Metatheory for Propositional Logic

PHI 201 – Introductory Logic

Week 13

Review

- We have formulated a theory *about* propositional logic, and now we're proving some facts.
- Our new inference rule is mathematical induction — usually on the construction of sentences, or on the construction of sequents.
- E.g., in the previous lecture, I showed that every sentence is equivalent to one in which just $\vee$ and $\neg$ occur.

Soundness

Theorem. Every line in a correctly written proof is semantically valid. That is, the sentence in the center column is a semantic consequence of the dependency sentences.

Method of proof:

1. Show that Rule of Assumptions lines are semantically valid.
2. Show that the other inference rules transform semantically valid lines to semantically valid lines.

Induction MP

Suppose that $\Gamma, \Delta \vdash \psi$ is derived from $\Gamma \vdash \varphi \to \psi$ and $\Delta \vdash \varphi$.

$$\begin{array}{rcl} \Gamma & (a) & \varphi \to \psi \\ \Delta & (b) & \varphi \\ \Gamma, \Delta & (c) & \psi \end{array}$$

Suppose that (a) and (b) are semantically valid.

Induction MP

Let v be an arbitrary valuation, and suppose that v assigns 1 to all elements of Γ, Δ. Since line (a) is valid, $v(\varphi \to \psi) = 1$. Since line (b) is valid, $v(\varphi) = 1$. By the truth table for $\to$, it follows that $v(\psi) = 1$. Since v was an arbitrary valuation, any valuation that assigns 1 to all elements of Γ, Δ also assigns 1 to ψ. Therefore, line (c) is semantically valid.

Induction RA

Suppose that Δ' is derived from $\varphi \vdash \varphi$ and $\Delta \vdash \bot$.

$$\begin{array}{llll} a & (a) & \varphi & \mathsf{A} \\ \Delta & (b) & \bot & \\ \Delta' & (c) & \neg\varphi & \end{array}$$

Suppose that (a) and (b) are semantically valid.

Induction RA

Let v be an arbitrary valuation, and suppose that v assigns 1 to every element of Δ'. Since (b) is valid, v does not assign 1 to every element of Δ. Therefore, $v(\varphi) = 0$, since φ is only thing in Δ that is not in Δ'. Therefore, $v(\neg\varphi) = 1$. Since v was an arbitrary valuation, every valuation that assigns 1 to all elements of Δ' also assigns 1 to $\neg\varphi$, and line (c) is valid.

Disjunctive normal form

Goal: Disjunctive Normal Form (DNF)

DNF: A sentence is in disjunctive normal form if it is a disjunction of conjunctions of literals.

- A **literal** is either an atomic sentence (e.g. P, Q, R) or the negation of an atomic sentence (e.g. $\neg P$).
- A **conjunction of literals** has the form

  $$L_1 \wedge L_2 \wedge \cdots \wedge L_n,$$

  where each L_i is a literal.
- A sentence is in **DNF** if it has the form

  $$C_1 \vee C_2 \vee \cdots \vee C_k,$$

  where each C_j is a conjunction of literals (or a single literal).

Fact 1: Every sentence is provably equivalent to a sentence in DNF.

Fact 2: A DNF sentence $C_1 \vee \cdots \vee C_n$ is a semantic tautology iff for each elementary conjunction E, there is a C_i such that $E \vdash C_i$.

DNF and truth tables

You can “guess” a DNF equivalent of a sentence by looking at its truth table and taking a disjunction of all the rows in which its true. For example:

P	Q	φ
1	0	1

Truth Table for $P \to Q$

P	Q	$P \to Q$	$(P \wedge Q) \vee (\neg P \wedge Q) \vee (\neg P \wedge \neg Q)$
1	1	1	
1	0	0	
0	1	1	
0	0	1	

DNF algorithm: High-level strategy

Given any sentence φ built from $\wedge, \vee, \neg, \to$ and atomic $P, Q, R, \ldots$:

1. Eliminate all occurrences of $\to$.
2. Push all occurrences of $\neg$ inwards so that they apply only to atomic sentences.
3. Distribute $\wedge$ over $\vee$ to obtain a disjunction of conjunctions.
4. Clean up: remove unnecessary parentheses, reorder conjuncts/disjuncts, and combine duplicates if desired.

Step 1: Eliminate conditionals

Replace every occurrence of $A \to B$ with $\neg A \vee B$.

- Do this recursively on all subformulas:

$$(\varphi \to \psi) \wedge (\chi \to \theta) \;\rightsquigarrow\; (\neg\varphi \vee \psi) \wedge (\neg\chi \vee \theta).$$

- After this step, your sentence uses only $\wedge, \vee, \neg$ and atomic letters.

Step 2: Push Negations Inward

Use these equivalences repeatedly until $\neg$ appears only directly in front of atomic sentences:

$$\neg\neg A \equiv A \qquad \neg(A \wedge B) \equiv \neg A \vee \neg B \qquad \neg(A \vee B) \equiv \neg A \wedge \neg B$$

- Apply these rules from the outside in, simplifying as you go.
- After this step, the sentence is built from $\wedge$, $\vee$ and literals (atoms or negated atoms).

