

PROPOSITIONAL LOGIC

Introduction

- “Weak” (search-based) problem-solving does not scale to real problems.
- To succeed, problem solving needs *domain specific knowledge*.
- In search, knowledge = heuristic.
- We need to be able to represent knowledge efficiently.
- One way to do this is to use logic.

What is a Logic?

- When most people say 'logic', they mean either *propositional logic* or *first-order predicate logic*.
- However, the precise definition is quite broad, and literally hundreds of logics have been studied by philosophers, computer scientists and mathematicians.
- Any 'formal system' can be considered a logic if it has:
 - a well-defined *syntax*;
 - a well-defined *semantics*; and
 - a well-defined *proof-theory*.

- The *syntax* of a logic defines the syntactically acceptable objects of the language, which are properly called *well-formed formulae* (wff). (We shall just call them formulae.)
- The *semantics* of a logic associate each formula with a *meaning*.
- The *proof theory* is concerned with manipulating formulae according to certain rules.

Propositional Logic

- The simplest, and most abstract logic we can study is called *propositional logic*.
- **Definition:** A *proposition* is a statement that can be either *true* or *false*; it must be one or the other, and it cannot be both.
- **EXAMPLES.** The following are propositions:
 - the reactor is on;
 - the wing-flaps are up;
 - Marvin K Mooney is president.

whereas the following are not:

- are you going out somewhere?
- $2+3$

- It is possible to determine whether any given statement is a proposition by prefixing it with:

It is true that ...

and seeing whether the result makes grammatical sense.

- We now define *atomic* propositions. Intuitively, these are the set of smallest propositions.
- **Definition:** An *atomic proposition* is one whose truth or falsity does not depend on the truth or falsity of any other proposition.
- So all the above propositions are atomic.

- Now, rather than write out propositions in full, we will abbreviate them by using *propositional variables*.
- It is standard practice to use the lower-case roman letters

p, q, r, \dots

to stand for propositions.

- If we do this, we must define what we mean by writing something like:

Let p be *Marvin K Mooney is president*.

- Another alternative is to write something like *reactor_is_on*, so that the interpretation of the propositional variable becomes obvious.

The Connectives

- Now, the study of atomic propositions is pretty boring. We therefore now introduce a number of *connectives* which will allow us to build up *complex propositions*.
- The connectives we introduce are:
 - \wedge and (& or .)
 - \vee or (| or +)
 - \neg not (\sim)
 - \Rightarrow implies (\supset or \rightarrow)
 - \Leftrightarrow iff (\leftrightarrow)
- (Some books use other notations; these are given in parentheses.)

And

- Any two propositions can be combined to form a third proposition called the *conjunction* of the original propositions.
- **Definition:** If p and q are arbitrary propositions, then the *conjunction* of p and q is written

$$p \wedge q$$

and will be true iff both p and q are true.

- We can summarise the operation of \wedge in a *truth table*. The idea of a truth table for some formula is that it describes the behaviour of a formula under all possible interpretations of the primitive propositions the are included in the formula.
- If there are n different atomic propositions in some formula, then there are 2^n different lines in the truth table for that formula. (This is because each proposition can take one 1 of 2 values — *true* or *false*.)
- Let us write T for truth, and F for falsity. Then the truth table for $p \wedge q$ is:

p	q	$p \wedge q$
F	F	F
F	T	F
T	F	F
T	T	T

Or

- Any two propositions can be combined by the word 'or' to form a third proposition called the *disjunction* of the originals.
- **Definition:** If p and q are arbitrary propositions, then the *disjunction* of p and q is written

$$p \vee q$$

and will be true iff either p is true, or q is true, or both p and q are true.

- The operation of \vee is summarised in the following truth table:

p	q	$p \vee q$
F	F	F
F	T	T
T	F	T
T	T	T

- Note that this 'or' is a little different from the usual meaning we give to 'or' in everyday language.

If... Then...

- Many statements, particularly in mathematics, are of the form:

if p is true then q is true.

Another way of saying the same thing is to write:

p implies q.

- In propositional logic, we have a connective that combines two propositions into a new proposition called the *conditional*, or *implication* of the originals, that attempts to capture the sense of such a statement.

- **Definition:** If p and q are arbitrary propositions, then the *conditional* of p and q is written

$$p \Rightarrow q$$

and will be true iff either p is false or q is true.

