Logic for Computer Scientists

Pascal Hitzler
http://www.pascal-hitzler.de

CS 499/699 Lecture, Winter Quarter 2011
Wright State University, Dayton, OH, U.S.A.

[final version: 03/10/2011]

Contents

1 Propositional Logic **2**
1.1 Syntax . . . 2
1.2 Semantics . . . 3
1.3 Equivalence . . . 6
1.4 Normal Forms . . . 7
1.5 Tableaux Algorithm . . . 9

2 First-order Predicate Logic **13**
2.1 Syntax . . . 13
2.2 Semantics . . . 14
2.3 Equivalence . . . 17
2.4 Normal Forms . . . 18
2.5 Tableaux Algorithm . . . 18

Appendix **21**
1.6 Theoretical Aspects (Propositional Logic) . . . 21
2.6 Theoretical Aspects (Predicate Logic) . . . 24

3 Application: Knowledge Representation for the World Wide Web **25**

References

[Ben-Ari, 1993] Ben-Ari, M. (1993). *Mathematical Logic for Computer Science*. Springer.

[Hitzler et al., 2009] Hitzler, P., Krötzsch, M., and Rudolph, S. (2009). *Foundations of Semantic Web Technologies*. Chapman & Hall/CRC.

[Schöning, 1989] Schöning, U. (1989). *Logic for Computer Scientists*. Birkhäuser.

1 Propositional Logic

1.1 Syntax

[Schöning, 1989, Chapter 1.1]

Let $\{A_1, A_2, \dots\}$ be an infinite set of *propositional variables*.

1.1 Definition
An *atomic formula* is a propositional variable.
Formulas are defined by the following inductive process.

1. All atomic formulas are formulas.
2. For every formula F, $\neg F$ is a formula, called the *negation* of F.
3. For all formulas F and G, also $(F \vee G)$ and $(F \wedge G)$ are formulas, called the *disjunction* and the *conjunction* of F and G, respectively.

If a formula F occurs in another formula G, then it is called a *subformula* of G. Note that every formula is a subformula of itself.

1.2 Notation
We use the following abbreviations:
$A, B, C, \dots$ instead of $A_1, A_2, \dots$ and other obvious variants.
[Be careful with the use of F and G!]
We sometimes omit brackets if it can be done safely. [Be careful with this!]
$(F \to G)$ instead of $(\neg F \vee G)$
$(F \leftrightarrow G)$ instead of $(F \to G) \wedge (G \to F)$
$(\bigvee_{i=1}^{n} F_i)$ instead of $(F_1 \vee F_2 \vee \dots \vee F_n)$
$(\bigwedge_{i=1}^{n} F_i)$ instead of $(F_1 \wedge F_2 \wedge \dots \wedge F_n)$

1.3 Example
$(\neg B \to F)$ is $(\neg\neg B \vee F)$.
Some Subformulas: $\neg\neg B$, $\neg B$.

1.4 Example
$((I \vee \neg B) \to \neg F)$ is $(\neg(I \vee \neg B) \vee \neg F)$.
Some Subformulas: $\neg(I \vee \neg B)$, I, $\neg B$.

Exercise 1 (no hand-in) Determine all subformulas of $((B \wedge F) \to \neg I)$.

1.5 Remark
Formulas can be represented in a unique way as *trees*. [Example 1.4 on whiteboard.]

Exercise 2 (no hand-in) Draw the formulas from Example 1.3 and Exercise 1 as trees.

1.2 Semantics

[Schöning, 1989, Chapter 1.1 cont.]

1.6 Definition
$\mathbb{T} = \{0, 1\}$ – the set of *truth values*: *false*, and *true*, respectively.
An *assignment* is a function $\mathcal{A} : \mathbf{D} \to \mathbb{T}$, where $\mathbf{D}$ is a set of atomic formulas.
Given an assignment $\mathcal{A}$, we extend it to $\mathcal{A}' : \mathbf{E} \to \mathbb{T}$, where $\mathbf{E}$ is the set of all formulas containing only elements from $\mathbf{D}$ as atomic subformulas:

1. $\mathcal{A}'(A_i) = \mathcal{A}(A_i)$ for each $A_i \in \mathbf{D}$
2. $\mathcal{A}'(F \wedge G) = \begin{cases} 1, & \text{if } \mathcal{A}'(F) = 1 \text{ and } \mathcal{A}'(G) = 1 \\ 0, & \text{otherwise} \end{cases}$
3. $\mathcal{A}'(F \vee G) = \begin{cases} 1, & \text{if } \mathcal{A}'(F) = 1 \text{ or } \mathcal{A}'(G) = 1 \\ 0, & \text{otherwise} \end{cases}$
4. $\mathcal{A}'(\neg F) = \begin{cases} 1, & \text{if } \mathcal{A}'(F) = 0 \\ 0, & \text{otherwise} \end{cases}$

[From now on, drop distinction between $\mathcal{A}$ and $\mathcal{A}'$.]

