CSC 438F/2404F **Notes** **Fall, 2006**

S. Cook

REFERENCES

The first two references have especially influenced these notes and are cited from time to time:

[Buss] Samuel Buss: Chapter I: An introduction to proof theory, in **Handbook of Proof Theory**, Samuel Buss Ed., Elsevier, 1998, pp1-78.

[B&M] John Bell and Moshe Machover: **A Course in Mathematical Logic**. North-Holland, 1977.

Other logic texts: The first is more elementary and readable.

[Enderton] Herbert Enderton: **A Mathematical Introduction to Logic**. Academic Press, 1972.

[Mendelson] E. Mendelson: **Introduction to Mathematical Logic**. Wadsworth & Brooks/Cole, 1987.

Computability text:

[Sipser] Michael Sipser: **Introduction to the Theory of Computation**. PWS, 1997.

[DSW] M. Davis, R. Sigal, and E. Weyuker: **Computability, Complexity and Languages: Fundamentals of Theoretical Computer Science**. Academic Press, 1994.

Propositional Calculus

Throughout our treatment of formal logic it is important to distinguish between *syntax* and *semantics*. Syntax is concerned with the structure of strings of symbols (e.g. formulas and formal proofs), and rules for manipulating them, without regard to their meaning. Semantics is concerned with their meaning.

Syntax

Formulas are certain strings of symbols as specified below. In this chapter we use formula to mean *propositional formula*. Later the meaning of formula will be extended to *first-order formula.*

(Propositional) formulas are built from *atoms* $P_1, P_2, P_3, \ldots$, the *unary connective* $\neg$, the binary connectives $\wedge, \vee$, and parentheses (,). (The symbols $\neg, \wedge$ and $\vee$ are read "not", "and" and "or", respectively.) We use $P, Q, R, \ldots$ to stand for atoms. Formulas are defined recursively as follows:

Definition of Propositional Formula

1) Any atom P is a formula.
2) If A is a formula so is $\neg A$.
3) If A, B are formulas, so is $(A \wedge B)$.
4) If A, B are formulas, so is $(A \vee B)$.

All (propositional) formulas are constructed from atoms using rules 2) - 4).

Examples of formulas: P , $(P \vee Q)$, $(\neg(P \wedge Q) \wedge (\neg P \vee \neg Q))$.

A *subformula* of a formula A is any substring of A which is a formula. For example, P, Q, $(P \wedge Q)$ and $\neg(P \wedge Q)$ are all subformulas of $\neg(P \wedge Q)$, but $P\wedge$ is not a subformula.

We will use $\supset$ ("implies") and $\leftrightarrow$ ("is equivalent to") as abbreviations as follows:

$(A \supset B)$ stands for $(\neg A \vee B)$

$(A \leftrightarrow B)$ stands for $((A \supset B) \wedge (B \supset A))$

Unique Readability Theorem: (The grammar for generating formulas is unambiguous) Suppose A, B, A', B' are formulas, c and c' are binary connectives, and $(AcB) =_{syn} (A'c'B')$. Then $A =_{syn} A'$, $B =_{syn} B'$ and $c =_{syn} c'$.

Here we write $A =_{syn} A'$ instead of $A = A'$ to emphasize that A and A' are equal as strings of symbols (syntactic identity, rather than semantic identity). Note that $=_{syn}$ is a symbol of the "metalanguage" rather than the formal "object language".

Proof Assign weights

0	to $\neg$
1	to each binary connective $\wedge, \vee$
1	to (
-1	to)
-1	to each atom P.

Definiton The *Weight* of A is the sum of the weights of the symbols in A.

Lemma The weight of any formula is -1, but the weight of any proper initial segment is ≥ 0. (Hence no proper initial segment of a formula is a formula.)

Proof Structural induction on length of A. By *structural induction* we mean induction on the length of A, following the definition of propositional formula given above. The base case of the induction is the case in which A is an atom P. The lemma is obvious in this case. The induction step has one case for each of the three ways of constructing new formulas from simpler formulas, using $\neg, \wedge, \vee$. For example, in the case of $\wedge$, the task is to prove the lemma for $(A \wedge B)$, assuming (by the induction hypothesis) that the lemma holds for both A and B. We leave this as an exercise.

The Unique Readability Theorem follows from the Lemma, because if $(AcB) =_{syn} (A'c'B')$ then either A must be an initial segment of A' or vice versa, so in either case $A = A'$ by the Lemma.

In practice we will omit some of the parentheses in a formula when it does not cause ambiguity. For example, we may write $P \vee Q$ when we really mean $(P \vee Q)$. We use the convention associativity to the left for $\wedge$ and $\vee$. For example,

$$(A_1 \vee A_2 \vee A_3 \vee A_4) \text{ stands for } (((A_1 \vee A_2) \vee A_3) \vee A_4)$$

Semantics

Definition A *truth assignment* is a map $\tau : \{\text{atoms}\} \to \{T, F\}$.