Step 3: Distribute $\wedge$ over $\vee$

To get a disjunction of conjunctions, repeatedly use the distributive laws:

$$A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$$

$$(A \vee B) \wedge C \equiv (A \wedge C) \vee (B \wedge C)$$

- Whenever you see a conjunction whose parts contain disjunctions, distribute.
- Also use associativity and commutativity of $\wedge, \vee$ to rearrange and "flatten":
  $A \vee (B \vee C) \equiv A \vee B \vee C, \qquad A \wedge (B \wedge C) \equiv A \wedge B \wedge C.$

Step 4: Cleanup to Get DNF

After distribution, your sentence should be a disjunction of conjunctions of literals. Then:

- Remove redundant parentheses using associativity.
- Optionally, reorder literals and conjunctions to a standard order (e.g. alphabetically).
- Optionally, simplify obvious redundancies, e.g. $(P \land P \land Q) \equiv (P \land Q)$

The resulting sentence is in disjunctive normal form and is logically equivalent to the original sentence.

Worked Example: From Formula to DNF

Start with $(P \to Q) \land \neg R$

1. **Eliminate $\to$:**

$$(P \to Q) \land \neg R \;\equiv\; (\neg P \lor Q) \land \neg R.$$

2. **Push negations inward:** nothing to do (already on atoms).
3. **Distribute $\land$ over $\lor$:**

$$(\neg P \lor Q) \land \neg R \;\equiv\; (\neg P \land \neg R) \lor (Q \land \neg R).$$

Now we have a disjunction of conjunctions of literals, which is in DNF.

Algorithm in Pseudocode

Input: sentence φ built from $\wedge, \vee, \neg, \rightarrow$ and atomic $P, Q, R, \ldots$

1. **ElimCond**(φ): recursively replace each subformula of the form $(A \rightarrow B)$ by $(\neg A \vee B)$.
2. **PushNeg**(φ): recursively apply $\neg\neg A \equiv A$, $\neg(A \wedge B) \equiv \neg A \vee \neg B$, $\neg(A \vee B) \equiv \neg A \wedge \neg B$ until every $\neg$ is on an atom.
3. **Distribute**(φ): recursively apply the distributive laws to move all $\wedge$ inside all $\vee$.
4. **Output** the resulting disjunction of conjunctions of literals as the DNF of the original sentence.

Completeness

Substitution theorem

For a formula φ, let φ' denote the result of uniformly substituting formulas for the atomic sentences that occur in φ. We say that φ' is an substitution instance of φ.

Proposition. If $\varphi_1, \ldots, \varphi_n \vdash \psi$ then $\varphi'_1, \ldots, \varphi'_n \vdash \psi'$.

Proposition. If φ is not provable, then it has a substitution instance φ' such that $\vdash \neg\varphi'$.

By the DNF theorem, φ is provably equivalent to a sentence $C_1 \vee \cdots \vee C_n$, where each C_i is a consistent conjunction of literals.

It's not hard to see that $\vdash E_1 \vee \cdots \vee E_m$, where the Es are an exhaustive set of elementary conjunctions.

If each E_j entailed some C_i, then $C_1 \vee \cdots \vee C_n$ would be provable.

Since φ is not provable, there is an elementary conjunction E that does not imply any C_i.

$E \vdash \neg C_i$

A substitution that takes E to something provable will take each C_i to something whose negation is provable.

$$E \vdash \neg C_i \implies \top \vdash \neg C_i'$$

Therefore $\vdash \neg(C_1' \vee \cdots \vee C_n')$, and hence $\vdash \neg\varphi'$.

Theorem. If φ is not provable, then there is a valuation v such that $v(\varphi) = 0$.

Take the elementary conjunction E from the previous argument and use it to define v.

$v(C_i) = 0$ for $i = 1, \dots, n$. Therefore $v(C_1 \vee \cdots \vee C_n) = 0$.

Since $\varphi \vdash C_1 \vee \cdots \vee C_n$, soundness implies that $v(\varphi) = 0$.

Corollary. If $\varphi_1, \ldots, \varphi_n \nvdash \psi$, then there is a valuation v such that $v(\varphi_i) = 1$ and $v(\psi) = 0$.

If $\vdash (\varphi_1 \wedge \cdots \wedge \varphi_n) \rightarrow \psi$, then $\wedge$I and MP give $\varphi_1, \ldots, \varphi_n \vdash \psi$. So $(\varphi_1 \wedge \cdots \wedge \varphi_n) \rightarrow \psi$ is not provable. By the previous theorem, there is a valuation v such that $v((\varphi_1 \wedge \cdots \wedge \varphi_n) \rightarrow \psi) = 0$. By the truth-tables for $\wedge$ and $\rightarrow$, it follows that $v(\varphi_i) = 1$ for $i = 1, \ldots, n$, while $v(\psi) = 0$.