1.7 Example
Let $\mathcal{A}(B) = \mathcal{A}(F) = 1$ and $\mathcal{A}(I) = 0$.

$$\begin{aligned} \mathcal{A}(\neg(B \wedge F) \vee \neg I) &= \begin{cases} 1, & \text{if } \mathcal{A}(\neg(B \wedge F)) = 1 \text{ or } \mathcal{A}(\neg I) = 1 \\ 0, & \text{otherwise} \end{cases} \\ &= \begin{cases} 1, & \text{if } \mathcal{A}(B \wedge F) = 0 \text{ or } \mathcal{A}(I) = 0 \\ 0, & \text{otherwise} \end{cases} \\ &= \begin{cases} 1, & \text{if } \mathcal{A}(B) = 0 \text{ or } \mathcal{A}(F) = 0 \text{ or } \mathcal{A}(I) = 0 \\ 0, & \text{otherwise} \end{cases} \\ &= 1 \end{aligned}$$

Exercise 3 (no hand-in) Do the calculation from Example 1.7 for the formula $\neg(I \vee \neg B) \vee \neg F$ from Example 1.4 and the values $\mathcal{A}(I) = 1$ and $\mathcal{A}(B) = \mathcal{A}(F) = 0$.

1.8 Remark
The same thing can be expressed via *truth tables*.

$\mathcal{A}(F)$	$\mathcal{A}(G)$	$\mathcal{A}(F \wedge G)$
0	0	0
0	1	0
1	0	0
1	1	1

$\mathcal{A}(F)$	$\mathcal{A}(G)$	$\mathcal{A}(F \vee G)$
0	0	0
0	1	1
1	0	1
1	1	1

$\mathcal{A}(F)$	$\mathcal{A}(\neg F)$
0	1
1	0

1.9 Example
Determining the truth values of formulas using truth tables:
[Use the tree structure of formulas.]

$\mathcal{A}(B)$	$\mathcal{A}(F)$	$\mathcal{A}(I)$	$\mathcal{A}(B \wedge F)$	$\mathcal{A}(\neg(B \wedge F))$	$\mathcal{A}(\neg I)$	$\mathcal{A}(\neg(B \wedge F) \vee \neg I)$
0	0	0	0	1	1	1
0	0	1	0	1	0	1
0	1	0	0	1	1	1
0	1	1	0	1	0	1
1	0	0	0	1	1	1
1	0	1	0	1	0	1
1	1	0	1	0	1	1
1	1	1	1	0	0	0

1.10 Remark
The truth value of a formula is uniquely determined by the truth values of the propositional variables it contains as subformulas.

Exercise 4 (no hand-in) Make the truth table for the formula from Exercise 3.

1.11 Remark

$\mathcal{A}(F)$	$\mathcal{A}(G)$	$\mathcal{A}(F \to G)$
0	0	1
0	1	1
1	0	0
1	1	1

$\mathcal{A}(F)$	$\mathcal{A}(G)$	$\mathcal{A}(F \leftrightarrow G)$
0	0	1
0	1	0
1	0	0
1	1	1

1.12 Definition
F, a formula, $\mathcal{A}$, an assignment.
$\mathcal{A}$ is *suitable* if it is defined for all atomic formulas occurring in F.
We write $\mathcal{A} \models F$ if $\mathcal{A}$ is suitable for F and $\mathcal{A}(F) = 1$. We say F *holds under* $\mathcal{A}$ or $\mathcal{A}$ *is a model for* F. Otherwise, we write $\mathcal{A} \not\models F$.
F is *satisfiable* if F has at least one model. Otherwise, it is called *unsatisfiable* or *contradictory*.
A set $\mathbf{M}$ of formulas is *satisfiable* if there is an assignment $\mathcal{A}$ which is a model for each formula in $\mathbf{M}$. In this case, $\mathcal{A}$ is called a *model* of $\mathbf{M}$, and we write $\mathcal{A} \models \mathbf{M}$. [Note the overloading of notation.]
F is called *valid* or a *tautology* if every suitable assignment for F is a model for F. In this case we write $\models F$, and otherwise $\not\models F$.