(Here $\{T, F\}$ represents { true, false }). A truth assignment τ can be extended to assign either T or F to every formula, as follows:

1) $(\neg A)^\tau = T$ iff $A^\tau = F$
2) $(A \wedge B)^\tau = T$ iff $A^\tau = T$ and $B^\tau = T$
3) $(A \vee B)^\tau = T$ iff $A^\tau = T$ or $B^\tau = T$

Definition τ *satisfies* A iff $A^\tau = T$; τ *satisfies* a set Φ of formulas iff τ satisfies A for all $A \in \Phi$. Φ is *satisfiable* iff some τ satisfies Φ; otherwise Φ is *unsatisfiable*. Similarly for A.

IMPORTANT DEFINITION $\Phi \models A$ (i.e. A is a *logical* consequence of Φ) iff τ satisfies A for every τ such that τ satisfies Φ.

Notation We sometimes use the notation $\models A$ for $\varnothing \models A$, and $B \models$A for $\{B\} \models A$, and $B, C \models A$ for $\{B, C\} \models A$, etc.

Transitivity of Logical Consequence: If $\Phi \models A$ and $\Phi \cup \{A\} \models B$, then $\Phi \models B$.

Proof: EXERCISE

Definition A formula A is *valid* iff $\models A$ (i.e. $A^\tau = T$ for all τ). A valid propositional formula is called a *tautology*. We say that A and B are *equivalent* (written $A \iff B$) iff $A \models B$ and $B \models A$.

Note that $\iff$ refers to semantic equivalence, as opposed to $=_{syn}$, which indicates syntactic equivalence. For example, $(P \vee Q) \iff (Q \vee P)$, but $(P \vee Q) \neq_{syn} (Q \vee P)$.

Convention: P, Q, R stand for *distinct* atoms, so for example $(P \vee Q) \neq_{syn} (Q \vee P)$. However $A, B, C, \ldots$ could stand for identical formulas.

Proposition $\Phi \models A$ iff $\Phi \cup \{\neg A\}$ is unsatisfiable. Also A is a tautology iff $\neg A$ is unsatisfiable.

Proof: Immediate from the definitions of "unsatisfiable" and $\models$.

Examples: (Verify these)

The following are tautologies for all formulas A, B, C:
$A \vee \neg A$
$A \supset A$
$\neg(A \wedge \neg A)$
$(\neg A \vee ((A \wedge B) \vee (A \wedge \neg B))$

Logical consequence:
$(A \wedge B) \models (A \vee B)$

Equivalences:
$(A \vee B) \iff (B \vee A)$ ($\vee$ is commutative)
$(A \wedge B) \iff (B \wedge A)$ ($\wedge$ is commutative)
$(A \vee (B \vee C)) \iff ((A \vee B) \vee C)$ ($\vee$ is associative)
$(A \wedge (B \wedge C)) \iff ((A \wedge B) \wedge C)$ ($\wedge$ is associative)
$(A \wedge (B \vee C)) \iff ((A \wedge B) \vee (A \wedge C))$ ($\wedge$ distributes over $\vee$.)
$(A \vee (B \wedge C)) \iff ((A \vee B) \wedge (A \vee C))$ ($\vee$ distributes over $\wedge$.)
$\neg(A \vee B) \iff (\neg A \wedge \neg B)$ (De Morgan's Law)
$\neg(A \wedge B) \iff (\neg A \vee \neg B)$ (De Morgan's Law)
$(A \supset B) \iff (\neg B \supset \neg A)$ (contrapositive)

Exercise 1 *Prove the following* **Duality Theorem** *by structural induction on A: Let A' be the result of interchanging $\vee$ and $\wedge$ in A, and replacing P by $\neg P$ for each atom P. Then $A' \iff \neg A$.*

Exercise 2 *Give a semantic proof of the* **Craig Interpolation Lemma:** *Given propositional formulas A and B, let S be the set of atoms which occur in both A and B, and* *assume that S is nonempty. If $A \supset B$ is valid, then there is a formula C (an "interpolant") containing only atoms from S such that both $A \supset C$ and $C \supset B$ are valid.*

Remark: The Lemma still holds even when S is empty, provided we include the symbols 0 and 1 (meaning False and True) as building blocks in our definition of formula. It is illuminating to consider this special case when trying to find the proof.

DNF and CNF

A formula of the form $(A_1 \vee A_2 \vee ... \vee A_n)$ is said to be a *disjunction* of the formulas $A_1, A_2, ..., A_n$. If $n = 1$, then the disjunction is just the formula A_1. If $n \geq 3$, then according to clause 4) in the definition of *propositional formula* (see page 2), extra parentheses must be inserted in order to make this a syntactically correct formula. Since $\vee$ is associative, the meaning of the formula does not depend on how these parentheses are inserted. For definiteness, we will use the convention *association to the left*. Thus, for example

$$(A_1 \vee A_2 \vee A_3 \vee A_4) \text{ means } (((A_1 \vee A_2) \vee A_3) \vee A_4)$$

Similarly, $(A_1 \wedge A_2 \wedge ... \wedge A_n)$ is said to be a *conjunction* of the formulas $A_1, A_2, ..., A_n$, and again we use the convention association to the left to specify the location of the extra parentheses.

Definitions: A *literal* ℓ is an atom P, or a negated atom $\neg P$. (We sometimes write $\bar{P}$ for $\neg P$.)

A *clause* C is a disjunction of literals such that no variable occurs twice (negated or not) in the disjunction.

A formula is in *conjunctive normal form* (CNF) if it is a conjunction of one or more clauses.

