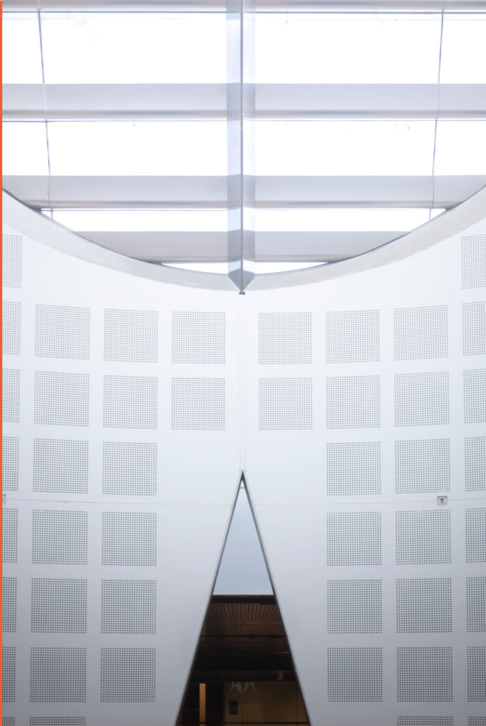
COMP2022|2922 Models of Computation

Propositional Logic
Logical Consequences

Sasha Rubin

October 6, 2022

THE UNIVERSITY OF SYDNEY



Agenda

1. Logical consequence
   - Useful for telling if an argument/reasoning is logically correct.
2. Logical equivalence
   - Useful for understanding if two formulas mean the same thing
   - Gives us normal forms

An argument is a statement of the form "if these facts are true, then that fact must be true". Logic allows us to formalise what it means for an argument to be logically correct.

Logically correct arguments

Example

Why is the following argument logically correct?

1. If $x = 5$ then $z = 4$
2. $x = 5$
3. Conclude $z = 4$

Vote now! (on mentimeter)

It is a particular instance of which rule of inference:

1. Negation elimination
2. Conjunction elimination
3. Disjunction elimination
4. Implication elimination

Logically correct arguments

Example

Why is the following argument logically correct?

1. Assuming it is hot, I wear a hat
2. Assuming it is windy, I wear a hat
3. It is either hot or windy
4. Conclude I wear a hat

Vote now! (on mentimeter)

It is a particular instance of which rule of inference:

1. Negation elimination
2. Conjunction elimination
3. Disjunction elimination
4. Implication elimination

Logically correct arguments

Example

Why is the following argument logically correct?

1. If it is raining then I take an umbrella.
2. I take an umbrella.
3. Conclude it is raining.

Vote now! (on mentimeter)

It is a particular instance of which rule of inference:

1. Negation elimination
2. Conjunction elimination
3. Disjunction elimination
4. Implication elimination

It is not a logically correct argument!

It is an instance of

$$(p \to q), q \vdash p$$

which cannot be proven! This is called the error of the converse.

Logically correct arguments

Example

Is the following argument logically correct?

1. If it is raining then I take an umbrella.
2. It is not raining.
3. Conclude I do not take an umbrella.

This would be an instance of

$$(p \to q), \neg p \vdash \neg q$$

which cannot be proven! This is called the error of the inverse.

Logical consequence

There is another way to formalise what a logically correct argument looks like.

Illustration

Why is it that we cannot deduce q by assuming $(p \to q)$ and $\neg p$?

- Because it is possible to have that both $(p \to q)$ and $\neg p$ are true, but q is false.
- For instance, take p to be false and q to be false.
- This corresponds to the scenario (in the last example) in which it is not raining and i do not take an umbrella.

This type of semantic reasoning is so important, we now give it its own definition.

Logical consequence

Definition
Say that F is a logical consequence of $E_1, \cdots, E_k$, if every assignment that makes all of the formulas E_i (for $1 \leq i \leq k$) true also makes F true. We write this as

$$E_1, \cdots, E_k \models F$$

In case $k = 0$ we write $\models F$, which means that every assignment makes F true, i.e., that F is valid.

Logical consequence

Show that $(p \to q), \neg q \models \neg p$.

Logical consequence

Show that $(p \to q), \neg p \not\models \neg q$.

Logical consequence: application

– Logic is used to study correct logical argumentation and reasoning.
– This is useful for coding, reasoning about the world, mathematics, etc.

Natural Deduction

- What is the connection between provability $\vdash$ and logical consequence $\models$?
- Although they are defined differently (syntactically vs semantically), they give us the same consequences!

Theorem (ND is sound)

if $E_1 \cdots, E_k \vdash F$ *then* $E_1, \cdots, E_k \models F$*.*

i.e., ND can prove only logical consequences.

Theorem (ND is complete)

If $E_1, \cdots, E_k \models F$ *then* $E_1, \cdots, E_k \vdash F$*.*

i.e., ND can prove all logical consequences.

