

This process of deciding what variable INSTANTIATIONS are required so that two terms match is called UNIFICATION

Thus, in order to UNIFY

$q(X)$ and $q(b)$

we need to instantiate $X$ to the value $b$ resulting in the following (resolution) inference

$p(b,Y) \leftarrow q(b), s(Y)$

$q(b) \leftarrow r(b)$.

-----------------------------------

$p(b,Y) \leftarrow r(b), s(Y)$
Example: The maths database described previously.

\[
\text{mathmajor}(X) \leftarrow \text{hastaken}(X,\text{calculus}), \\
\text{hastaken}(X,\text{algebra})
\]

\[
\text{hastaken}(X,\text{calculus}) \leftarrow \text{hastaken}(X,\text{bascalc}), \\
\text{hastaken}(X,\text{advcalc})
\]

\[
\text{hastaken}(X,\text{bascalc}) \leftarrow \text{hastaken}(X,\text{teecalc})
\]

\[
\text{hastaken}(X,\text{bascalc}) \leftarrow \text{hastaken}(X,\text{calcA}), \\
\text{hastaken}(X,\text{calcB})
\]

\[
\text{hastaken}(X,\text{algebra}) \leftarrow \text{hastaken}(X,\text{group})
\]

\[
\text{hastaken}(X,\text{algebra}) \leftarrow \text{hastaken}(X,\text{linear})
\]

\[
\text{hastaken}(X,\text{algebra}) \leftarrow \text{hastaken}(X,\text{discrete})
\]

\[
\text{hastaken}(\text{joanne},\text{teecalc}) \leftarrow
\]

\[
\text{hastaken}(\text{joanne},\text{advcalc}) \leftarrow
\]

\[
\text{hastaken}(\text{joanne},\text{linear}) \leftarrow
\]

\[
\text{hastaken}(\text{john},\text{calcA}) \leftarrow
\]

\[
\text{hastaken}(\text{john},\text{calcB}) \leftarrow
\]

\[
\text{hastaken}(\text{john},\text{advcalc}) \leftarrow
\]
Programming in Logic

The ability to calculate values of variables through unification allows us to program.

Resolution can be thought of as a *GOAL DIRECTED SEARCH*

\[
p(X,Y) \leftarrow q(X), r(Y,Z), s(Z)
\]

\[
p(X,Y)\text{ is the } \textit{GOAL} \text{ to be proved by satisfying the } \textit{SUBGOALS} q(X), r(Y,Z), \text{ and } s(Z)
\]

Consider what happens at a typical resolution step:

\[Q \leftarrow P, S_1, \ldots, S_n\]
\[P \leftarrow R_1, \ldots, R_m\]

\[\begin{array}{c}
Q \leftarrow R_1, \ldots, R_m, S_1, \ldots, S_n
\end{array}\]

The first clause tells us what sub-goals need to be proved in order to prove the main goal Q.

The second clause tells us what further sub-goals are required in order to prove the (first) sub-goal P.