Note: We will consider the empty conjunction $\wedge\varnothing$ to be a CNF formula, even though it is not a formula according to our definition on page 2. By way of semantics, $\wedge\varnothing$ is valid.

Examples: The following formulas are in CNF:

$\wedge\varnothing$
Q
$\neg Q$
$(P \vee \neg Q \vee R)$
$\neg R \wedge (R \vee S) \wedge (\neg R \vee \neg S)$

The dual notion to CNF is DNF (disjunctive normal form). We say an $\wedge$-clause is a conjunction of literals with no repeated variable, and a formula is in DNF if it is a disjunction of $\wedge$-clauses.

We allow the empty disjunction $\vee\varnothing$ to be a DNF formula, with the semantics that $\vee\varnothing$ is unsatisfiable.

Examples of formulas in DNF can be obtained by interchanging $\wedge$ and $\vee$ in the above examples of CNF formulas.

Theorem: Every formula is equivalent to a formula in CNF, and to a formula in DNF.

Proof: One way to form a DNF equivalent to A is to put in an $\wedge$-clause corresponding to each truth assignment satisfying A. For example, if the truth assignment $P^\tau = F$, $Q^\tau = T$, $R^\tau = F$ satisfies A, then include the $\wedge$-clause $\neg P \wedge Q \wedge \neg R$ in the disjunction forming the DNF formula. If A is unsatisfiable, then its DNF is the empty disjunction $\vee\varnothing$.

CNF equivalent formulas are constructed in a dual fashion. $\Box$

Of course CNF and DNF equivalent formulas are far from unique. For example, $(P \vee Q) \wedge (P \vee \neg Q)$ is a CNF formula, and it is equivalent to the simpler CNF formula P. A traditional (but computationally intractable) problem is to find a smallest DNF (or CNF) formula equivalent to a given formula.

Exercise 3 *Prove that every CNF formula equivalent to*

$$(P_1 \wedge Q_1) \vee (P_2 \wedge Q_2) \vee ... \vee (P_n \wedge Q_n)$$

must have at least 2^n clauses. (Hint: Show that for every assignment of either P or Q to each of the subscripts $\{1, 2, ..., n\}$ there is a clause in the CNF which has exactly one of P_i, Q_i for each i, according to whether P or Q is assigned to i. For example, if $n = 3$, then there must be a clause whose positive literals are exactly $\{Q_1, P_2, Q_3\}$. (A literal is positive *if is has no $\neg$.)*

Formal Proofs

One way to establish that a formula A with n atoms is a tautology is to verify that $A^\tau = T$ for all 2^n truth assignments τ to to the atoms of A. A similar exhaustive method can be used to verify that A is a logical consequence of a finite set Φ of formulas. However another way is to use the notion of a formal proof, which may be both more efficient and more illuminating. A formal proof is a syntactic notion, in contrast to validity, which is a semantic notion. Many formal proof systems have been studied. Here we present two examples: resolution and Gentzen's system PK. We give a brief introduction to the former, but we concentrate on the latter, since it will serve as the basis for our proof system for the predicate calculus.

Resolution

Resolution is important because it serves as the basis of most automated theorem provers, and is has been thoroughly studied and analyzed. Resolution in the propositional calculus is a proof system for establishing the unsatisfiability of CNF formulas. However it can be generalized to apply to arbitrary propositional formulas A, establishing validity if A is valid, or unsatisfiability if A is unsatisfiable, or that $\Phi \models A$ if that is the case. According to the Proposition on page 4, all these things can be reduced to establishing the unsatisfiability of a set of formulas. The next result shows that it is sufficient to establish the unsatisfiability of a set of clauses. The condition that the set Φ of formulas is finite is made less important by the Propositional Compactness Theorem (see page 14).

SAT Theorem: There is a polynomial time procedure which transforms a given finite set Φ of propositional formulas to a finite set $S = S_\Phi$ of clauses, such that Φ is satisfiable iff S is satisfiable.

Proof sketch: Our first try might be to place every formula in Φ in CNF, and let S be the set of all clauses that occur as a conjunct in one of these CNF formulas. Indeed this S is satisfiable iff Φ is satisfiable (because the conjunction of the clauses in S is equivalent to the conjunction of the formulas in Φ), but by Exercise 3 this is not a polynomial time procedure.

The correct proof is based on a standard method for showing that the problem General Propositional Satisfiability is polynomial time reducible to SAT (satisfiability of CNF formulas). (See any text on NP-completeness.) The idea is to introduce a new atom P_B for every subformula B of every formula in Φ, except let P_B be B if B is a literal. Now place in S clauses which assert that each new atom P_B has the appropriate truth value with respect to the atoms or literals corresponding to the principle subformulas of B. Finally place in S the clause P_A, for every formula A in Φ.

For example, if Φ consists of the single formula

$$A = (Q \wedge R) \vee \neg Q$$

then we define $B = (Q \wedge R)$, and introduce the new atoms P_A and P_B. Let

$$S = \{\bar{P_B} \vee Q, \bar{P_B} \vee R, P_B \vee \bar{Q} \vee \bar{R}, \bar{P_A} \vee P_B \vee \bar{Q}, P_A \vee \bar{P_B}, P_A \vee Q, P_A\}$$

The first three clauses in S assert $P_B \iff (Q \wedge R)$, and the second three clauses assert $P_A \iff (P_B \vee \neg Q)$.