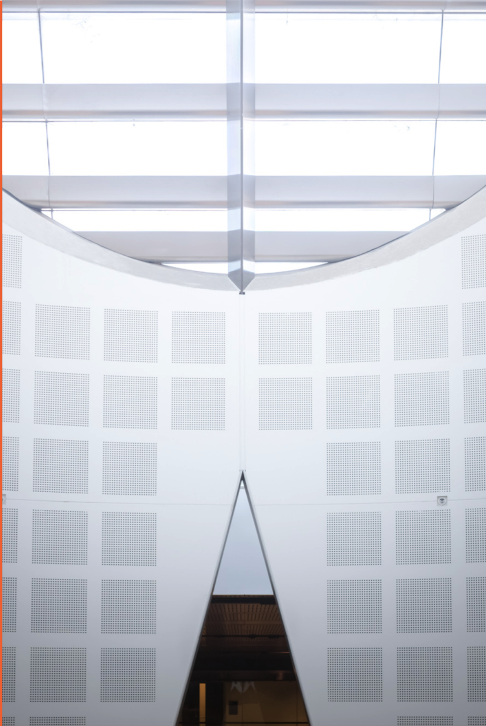
COMP2022|2922 Models of Computation

Propositional Logic
Logical Equivalences

Sasha Rubin

October 6, 2022

THE UNIVERSITY OF SYDNEY



Formulas that "mean the same thing" are called equivalent. We now study common equivalences, also called laws.

Equivalences

- Although the formulas $(A \wedge B)$ and $(B \wedge A)$ are syntactically different formulas, they mean the same thing. This is captured by the fact that they are logically equivalent.
- Two formulas F and G are logically equivalent if they are assigned the same truth value under every assignment. This is written $F \equiv G$.

Equivalences

Are the following pairs of formulas equivalent?

1. $(p \wedge q)$, $(q \wedge p)$?
2. p , q?
3. $\top$, $(p \rightarrow p)$?
4. $(p \rightarrow q)$, $(q \rightarrow p)$?
5. $(p \vee \neg p)$, $(q \vee \neg q)$?
6. $(p \rightarrow q)$, $(\neg q \rightarrow \neg p)$?

A Table of Equivalences

(Idempotent Laws)	$F \equiv (F \wedge F)$
	$F \equiv (F \vee F)$
(Commutative Laws)	$(F \wedge G) \equiv (G \wedge F)$
	$(F \vee G) \equiv (G \vee F)$
(Associative Laws)	$(F \wedge (G \wedge H)) \equiv ((F \wedge G) \wedge H)$
	$(F \vee (G \vee H)) \equiv ((F \vee G) \vee H)$
(Absorption Laws)	$(F \wedge (F \vee G)) \equiv F$
	$(F \vee (F \wedge G)) \equiv F$
(Distributive Laws)	$(F \wedge (G \vee H)) \equiv ((F \wedge G) \vee (F \wedge H))$
	$(F \vee (G \wedge H)) \equiv ((F \vee G) \wedge (F \vee H))$
(de Morgan's Laws)	$\neg(F \wedge G) \equiv (\neg F \vee \neg G)$
	$\neg(F \vee G) \equiv (\neg F \wedge \neg G)$
(Double Negation Law)	$\neg\neg F \equiv F$
(Validity Law)	$(F \vee \top) \equiv \top$
	$(F \wedge \top) \equiv F$
(Unsatisfiability Law)	$(F \vee \bot) \equiv F$
	$(F \wedge \bot) \equiv \bot$
(Constant Laws)	$\top \equiv (F \vee \neg F)$
	$\bot \equiv (F \wedge \neg F)$
(Negating constants Laws)	$\neg\top \equiv \bot$
	$\neg\bot \equiv \top$
(Conditional Law)	$(F \to G) \equiv (\neg F \vee G)$
(Bi-conditional Law)	$(F \leftrightarrow G) \equiv ((F \to G) \wedge (G \to F))$

Equivalences

There are a number of ways to verify an equivalence.

1. Use truth tables.
2. Deduce it from other equivalences.
3. Use ND.

Equivalences

Verify the following equivalence using other equivalences:

$$(p \to q) \equiv (\neg q \to \neg p)$$

$$\begin{aligned}
(p \to q) &\equiv (\neg p \vee q) && \text{Conditional Law} \\
&\equiv (q \vee \neg p) && \text{Commutative Law for } \vee \\
&\equiv (\neg\neg q \vee \neg p) && \text{Double Negation Law} \\
&\equiv (\neg q \to \neg p) && \text{Conditional Law}
\end{aligned}$$

Aside. What justifies the use of Double Negation inside the formula?
Fact. If F is a subformula of H, and $F \equiv G$, then H is equivalent to formulas that result by substituting an occurrence of F in H by G. This is called the Substitution Rule.