Exercise 5 (hand-in) Give a model for $\neg(B \wedge F) \vee \neg I$.

1.13 Example
$A \vee \neg A$ is a tautology.
[This is established by the following truth table:

$\mathcal{A}(A)$	$\mathcal{A}(\neg A)$	$\mathcal{A}(A \vee \neg A)$
0	1	1
1	0	1

]

Exercise 6 (hand-in) Show the following.

1. $A \wedge \neg A$ is unsatisfiable.
2. $A \to \neg A$ is satisfiable.

1.14 Theorem
A formula F is a tautology if and only if $\neg F$ is unsatisfiable.

Proof: F is a tautology
iff every suitable assignment for F is a model for F
iff every suitable assignment for F (hence also for $\neg F$) is not a model for $\neg F$
iff $\neg F$ does not have a model
iff $\neg F$ is unsatisfiable ■

1.15 Definition
A formula G is a (*logical*) *consequence* of a set $M = \{F_1, \ldots, F_n\}$ of formulas if for every assignment $\mathcal{A}$ which is suitable for G and for all elements of M, it follows that whenever $\mathcal{A} \models F_i$ for all $i = 1, \ldots, n$, then $\mathcal{A} \models G$.
If G is a logical consequence of M, we write $M \models G$ and say M *entails* G. [Note the overloading of notation!]

01/13/11

1.16 Theorem
The following assertions are equivalent.

1. G is a consequence of $\{F_1, \ldots, F_n\}$.
2. $((\bigwedge_{i=1}^{n} F_i) \to G)$ is a tautology.
3. $((\bigwedge_{i=1}^{n} F_i) \wedge \neg G)$ is unsatisfiable.

Exercise 7 (hand-in) Show that an assignment is a model for $(\bigwedge_{i=1}^{n} F_i)$ if and only if it is a model for $\{F_1, \ldots, F_n\}$.

Exercise 8 (no hand-in) Prove that 1. and 2. of Theorem 1.16 are equivalent. [Hint: Use Exercise 7.]

1.17 Example
Using Theorem 1.16, we can determine logical consequences using truth tables.
E.g., *modus ponens*: $\{P, P \to Q\} \models Q$.
We have to show: $(P \wedge (P \to Q)) \to Q$ is a tautology.

$\mathcal{A}(P)$	$\mathcal{A}(Q)$	$\mathcal{A}(P \to Q)$	$\mathcal{A}(P \wedge (P \to Q))$	$\mathcal{A}((P \wedge (P \to Q)) \to Q)$
0	0	1	0	1
0	1	1	0	1
1	0	0	0	1
1	1	1	1	1

Exercise 9 (no hand-in) Express modus tollens, modus tollendo ponens, and modus ponendo tollens in propositional logic.

Exercise 10 (no hand-in) Show, using truth tables, that the modi from Exercise 9 are valid.

1.18 Remark
The symbols $\neg$, $\vee$, $\wedge$ (and also $\to$, $\leftrightarrow$) are called *connectives*.

1.3 Equivalence

[Schöning, 1989, Chapter 1.2]

1.19 Definition
Formulas F and G are (*semantically*) *equivalent* (written $F \equiv G$) if for every assignment $\mathcal{A}$ that is suitable for F and G, $\mathcal{A}(F) = \mathcal{A}(G)$.