Note that A is not equivalent to the conjunction of the clauses in S, but A is satisfiable iff S is satisfiable (in fact both are satisfiable). $\Box$

Exercise 4 *Give a truth assignment demonstrating the lack of equivalence asserted in the last sentence above. (See the definition of $\iff$, bottom of page 3.)*

Notation: If ℓ is a literal, then $\bar{\ell}$ is defined to be $\bar{P}$ if $\ell = P$, and P if $\ell = \bar{P}$. We say that $\bar{\ell}$ is the *complement* of ℓ.

Order Convention: We think of a clause as a set of literals, meaning their disjunction. Thus if two clauses have the same literals, but written in different orders or with different repetitions, we treat them as the same clause.

Resolution Rule: Let C_1, C_2 be clauses of the form $C_1 = (A \vee \ell)$, and $C_2 = (B \vee \bar{\ell})$, where A and B are clauses not containing ℓ or $\bar{\ell}$. Then the *resolvant* of C_1 and C_2 is the clause

$C_3 = (A \vee B)$. We assume that A and B have no literal clashes, so that $(A \vee B)$ cannot contain both a literal and its complement.

Examples: The resolvant of P and $\bar{P}$ is the empty clause $\vee\varnothing$. The resolvant of $(P \vee Q)$ and $(\bar{Q} \vee P)$ is P. The clauses $(P \vee Q)$ and $(\bar{P} \vee \bar{Q})$ have no resolvant, because they have two clashes. The resolvant of $(P \vee \bar{Q} \vee R)$ and $(Q \vee S)$ is $(P \vee R \vee S)$.

Resolvant Soundness Principle: If C_3 is the resolvant of C_1 and C_2, then

$$C_1, C_2 \models C_3$$

That is, the resolvant of two clauses is a logical consequence of the clauses. (See the IMPORTANT DEFINITION, page 3.) This applies in particular if C_3 is the empty clause $\vee\varnothing$, which is unsatisfiable.

RES Definition: A *resolution refutation* of a set S of clauses is a sequence $C_1, C_2, ..., C_q$ of clauses such that the final clause C_q is the empty clause $\vee\varnothing$, and each C_i is either in S or is the resolvant of earlier clauses in the sequence.

Example: Let S $= \{(P \vee Q), (\bar{Q} \vee R), (\bar{P} \vee S), (\bar{P} \vee \bar{S}), \bar{R}\}$. Then a resolution refutation of S is the sequence

$$(P \vee Q), (\bar{Q} \vee R), (P \vee R), (\bar{P} \vee S), (\bar{P} \vee \bar{S}), \bar{P}, R, \bar{R}, \vee\varnothing$$

It is helpful to write this refutation in tree form, where the parents of a resolvant are the two clauses forming the resolvant.

RES Soundness Theorem: If a set S of clauses has a resolution refutation, then S is unsatisfiable.

Proof: Let $C_1, C_2, ..., C_q$ be a resolution refutation of S. Using the Resolvant Soundness Principle above, and Transitivity of Logical Consequence (page 3), it follows by induction on i that every clause C_i is a logical consequence of S. In particular, the empty clause C_q is a logical consequence of S. But $\vee\varnothing$ is unsatisfiable. Hence no truth assignment can satisfy S. $\square$

RES Completeness Theorem: Every unsatisfiable set of clauses has a resolution refutation.

Proof: We will prove this for finite sets S of clauses, although by the Propositional Compactness Theorem (page 14) it follows also that every unsatisfiable infinite set S of clauses has a finite resolution refutation.

To prove the theorem, we outline a procedure which can be used in practice to generate a resolution refutation of S if S is unsatisfiable, or to find a satisfying assignment for S if S is satisfiable. The procedure maintains a set $S' \supseteq S$ of clauses which are arranged in a sequence forming a partial resolution refutation of S (i.e. a sequence of clauses each of which is either in S or is a resolvant of earlier clauses in the sequence). The procedure also maintains a stack $\ell_1, \ell_2, ..., \ell_k$ of literals representing a partial truth assignment to the atoms of S. This partial assignment τ makes each literal ℓ_j on the stack true, and it has the property that no clause in S' is falsified by τ. (I.e. every clause in S' has at least one literal not falsified by τ.)

1. If S includes the empty clause $\vee\varnothing$, then the resolution refutation consists simply of $\vee\varnothing$. Otherwise, initialize $S' = S$ and initialize the stack of literals to be empty.

2. The general step is as follows. If the partial assignment τ satisfies every clause in S, then output τ and halt. Otherwise select a clause C in S which is not satisfied by τ, and a literal ℓ in C which is not falsified by τ. Push ℓ onto the stack, and let τ' be the resulting extension of τ (so τ' makes ℓ true). If τ' does not falsify any clause in S', then go to step 2 with $\tau \leftarrow \tau'$.

3. Otherwise suppose that τ' falsifies a clause C' in S'. Then replace ℓ on the stack by $\bar{\ell}$, and let τ'' be the resulting partial truth assignment (so τ'' falsifies ℓ). If τ'' does not falsify any clause in S', then go to step 2 with $\tau \leftarrow \tau''$.