Equivalences: applications (1)

- Equivalences can be used to rewrite a formula into an equivalent one having a special structure, called a normal form.
- We will shortly study a particular useful normal form.

Equivalences: applications (2)

We originally defined the syntax of propositional formulas to only use the connectives $\wedge, \vee, \neg$.

We extended this to $\rightarrow, \leftrightarrow, \top, \bot$.

Which connectives are really needed?

Equivalences: applications (2)

1. Every propositional formula is logically equivalent to a formula which only contains the connectives $\neg$ and $\wedge$.

   e.g., $(\neg p \vee q)$ is logically equivalent to $\neg(p \wedge \neg q)$.

2. Every propositional formula is logically equivalent to a formula which only contains the connectives $\neg$ and $\rightarrow$.

   e.g., $(p \wedge q)$ is logically equivalent to $\neg(p \rightarrow \neg q)$.

3. Every prop formula is logically equivalent to a formula which only contains the connective $p \uparrow q$ defined as $\neg(p \wedge q)$.
   - This is called the NAND ("not and") connective.
   - Then $p \uparrow p$ is logically equivalent to $\neg p$.
   - So $(p \uparrow q) \uparrow (p \uparrow q)$ is logically equivalent to $p \wedge q$.

Formulas vs statements about formulas

Q. What is the difference between $E \leftrightarrow F$ and $E \equiv F$?

- $E \leftrightarrow F$ is a logical formula.
- $E \equiv F$ is a mathematical statement about logical formulas.

Note: $E \equiv F$ means exactly the same thing as "$(E \leftrightarrow F)$ is valid".

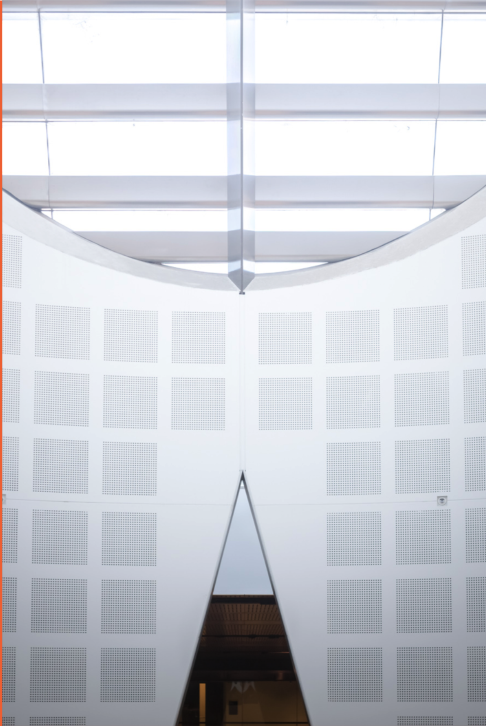
COMP2022|2922
Models of Computation

Propositional Logic: Normal forms and modeling problems in logic

Sasha Rubin

October 6, 2022

THE UNIVERSITY OF SYDNEY



Agenda

1. Normal forms (NNF, CNF)
2. Modeling problems as satisfiability problems

In this lecture we use the original syntax of propositional logic, i.e., the only operators are $\neg, \wedge$ and $\vee$. If you have a formula with any other operator, you should first get rid of it by replacing by equivalent formulas. E.g.,

- replace $(p \rightarrow q)$ by $(\neg p \vee q)$
- replace $\perp$ by $(p \wedge \neg p)$

Normal forms: NNF

Definition

A formula F is in negation normal form (NNF) if negations only occur immediately in front of atoms.

Vote now! (on mentimeter)

Which of the following are in NNF?

1. p
2. $\neg p$
3. $\neg\neg p$
4. $\neg\neg\neg p$
5. $(\neg q \vee p)$
6. $\neg(q \wedge \neg p)$

Normal forms: NNF

Theorem

For every formula F there is an equivalent formula in NNF.

Algorithm ("push negation inwards")

Here is the algorithm: substitute in F every occurence of a subformula of the form

$$\begin{array}{rr}\neg\neg G \text{ by } G & \text{Double Negation Law} \\ \neg(G \wedge H) \text{ by } (\neg G \vee \neg H) & \text{de Morgan's Law} \\ \neg(G \vee H) \text{ by } (\neg G \wedge \neg H) & \text{de Morgan's Law}\end{array}$$

until no such subformulas occur, and return the result.

Why is this algorithm correct?

Normal forms: NNF

Example

Put $\neg(\neg p \wedge (\neg(r \wedge s) \vee q))$ into NNF.

Normal forms: CNF

Definition

- A literal is an atomic formula or the negation of an atomic formula.

  i.e., a literal has the form p or $\neg p$; note that $\neg\neg p$ is not a literal.
- A clause is a disjunction of literals.

  e.g., both $(p \vee \neg q \vee r)$ and $\neg r$ are clauses.
  but $(p \vee q) \wedge r$ is not a clause.
- A formula F is in conjunctive normal form (CNF) if it is a conjunction of clauses.