- The truth table for \Rightarrow is:

p	q	$p \Rightarrow q$
F	F	T
F	T	T
T	F	F
T	T	T

- The \Rightarrow operator is the hardest to understand of the operators we have considered so far, and yet it is extremely important.
- If you find it difficult to understand, just remember that the $p \Rightarrow q$ means 'if p is true, then q is true'.
If p is false, then we don't care about q , and by default, make $p \Rightarrow q$ evaluate to T in this case.
- Terminology: if ϕ is the formula $p \Rightarrow q$, then p is the *antecedent* of ϕ and q is the *consequent*.

Iff

- Another common form of statement in maths is:

p is true if, and only if, q is true.

- The sense of such statements is captured using the *biconditional* operator.
- **Definition:** If p and q are arbitrary propositions, then the *biconditional* of p and q is written:

$$p \Leftrightarrow q$$

and will be true iff either:

1. p and q are both true; or
2. p and q are both false.

- The truth table for \Leftrightarrow is:

p	q	$p \Leftrightarrow q$
F	F	T
F	T	F
T	F	F
T	T	T

- If $p \Leftrightarrow q$ is true, then p and q are said to be *logically equivalent*. They will be true under exactly the same circumstances.

Not

- All of the connectives we have considered so far have been *binary*: they have taken *two* arguments.
- The final connective we consider here is *unary*. It only takes *one* argument.
- Any proposition can be prefixed by the word 'not' to form a second proposition called the *negation* of the original.

- **Definition:** If p is an arbitrary proposition then the *negation* of p is written

$$\neg p$$

and will be true iff p is false.

- Truth table for \neg :

p	$\neg p$
F	T
T	F

Comments

- We can *nest* complex formulae as deeply as we want.
- We can use *parentheses* i.e., $)$, $($, to *disambiguate* formulae.
- EXAMPLES. If p, q, r, s and t are atomic propositions, then all of the following are formulae:

$$- p \wedge q \Rightarrow r$$

$$- p \wedge (q \Rightarrow r)$$

$$- (p \wedge (q \Rightarrow r)) \vee s$$

$$- ((p \wedge (q \Rightarrow r)) \vee s) \wedge t$$

whereas none of the following is:

$$- p \wedge$$

$$- p \wedge q)$$

$$- p^{-1}$$

Tautologies & Consistency

- Given a particular formula, can you tell if it is true or not?
- No — you usually need to know the truth values of the component atomic propositions in order to be able to tell whether a formula is true.
- **Definition:** A *valuation* is a function which assigns a truth value to each primitive proposition.
- In Modula-2, we might write:

```
PROCEDURE Val(p : AtomicProp) :  
    BOOLEAN;
```

- Given a valuation, we can say for any formula whether it is true or false.
- A valuation is also known as an *interpretation*

- EXAMPLE. Suppose we have a valuation v , such that:

$$v(p) = F$$

$$v(q) = T$$

$$v(r) = F$$

Then we truth value of $(p \vee q) \Rightarrow r$ is evaluated by:

$$(v(p) \vee v(q)) \Rightarrow v(r) \tag{1}$$

$$= (F \vee T) \Rightarrow F \tag{2}$$

$$= T \Rightarrow F \tag{3}$$

$$= F \tag{4}$$

Line (3) is justified since we know that $F \vee T = T$.

Line (4) is justified since $T \Rightarrow F = F$.

If you can't see this, look at the truth tables for \vee and \Rightarrow .

- When we consider formulae in terms of interpretations, it turns out that some have interesting properties.
- **Definition:**
 1. A formula is a *tautology* iff it is true under *every* valuation;
 2. A formula is *consistent* iff it is true under *at least one* valuation;
 3. A formula is *inconsistent* iff it is not made true under *any* valuation.
- A tautology is said to be *valid*.
- A consistent formula is said to be *satisfiable*.
- An inconsistent formula is said to be *unsatisfiable*.

- **Theorem:** ϕ is a tautology iff $\neg\phi$ is unsatisfiable.
- Now, each line in the truth table of a formula corresponds to a valuation.
- So, we can use truth tables to determine whether or not formulae are tautologies.
- If every line in the truth table has value T , then the formula is a tautology.
- Also use truth-tables to determine whether or not formulae are *consistent*.

- To check for consistency, we just need to find *one* valuation that satisfies the formula.
- If this turns out to be the first line in the truth-table, we can stop looking immediately: we have a *certificate* of satisfiability.
- To check for validity, we need to examine *every* line of the truth-table.
No short cuts.
- The lesson? *Checking satisfiability is easier than checking validity.*

Syntax

- We have already informally introduced propositional logic; we now define it formally.
- To define the syntax, we must consider what symbols can appear in formulae, and the rules governing how these symbols may be put together to make acceptable formulae.
- **Definition:** Propositional logic contains the following symbols:
 1. A set of *primitive propositions*, $\Phi = \{p, q, r \dots\}$.
 2. The unary logical connective ' \neg ' (not), and binary logical connective ' \vee ' (or).
 3. The punctuation symbols ')' and '('.