4. Otherwise suppose that τ'' falsifies the clause C'' in S'. In this case the clauses C' and C'' can be resolved, forming a resolvant R which eliminates the literals ℓ and $\bar{\ell}$, and such that R is falsified by the original truth assignment τ from step 2 (and hence does not occur in S'). If R is the empty clause, then output the resolution refutation $S' \cup \{R\}$ and halt. Otherwise pop the stack until the first point at which R is not falsified. Go to step 2 with $S' \leftarrow S' \cup \{R\}$.

To complete the proof, we need only show that the procedure always halts, since all halting steps end either with a satisfying assignment (step 2) or a resolution refutation (step 4). To see that the procedure halts, simply note that each execution of the general step results either in adding a new distinct literal to the stack, or adding a new clause to the list S'. There are only finitely many literals, so eventually a new clause must be added to S', and there are only finitely many distinct clauses that can be formed from the literals, so if no satisfying assignment is found, eventually the empty clause must be added to S'. $\square$

An important theorem in proof complexity states that there are arbitrarily large unsatisfiable clause sets S whose minimum resolution refutation contains a number of clauses exponential in the number of clauses in S. From this it can be shown that most programs used in practice for satisfiability testing require exponential time, in the worst case.

Gentzen's Proof System PK

We now present the system PK based on the very elegant sequent calculus, introduced by Gerhard Gentzen in 1935 (see [Buss], section 1.2.1).

In the propositional sequent calculus system PK, each line in a proof is a *sequent* of the form

$$S = A_1, ..., A_k \rightarrow B_1, ..., B_\ell \tag{1}$$

where $\rightarrow$ is a new symbol (not to be confused with $\supset$), and $A_1, ..., A_k$ and $B_1, ..., B_\ell$ are sequences of formulas ($k, \ell \geq 0$) called *cedents*. We call the cedent $A_1, ..., A_k$ the *antecedent* and $B_1, ..., B_\ell$ the *succedent*.

Semantics of Sequents

The semantics of sequents is given as follows. We say that a truth assignment τ *satisfies* the sequent S in (1) iff either τ falsifies some A_i or τ satisfies some B_i. Thus the sequent is equivalent to the formula

$$A_S = (A_1 \wedge A_2 \wedge ... \wedge A_k) \supset (B_1 \vee B_2 \vee ... \vee B_\ell) \tag{2}$$

(In other words, the conjunction of the A's implies the disjunction of the B's.) In the cases in which the antecedent or succedent is empty, we see that the sequent $\rightarrow A$ is equivalent to the formula A, and $A \rightarrow$ is equivalent to $\neg A$, and just $\rightarrow$ (with both antecedent and succedent empty) is false (unsatisfiable). We say that a sequent is *valid* if it is true under all truth assignments (which is the same as saying that its corresponding formula A_S is a tautology). Similarly we can defining the notion of *logical consequence* for sequents, by referring to the corresponding formulas.

Examples: The following are valid sequents, for any formulas A, B:

$A \rightarrow A$
$\rightarrow A, \neg A$
$A, \neg A \rightarrow$
$\rightarrow A \vee \neg A$
$A, (A \supset B) \rightarrow B$

A formal proof (or just *proof*) in the propositional sequent calculus PK is a finite rooted tree in which the nodes are (labeled with) sequents. The sequent at the root (written at the bottom) is what is being proved, and is called the *endsequent*. The sequents at the leaves, written at the top, are *logical axioms*, and must be of the form $A \rightarrow A$, where A is a formula. Each sequent other than the logical axioms must follow from its parent sequent(s) by one of the following rules of inference. For each rule, the sequent on the bottom follows from the sequent on the top. Here Γ and Δ denote finite sequences (possibly empty) of formulas, and A and B denote formulas.

weakening rules

$$\textbf{left}\ \frac{\Gamma \rightarrow \Delta}{A, \Gamma \rightarrow \Delta} \qquad \qquad \textbf{right}\ \frac{\Gamma \rightarrow \Delta}{\Gamma \rightarrow \Delta, A}$$

exchange rules

$$\textbf{left}\ \frac{\Gamma_1, A, B, \Gamma_2 \rightarrow \Delta}{\Gamma_1, B, A, \Gamma_2 \rightarrow \Delta} \qquad \qquad \textbf{right}\ \frac{\Gamma \rightarrow \Delta_1, A, B, \Delta_2}{\Gamma \rightarrow \Delta_1, B, A, \Delta_2}$$

contraction rules

$$\textbf{left}\ \frac{\Gamma, A, A \rightarrow \Delta}{\Gamma, A \rightarrow \Delta} \qquad \qquad \textbf{right}\ \frac{\Gamma \rightarrow \Delta, A, A}{\Gamma \rightarrow \Delta, A}$$

$\neg$ introduction rules

$$\textbf{left}\ \frac{\Gamma \to \Delta, A}{\neg A, \Gamma \to \Delta} \qquad\qquad \textbf{right}\ \frac{A, \Gamma \to \Delta}{\Gamma \to \Delta, \neg A}$$

