

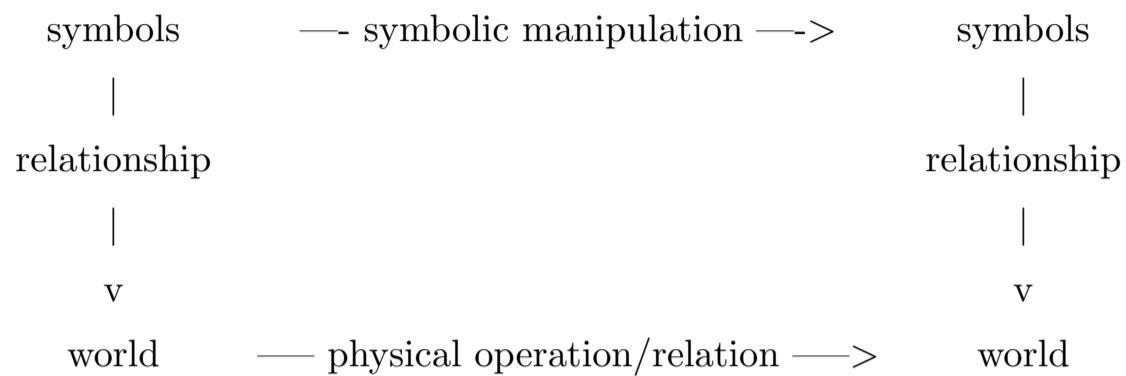
Lecture 2: Propositional Logic Review

Overview:

- syntax of propositional logic, parse trees
- translation from English
- truth functional semantics of propositional logic, truth tables
- validity, satisfiability and entailment
- logical equivalences
- *applications*: digital circuits, constraint satisfaction problems, reasoning about specifications

Reading for this lecture: Huth & Ryan, sections 1.1, 1.3, 1.4.1

Reading for the next week: Huth & Ryan, pp. 1-51



The main relationship between symbols and the world of concern in logic is that of a *sentence of a language* being *true* in the world.

A sentence of a natural language (like English, Chinese, Warlpiri) is *declarative*, or a *proposition*, if it can be meaningfully be said to be either true or false.

Examples:

- Richard Nixon was president of Equador.
- The square root of 16 is 4.
- This program gets stuck in an infinite loop if you input 3.
- Whatever list of numbers you give as input to this program, it outputs the same list but in increasing order.
- $x^n + y^n = z^n$ has no nontrivial integer solutions for $n > 2$.

The following are *not* declarative sentences of English:

- Gubble gimble goo
- For Pete's sake, take out the garbage!
- Did you watch South Park last night?
- Please waive the prerequisites for this subject for me.

Declarative sentences in natural languages can be *compound* sentences, built out of other sentences.

Propositional Logic is a formal representation of some constructions for which the truth value of the compound sentence can be determined from the truth value of its components.

- Cook is a bit of a Romeo *and* Kenny is always getting killed.
- Either Bill is a liar *or* Hillary is innocent of Whitewater.
- *It is not the case that* this program always halts.

Not all constructions of natural language are truth-functional:

- *Hillary suspects that* Starr has the memo.
- *Cook said* they killed Kenny.
- This program always halts *because* it contains no loops.
- The disk crashed *after* I saved my file.

Connectives of Propositional Logic and their Truth Tables

\wedge “and”, “but”, “;”, “:”

\vee “or”, “either ... or ...”

\neg “not”, “it is not the case that”

A	B	$A \wedge B$
F	F	F
F	T	F
T	F	F
T	T	T

A	B	$A \vee B$
F	F	F
F	T	T
T	F	T
T	T	T

A	$\neg A$
F	T
T	F

Somewhat more controversially, consider the following constructions:

- if A then B
- A only if B
- B if A
- A implies B
- it follows from A that B
- whenever A, B
- A is a sufficient condition for B
- B is a necessary condition for A

Each has the property that if true, and A is true, then B is true.

We can *approximate* the English meaning of these by “not (A and not B)”, written $A \longrightarrow B$, which has the following truth table:

A	B	$A \longrightarrow B$
F	F	T
F	T	T
T	F	F
T	T	T

While only an approximation to the English, 100+ years of experience have shown this to be adequate for capturing *mathematical reasoning*. (Moral: mathematical reasoning does not need all the features of English.)

Unless

A unless B can be approximated as $\neg B \longrightarrow A$

E.g. I go swimming unless it rains = If it is not raining I go swimming.

Correctness the translation is perhaps easier to see in:

I don't go swimming unless the sun shines = If the sun does not shine then I don't go swimming.