Self test

Which of the following are in CNF?

Vote now! (on mentimeter)

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

Say we have three variables x, y, z.

Write a formula in CNF that says that "not all variables are true, and, not all variables are false".

$$(\neg x \vee \neg y \vee \neg z) \wedge (x \vee y \vee z)$$

Write $(x \leftrightarrow y)$ in CNF.

- This formula says that x, y have the same value.
- Its negation says that x, y have different values, which we can easily write:

$$(x \wedge \neg y) \vee (\neg x \wedge y)$$

- So let's negate to get back to the meaning we want:

$$\begin{aligned}
&\neg((x \wedge \neg y) \vee (\neg x \wedge y)) \\
&\qquad \equiv \neg(x \wedge \neg y) \wedge \neg(\neg x \wedge y) \\
&\qquad \equiv (\neg x \vee \neg\neg y) \wedge (\neg\neg x \vee \neg y) \\
&\qquad \equiv (\neg x \vee y) \wedge (x \vee \neg y)
\end{aligned}$$

Which is in CNF!

This process, of negating, writing a formula, negating again, and simplifying I call the *duality trick*.

Normal forms: CNF

Theorem

For every formula F there is an equivalent formula in CNF.

Proof

Here is the algorithm:

1. Put F in NNF, call it F'.
2. Substitute in F' each occurrence of a subformula of the form

$$((H \wedge I) \vee G) \text{ or } (G \vee (H \wedge I)) \qquad \text{Commutative Law}$$
$$\text{by } ((G \vee H) \wedge (G \vee I)) \qquad \text{Distributive Law}$$

   until no such subformulas occur, and return the result.

Why is the algorithm correct?

Normal forms

Why is CNF important?

- It allows one to restrict to formulas with a uniform structure.
- It is used in practice... there are many tools for solving the satisfiability problem, that take CNF formulas as input.

Solving problems with logic

CLIQUE: given an undirected graph (V, E) and an integer K, decide if it has a clique of size at least K.

- A clique (aka complete graph) of an undirected graph (V, E) is a set $C \subseteq V$ of vertices such that every two (distinct) vertices in C are adjacent.
- The size of C is the number of vertices in it, written $|C|$.

Although this problem can be solved using graph-algorithms, we will solve it using logic!

Solving CLIQUE with logic

Here are the steps:

1. Given input: graph (V, E) and number K
2. Encode the input as a formula $\Phi_{V,E,K}$.
3. Check if the formula $\Phi_{V,E,K}$ is satisfiable.
4. If it is satisfiable, return "Yes, there is a clique of size K in the graph", otherwise return "No".

Solving CLIQUE with logic

Given a graph (V, E) and $K \geq 1$.

Idea. Introduce one variable for each vertex, say variable x_i for vertex $i \in V$, and write a formula expressing "the true variables form a clique in (V, E) of size K."

i.e.,

1. There are exactly K many true variables.
2. Every pair of true variables correspond to an edge in the graph.

Then: the satisfying assignments of the formula identify the cliques of size K in (V, E)!

So: checking if the formula is satisfiable will check if there is a clique of size K in (V, E).

Solving CLIQUE with logic

Express "there are exactly K many true variables"?

For a set $S \subseteq \{1, 2, \cdots, |V|\}$ of indices:

- $\bigwedge_{i \in S} x_i$ says that all the variables indexed by S are true.
- $\bigwedge_{i \notin S} \neg x_i$ says that all the variables not indexed by S are false.
- So

$$\bigvee_{|S|=K} (\wedge_{i \in S} x_i) \wedge (\wedge_{i \notin S} \neg x_i)$$

  says what we want.

Solving CLIQUE with logic

Express "every pair of true variables corresponds to an edge"?

$$? \bigwedge_{(i,j)\in E} x_i \wedge x_j$$

- says "edges are incident with two true variables".
- but, this forces a vertex to be true if it is incident with an edge.

$$? \bigwedge_{(i,j)\notin E} \neg(x_i \wedge x_j)$$

- says "non-edges are not incident with two true variables".
- which is the same as saying that "two true variables are incident to an edge" (which is what we want!)

Solving CLIQUE with logic

So, the formula encoding a graph (V, E) and size K is:

$$\bigvee_{|S|=K} (\wedge_{i \in S} x_i) \wedge (\wedge_{i \notin S} \neg x_i)$$

$$\wedge$$

$$\bigwedge_{(i,j) \notin E} \neg(x_i \wedge x_j)$$

How big is this formula? $O(|V|^K + |V|^2)$, which is exponential in the size of the input.

In the tutorial you will explore a different encoding which is polynomial.