$\wedge$ introduction rules

$$\textbf{left}\ \frac{A, B, \Gamma \to \Delta}{(A \wedge B), \Gamma \to \Delta} \qquad\qquad \textbf{right}\ \frac{\Gamma \to \Delta, A \quad \Gamma \to \Delta, B}{\Gamma \to \Delta, (A \wedge B)}$$

$\vee$ introduction rules

$$\textbf{left}\ \frac{A, \Gamma \to \Delta \quad B, \Gamma \to \Delta}{(A \vee B), \Gamma \to \Delta} \qquad\qquad \textbf{right}\ \frac{\Gamma \to \Delta, A, B}{\Gamma \to \Delta, (A \vee B)}$$

cut rule

$$\frac{\Gamma \to \Delta, A \quad A, \Gamma \to \Delta}{\Gamma \to \Delta}$$

The formula A in the cut rule is called the *cut formula.*

Note that there is one **left** introduction rule and one **right** introduction rule for each of the three logical connectives $\wedge, \vee, \neg$. Further, these rules seem to be the simplest possible, given that the fact that for each introduction rule the bottom sequent is valid iff all top sequents are valid.

Definition: A PK proof of a formula A is a PK proof of $\to A$.

Exercise 5 *Write down each of the six introduction rules from memory.*

Sequent Soundness Principle: For each PK rule, the sequent on the bottom is a logical consequence of the sequent(s) on the top.

Proof: EXERCISE

Note that repeated use of the exchange rules allows us to execute an arbitrary reordering of the formulas in the antecedent or succedent of a sequent. In presenting a proof in the system PK, we will usually omit mention of the steps requiring the exchange rules, but of course they are there.

As an example, we give a PK proof of one of DeMorgan's laws:

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

To find this (or any) proof, it is a good idea to start with the conclusion at the bottom, and work up by removing the connectives one at a time, outermost first, by using the introduction rules in reverse. This can be continued until some atom P occurs on both the left and right side of a sequent. Then this sequent can be derived from the axiom $P \to P$ using weakenings and exchanges. The cut and contraction rules are not necessary, and weakenings are only needed immediately below axioms. (The cut rule can be used to shorten proofs, and contraction will be needed later for the predicate calculus.)

$$
\dfrac{\dfrac{\dfrac{P \to P}{P \to P, \neg Q}\,(weakening)}{\to P, \neg P, \neg Q}\,(\neg\ right) \qquad \dfrac{\dfrac{Q \to Q}{Q \to Q, \neg P}\,(weakening)}{\to Q, \neg P, \neg Q}\,(\neg\ right)}{\dfrac{\dfrac{\to P \wedge Q, \neg P, \neg Q}{\to P \wedge Q, \neg P \vee \neg Q}\,(\vee\ right)}{\neg(P \wedge Q) \to \neg P \vee \neg Q}\,(\neg\ left)}\,(\wedge\ right)
$$

Exercise 6 *Give PK proofs for each of the following valid sequents:*

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

Exercise 7 *Show that the contraction rules can be derived from the cut rule (with weakenings and exchanges).*

Exercise 8 *Suppose that we allowed $\supset$ as a primitive connective, rather than one introduced by definition. Give the appropriate left and right introduction rules for $\supset$.*

Now we prove that PK is both sound and complete. That is, a propositional sequent is provable in PK iff it is valid.

PK Soundness Theorem: Every sequent provable in PK is valid.

Proof: We show that the endsequent in every PK proof is valid, by induction on the number of sequents in the proof. For the base case, the proof is a single line; an axiom $A \to A$. This is obviously valid. For the induction step, one need only verify for each rule, if all top sequents are valid, then the bottom sequent is valid. This follows from the Sequent Soundness Principle above. □

Cut-free proofs A PK proof is *cut-free* if it does not use the cut rule. The following is a useful property of cut-free proofs.

Subformula Property: Every formula in every sequent in a cut-free PK proof is a subformula of a formula in the endsequent.

This principle is proved by a simple induction on the length of cut-free PK proofs, by observing that for every PK rule except cut, every formula on the top is a subformula of some formula on the bottom. In other words, once a formula occurs in a PK proof, there is no way to get rid of it except by using the cut rule.

In fact, the cut rule is not necessary for proving that a formula is valid. However the cut rule can shorten the proof of validity. Also, as we shall see, the cut rule is sometimes necessary for showing that a formula is a logical consequence of other formulas.

It turns out that the contraction rule is not necessary either (although it is necessary in the system LK for the predicate calculus).

PK Completeness Theorem: Every valid propositional sequent has a cut-free PK proof which does not use the contraction rule.

Proof: The idea is discussed in the example proof above of DeMorgan's laws. We need to use the inversion principle.

Inversion Principle: For each PK rule except weakening, if the bottom sequent is valid, then all top sequents are valid.

This principle is easily verified by inspecting each of the ten rules in question.

Now for the completeness theorem: We show that every valid sequent $\Gamma \to \Delta$ has a PK proof, by induction on the total number of logical connectives $\wedge, \vee, \neg$ occurring in $\Gamma \to \Delta$. For the base case, every formula in Γ and Δ is an atom, and since the sequent is valid, some atom P must occur in both Γ and Δ. Hence $\Gamma \to \Delta$ can be derived from the axiom $P \to P$ by weakenings and exchanges.