Note that "I go swimming unless it rains, but sometimes I swim even though it is raining" makes sense, so the translation of "A unless B" should not imply $B \longrightarrow \neg A$.

Propositional Logic as a Formal Language

A *string* is just a sequence of symbols.

Let $Prop = \{p, q, r, \dots\}$ be a set of basic propositional letters.

The set of formulae of propositional logic is the smallest set such that

- If $x \in Prop$ then x is a formula,
- If ϕ is a formula then so is $\neg\phi$
- If ϕ and ψ are formulae, then so are $(\phi \wedge \psi)$, $(\phi \vee \psi)$, $(\phi \longrightarrow \psi)$, and $(\phi \longleftrightarrow \psi)$.

Convention: we often drop parentheses when there is no ambiguity. \neg binds more tightly than \wedge and \vee , which in turn bind more tightly than \longrightarrow and \longleftrightarrow .

Applications of Propositional Logic I: Digital Circuits

A formula, can via its parse tree, be viewed as defining a digital circuit, which computes a boolean function of the input propositions. The function is given by the truth table of the formula.

Logical Equivalence

Two formulas ϕ, ψ are *logically equivalent*, denoted $\phi \equiv \psi$ if they have the same truth value for all values of their basic propositions.

Application: If ϕ and ψ are two formulae such that $\phi \equiv \psi$, then the circuits corresponding to ϕ and ψ compute the same function. Thus, proving equivalence of formulas can be used to optimize circuits.

Some well known equivalences:

Excluded Middle $p \vee \neg p \equiv \top$

Contradiction $p \wedge \neg p \equiv \perp$

Identity $p \vee \perp \equiv p$

$$p \wedge \top \equiv p$$

Domination $p \vee \top \equiv \top$

$$p \wedge \perp \equiv \perp$$

Idempotence $p \vee p \equiv p$

$$p \wedge p \equiv p$$

Double Negation $\neg\neg p \equiv p$

Commutativity $p \vee q \equiv q \vee p$

$$p \wedge q \equiv q \wedge p$$

Associativity $(p \vee q) \vee r \equiv p \vee (q \vee r)$

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

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

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

De Morgan's laws $\neg(p \wedge q) \equiv \neg p \vee \neg q$

$$\neg(p \vee q) \equiv \neg p \wedge \neg q$$

Implication $p \longrightarrow q \equiv \neg p \vee q$

$$p \longleftrightarrow q \equiv (p \longrightarrow q) \wedge (q \longrightarrow p)$$

Example:

$$\begin{aligned} & ((r \wedge \neg p) \vee (r \wedge q)) \vee ((\neg r \wedge \neg p) \vee (\neg r \wedge q)) \\ \equiv & ((r \wedge (\neg p \vee q)) \vee (\neg r \wedge (\neg p \vee q))) && \text{Assoc.} \\ \equiv & (r \vee \neg r) \wedge (\neg p \vee q) && \text{Assoc., Commut.} \\ \equiv & \top \wedge (\neg p \vee q) && \text{Excl. Mid.} \\ \equiv & (\neg p \vee q) && \text{Ident.} \end{aligned}$$

Examples

A formula is *satisfiable*, if it evaluates to T for some assignment of truth value to its basic propositions.

Example:

A	B	$\neg(A \longrightarrow B)$
F	F	F
F	T	F
T	F	T
T	T	F

Applications of Propositional Logic II: Constraint Satisfaction Problems

These are problems such as timetabling, activity planning, etc. Many can be understood as showing that a formula is satisfiable.

Example: You are planning a party, but your friends are a bit touchy about who will be there.

1. If John comes, he will get very hostile if Sarah is there.
2. Sarah will only come if Kim will be there also.
3. Kim says she will not come unless John does.

Who can you invite without making someone unhappy?

Translation to logic: let $J, (S, K)$ represent “John (Sarah, Kim) comes to the party”. Then the constraints are:

1. $J \longrightarrow \neg S$
2. $S \longrightarrow K$
3. $K \longrightarrow J$

Thus, for a successful party to be possible, we want the formula $\phi = (J \longrightarrow \neg S) \wedge (S \longrightarrow K) \wedge (K \longrightarrow J)$ to be satisfiable.

Truth values for J, S, K making this true are called *satisfying assignments*.

We figure out where the conjuncts are false, below. (so blank = T)

J	K	S	$J \longrightarrow \neg S$	$S \longrightarrow K$	$K \longrightarrow J$	ϕ
F	F	F				
F	F	T		F		F
F	T	F			F	F
F	T	T			F	F
T	F	F				
T	F	T	F	F		F
T	T	F				
T	T	T	F			F

Conclusion: a party satisfying the constraints can be held. Invite nobody, or invite John only, or invite Kim and John.