- The primitive propositions will be used to represent statements such as:

I am in Brooklyn

It is raining

It is Friday 11th November 2005.

These are primitive in the sense that they are *indivisible*; we cannot break them into smaller propositions.

- The remaining logical connectives (\wedge , \Rightarrow , \Leftrightarrow) will be introduced as abbreviations.

- We now look at the rules for putting formulae together.
- **Definition:** The set \mathcal{W} , of (well formed) formulae of propositional logic, is defined by the following rules:
 1. If $p \in \Phi$, then $p \in \mathcal{W}$.
 2. If $\phi \in \mathcal{W}$, then:
$$\neg\phi \in \mathcal{W}$$
$$(\phi) \in \mathcal{W}$$
 3. If $\phi \in \mathcal{W}$ and $\psi \in \mathcal{W}$, then $\phi \vee \psi \in \mathcal{W}$.

- The remaining connectives are defined by:

$$\phi \wedge \psi = \neg(\neg\phi \vee \neg\psi)$$

$$\phi \Rightarrow \psi = (\neg\phi) \vee \psi$$

$$\phi \Leftrightarrow \psi = (\phi \Rightarrow \psi) \wedge (\psi \Rightarrow \phi)$$

- These connectives are interpreted:

\wedge And

\Rightarrow Implies (if... then ...)

\Leftrightarrow If, and only if

- This concludes the formal definition of syntax.

Semantics

- We now look at the more difficult issue of *semantics*, or *meaning*.
- What does a proposition *mean*?
- That is, when we write

It is raining.

what does it mean?

From the point of view of logic, this statement is a *proposition*: something that is either \top or \perp .

- *The meaning of a primitive proposition is thus either \top or \perp .*
- In the same way, the meaning of a formula of propositional logic is either \top or \perp .

- QUESTION: How can we tell whether a formula is \top or \perp ?
- For example, consider the formula

$$(p \wedge q) \Rightarrow r$$

Is this \top ?

- The answer must be: *possibly*. It depends on your *interpretation* of the primitive propositions p , q and r .
- The notion of an interpretation is easily formalised.
- **Definition:** An *interpretation* for propositional logic is a function

$$\pi : \Phi \mapsto \{T, F\}$$

which assigns T (true) or F (false) to every primitive proposition.

- But an interpretation only gives us the meaning of primitive propositions; what about complex propositions — arbitrary formulae?
- We use some *rules* which tell us how to obtain the meaning of an arbitrary formulae, given some interpretation.
- Before presenting these rules, we introduce a symbol: \models . If π is an interpretation, and ϕ is a formula, then the expression

$$\pi \models \phi$$

will be used to represent the fact that ϕ is \top under the interpretation π .

Alternatively, if $\pi \models \phi$, then we say that:

- π *satisfies* ϕ ; or
- π *models* ϕ .
- The symbol \models is called the *semantic turnstile*.

- The rule for primitive propositions is quite simple. If $p \in \Phi$ then

$$\pi \models p \text{ iff } \pi(p) = T.$$

- The remaining rules are defined *recursively*.
- The rule for \neg :

$$\pi \models \neg\phi \text{ iff } \pi \not\models \phi$$

(where $\not\models$ means 'does not satisfy'.)

- The rule for \vee :

$$\pi \models \phi \vee \psi \text{ iff } \pi \models \phi \text{ or } \pi \models \psi$$

- Since these are the only connectives of the language, these are the only semantic rules we need.
- Since:

$$\phi \Rightarrow \psi$$

is defined as:

$$(\neg\phi) \vee \psi$$

it follows that:

$$\pi \models \phi \Rightarrow \psi \text{ iff } \pi \not\models \phi \text{ or } \pi \models \psi$$

- And similarly for the other connectives we defined.

- If we are given an interpretation π and a formula ϕ , it is a simple (if tedious) matter to determine whether $\pi \models \phi$.
- We just apply the rules above, which eventually bottom out of the recursion into establishing if some proposition is true or not.
- So for:

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

we first establish if $p \vee q$ or $q \vee r$ are true and then work up to the compound proposition.

Summary

- This lecture started to look at logic from the standpoint of artificial intelligence.
- The main use of logic from this perspective is as a means of knowledge representation.
- We introduced the basics of propositional logic, and talked about some of the properties of sentences in logic.
- We also looked at a formal definition of syntax and semantics, and the semantic approach to inference.
- The next lecture will look at the syntactic approach—proof theory.