For the induction step, let A be any nonatomic formula (i.e. A is not an atom) in Γ or Δ. Then by the definition of propositional formula A must have one of the forms $(B \wedge C)$, $(B \vee C)$, or $\neg B$. Thus $\Gamma \to \Delta$ can be derived from $\wedge$ introduction, $\vee$ introduction, or $\neg$ introduction, respectively, using either the **left** case or the **right** case, depending on whether A is in Γ or Δ, and also using exhcanges, but no weakenings. In each case, each top sequent of the rule will have at least one fewer connective than $\Gamma \to \Delta$, and the sequent is valid by the inversion principle. Hence each top sequent has a PK proof, by the induction hypothesis. $\square$

Remark: The soundness and completeness theorems relate the *semantic* notion of validity to the *syntactic* notion of proof.

We generalize the (semantic) definition of logical consequence from formulas to sequents in the obvious way: A sequent S is a *logical consequence* of a set Φ of sequents iff every truth assignment τ that satisfies Φ also satisfies S.

We generalize the (syntactic) definition of PK proof of a sequent S to a PK proof of S *from a set* Φ *sequents* (also called a $PK - \Phi$ proof) by allowing sequents in Φ to be leaves (or nonlogical axioms) in the proof tree, in addition to the logical axioms $A \to A$. The $PK - \Phi$ proof must always be finite, even when Φ is infinite. (Of course not all members of Φ need occur in the proof.)

It turns out that soundness and completeness generalize to this setting.

Derivational Soundness and Completeness Theorem: A sequent S is a logical consequence of a set Φ of sequents iff S has a (finite) $PK - \Phi$ proof.

A remarkable aspect of completeness is that a finite proof exists even in case Φ is an infinite set. This is because of the compactness theorem (below) which implies that if S is a logical consequence of Φ, then S is a logical consequence of some finite subset of Φ.

In general, to prove S from Φ, the cut rule is required. In particular, there is no cut-free PK proof of $\to P$ from $\to P \wedge Q$. This follows from the subformula property for cut-free $PK - \Phi$ proofs (see page 12 for the case when $\Phi = \varnothing$): Every formula in every sequent in a cut-free $PK - \Phi$ proof is a subformula of a formula in the endsequent.

Proof of Derivational Soundness and Completeness:

Derivational soundness is proved in the same way as simple soundness: by induction on the number of sequents in the PK proof. In the previous proof we observed that if the top sequents of a rule are valid, then the bottom sequent is valid. Now we observe that the bottom sequent is a logical consequence of the top sequent(s).

To prove completeness, by the Compactness Theorem below it suffices to consider the case in which $\Phi = \{S_1, \ldots, S_k\}$ is a finite set of sequents. We use the PK Completeness Theorem (page 13) and the formula A_S giving the semantics of a sequent S (see (2) on page 10). Recall that the formula A_S is logically equivalent to the sequent S (i.e. A_S and S get the same truth values for every truth assignment τ). From this, assuming that the sequent $\Gamma \to \Delta$ is a logical consequence of the set $\{S_1, \ldots, S_k\}$ of sequents, it follows that the sequent

$$\Gamma, A_{S_1}, \ldots, A_{S_k} \to \Delta \tag{3}$$

is valid. Hence by the PK Completeness Theorem, (3) has a PK proof. From the Exercise below, it follows that for each i, $1 \le i \le k$, the sequent $\to A_{S_i}$ has a PK derivation from the sequent S_i. Finally, the sequent $\Gamma \to \Delta$ can be derived from (3) and $\to A_1, \ldots, \to A_k$ using k cuts (together with weakenings and exchanges). □

Exercise 9 *For every sequent S, there is a PK derivation of A_S from S.*

Anchored Proofs (This notion is not needed for the rest of the course.) Note that in the above proof of derivational completeness, the only cut formulas needed are the sequent semantic formulas A_{S_i}, where the sequent S_i is in the hypothesis set Φ. For some applications it is important to know that in fact the only cut formulas needed are those that occur as formulas in the hypotheses S_i. (Here the formulas that occur in the sequent

$$A_1, \ldots, A_k \to B_1 \ldots B_\ell$$

are the formulas $A_1, \ldots, A_k, B_1, \ldots, B_\ell$.)

We say that a $PK - \Phi$ proof π is *anchored* if every cut formula in π is a formula that occurs in one of the sequents in Φ.

Anchored Completeness Theorem: If a sequent S is a logical consequence of a set Φ of sequents, then S has an anchored $PK - \Phi$ proof.

We illustrate the anchored completeness theorem by proving the special case in which Φ consists of the single sequent $A \to B$. Assume that the sequent $\Gamma \to \Delta$ is a logical consequence of $A \to B$. Then both of the sequents $\Gamma \to \Delta, A$ and $B, A, \Gamma \to \Delta$ are valid (why?). Hence by the earlier completeness theorem, they have PK proofs π_1 and π_2. We can use these proofs to get a proof of $\Gamma \to \Delta$ from $A \to B$ as shown below, where the double line indicates several rules have been applied.

$$\dfrac{\begin{array}{c}\vdots\ \pi_1\\ \Gamma \to \Delta, A\end{array} \qquad \dfrac{\dfrac{A \to B}{A, \Gamma \to \Delta, B}\ (weakenings, exchanges) \qquad \begin{array}{c}\vdots\ \pi_2\\ B, A, \Gamma \to \Delta\end{array}}{A, \Gamma \to \Delta}\ (cut)}{\Gamma \to \Delta}\ (cut)$$

Next consider the case in which Φ has the form $\{\to A_1, \to A_2, ..., \to A_k\}$ for some set $\{A_1, ..., A_k\}$ of formulas. Assume that $\Gamma \to \Delta$ is a logical consequence of Φ in this case. Then the sequent

$$A_1, A_2, ..., A_k, \Gamma \to \Delta$$

is valid (why?), and hence has a PK proof π. Now we can use the assumptions Φ and the cut rule to successively remove $A_1, A_2, ..., A_k$ from the above sequent to conclude $\Gamma \to \Delta$. For example, A_1 is removed as follows:

$$\dfrac{\dfrac{\to A_1}{A_2, ..., A_k, \Gamma \to \Delta, A_1}\ (weakenings, exchanges) \qquad \begin{array}{c}\vdots\ \pi\\ A_1, A_2, ..., A_k, \Gamma \to \Delta\end{array}}{A_2, ..., A_k, \Gamma \to \Delta}\ (cut)$$

Exercise 10 *Prove the anchored completeness theorem for the more general case in which Φ is any finite set of sequents.*

Propositional Compactness Theorem: We state three different forms of this result. All three are equivalent.

Form 1: If Φ is an unsatisfiable set of propositional formulas, then some finite subset of Φ is unsatisfiable.

Form 2: If a formula A is a logical consequence of a set Φ of formulas, then A is a logical consequence of some finite subset of Φ.

Form 3: If every finite subset of a set Φ of formulas is satisfiable, then Φ is satisfiable.

Exercise 11 *Prove the equivalence of the three forms. (Note that Form 3 is the contrapositive of Form 1.)*

Proof of Form 1: Let Φ be an unsatisfiable set of formulas. We assume that the set of atoms occurring in formulas in Φ is finite or countable. In other words, there is an infinite list $P_1, P_2, P_3, \ldots$ of distinct atoms which includes all atoms occurring in Φ. The exercise below concerns the general case. Organize the set of truth valuations into an infinite rooted binary tree B. Each node except the root is labelled with a literal P_i or $\neg P_i$. The two children of the root are labelled P_1 and $\neg P_1$, indicating that P_1 is assigned T or F, respectively. The two children of each of these nodes are labelled P_2 and $\neg P_2$, respectively, indicating the truth value of P_2. Thus each infinite branch in the tree represents a complete truth assignment, and each path from the root to a node represents a truth assignment to the atoms $P_1, \ldots, P_i$, for some i.

Now for every node ν in the tree B, prune the tree at ν (i.e. remove the subtree rooted at ν, keeping ν itself) if the partial truth assignment τ_ν represented by the path to ν falsifies some formula A_ν in Φ, where all atoms in A_ν get values from τ_ν. Let B' be the resulting pruned tree. Since Φ is unsatisfiable, every path from the root in B' must end after finitely many steps in some leaf ν labelled with a formula A_ν in Φ. It follows from König's Lemma below that B' is finite. Let Φ' be the finite subset of Φ consisting of all formulas A_ν labelling the leaves of B'. Since every truth assignment τ determines a path in B' which ends in a leaf A_ν falsified by τ, it follows that Φ' is unsatisfiable. $\square$

König's Lemma: Suppose T is a rooted tree in which every node has only finitely many children. If every branch in T is finite, then T is finite.

Proof: We prove the contrapositive: If T is infinite (but every node has only finitely many children) then T has an infinite branch. We can define an infinite path in T as follows: Start at the root. Since T is infinite but the root has only finitely many children, the subtree rooted at one of these children must be infinite. Choose such a child as the second node in the branch, and continue. $\square$

Exercise 12 (For those with some knowledge of set theory or point set topology) *The above proof of the propositional compactness theorem only works when the set of atoms is countable, but the result still holds even when Φ is an uncountable set with an uncountable set $\mathcal{A}$ of atoms. Complete each of the two proof outlines below.*

(a) Prove Form 3 using Zorn's Lemma as follows: Call a set Ψ of formulas *finitely satisfiable* if every finite subset of Ψ is satisfiable. Assume that Φ is finitely satisfiable. Let $\mathcal{C}$ be the class of all finitely satisfiable sets $\Psi \supseteq \Phi$ of propositional formulas using atoms in Φ. Order these sets Ψ by inclusion. Show that the union of any chain of sets in $\mathcal{C}$ is again in the class $\mathcal{C}$. Hence by Zorn's Lemma, $\mathcal{C}$ has a maximal element Ψ_0. Show that Ψ_0 has a unique satisfying assignment, and hence Φ is satisfiable.

(b) Show that Form 1 follows from Tychonoff's Theorem: The product of compact topological spaces is compact. The set of all truth assignments to the atom set $\mathcal{A}$ can be given the product topology, when viewed as the product for all atoms P in $\mathcal{A}$ of the two-point space $\{T, F\}$ of assignments to P, with the discrete topology. By Tychonoff's Theorem, this space of assignments is compact. Show that for each formula A, the set of assignments falsifying A is open. Thus Form 1 follows from the definition of compact: every open cover has a finite subcover.