1.20 Example
$A \vee B \equiv B \vee A$. (*commutativity* of $\vee$)
[

$\mathcal{A}(A)$	$\mathcal{A}(B)$	$\mathcal{A}(A \vee B)$	$\mathcal{A}(B \vee A)$
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	1

]
$A \vee \neg A \equiv B \vee \neg B$. [truth table]

1.21 Example
$F \equiv G$ iff $\models (F \leftrightarrow G)$. [truth table]

1.22 Theorem
The following hold for all formulas F, G, and H.

$F \wedge F \equiv F$	$F \vee F \equiv F$	Idempotency
$F \wedge G \equiv G \wedge F$	$F \vee G \equiv G \vee F$	Commutativity
$(F \wedge G) \wedge H \equiv F \wedge (G \wedge H)$	$(F \vee G) \vee H \equiv F \vee (G \vee H)$	Associativity
$F \wedge (G \vee H) \equiv (F \wedge G) \vee (F \wedge H)$	$F \vee (G \wedge H) \equiv (F \vee G) \wedge (F \vee H)$	Distributivity
$\neg\neg F \equiv F$		Double Negation
$\neg(F \wedge G) \equiv \neg F \vee \neg G$	$\neg(F \vee G) \equiv \neg F \wedge \neg G$	de Morgan's Laws

Proof: Straightforward using truth tables. ■

Exercise 11 (hand-in) Prove that 2. and 3. of Theorem 1.16 are equivalent.

Exercise 12 (no hand-in) Translate the "secrets" of the centenarian (slide 14 of slideset 1) into formulas, where B stands for *beer for dinner*, F for *fish for dinner* and I for *ice cream for dinner*.

Exercise 13 (no hand-in) Show that the claim on slide 14 of slideset 1 holds.

1.23 Remark
Disjunction is dispensable. $[F \vee G \equiv \neg(\neg F \wedge \neg G)]$
Alternatively, conjunction is dispensable. $[F \wedge G \equiv \neg(\neg F \vee \neg G)]$

1.24 Remark
Let $F \uparrow G = \neg(F \wedge G)$.
$\neg F \equiv \neg(F \wedge F) \equiv F \uparrow F$.
$F \vee G \equiv \neg(\neg F \wedge \neg G) \equiv \neg F \uparrow \neg G \equiv (F \uparrow F) \uparrow (G \uparrow G)$
$F \wedge G \equiv \neg\neg(F \wedge G) \equiv \neg(F \uparrow G) \equiv (F \uparrow G) \uparrow (F \uparrow G)$.

1.25 Remark (The contraposition principle)
$\{F\} \models G$ iff $\{\neg G\} \models \neg F$.
$[\{F\} \models G$ iff $F \to G$ is a tautology (Theorem 1.16).
$F \to G \equiv \neg F \vee G \equiv \neg(\neg G) \vee (\neg F) \equiv (\neg G) \to (\neg F)$.
$(\neg G) \to (\neg F)$ is a tautology iff $\{\neg G\} \models \neg F$ (Theorem 1.16)]

01/18/11

1.4 Normal Forms

[Schöning, 1989, Chapter 1.2 cont.]

1.26 Definition
A *literal* is an atomic formula (a *positive* literal) or the negation of an atomic formula (a *negative* literal).
A formula F is in *negation normal form* (NNF) if it is made up only of literals, $\vee$, and $\wedge$.

1.27 Theorem
For every formula F, there is a formula $G \equiv F$ which is in NNF.

Proof: The proof of Theorem 1.30 below shows this as well. ■

1.28 Example
$(\neg(I \vee \neg B) \vee \neg F) \equiv (\neg I \wedge B) \vee \neg F$

1.29 Definition
A formula F is in *conjunctive normal form* (CNF) if it is a conjunction of disjunctions of literals, i.e., if

$$F = \left(\bigwedge_{i=1}^{n} \left(\bigvee_{j=1}^{m} L_{i,j} \right) \right),$$

where the $L_{i,j}$ are literals.
A formula F is in *disjunctive normal form* (DNF) if it is a disjunction of conjunctions of literals, i.e., if

$$F = \left(\bigvee_{i=1}^{n} \left(\bigwedge_{j=1}^{m} L_{i,j} \right) \right),$$

where the $L_{i,j}$ are literals.

1.30 Theorem
For every formula F there is a formula $F_1 \equiv F$ in CNF and a formula $F_2 \equiv F$ in DNF.

Proof: Proof by structural induction.
Induction base: If F is atomic, then it is already in CNF and in DNF.
Induction hypothesis: G has CNF G_1 and DNF G_2, H has CNF H_1 and DNF H_2. *Induction step*: We have 3 cases.
Case 1: F has the form $F = \neg G$.
Then

$$F \equiv \neg G_1 \equiv \neg \left(\bigwedge_{i=1}^{n} \left(\bigvee_{j=1}^{m} L_{i,j} \right) \right) \equiv \left(\bigvee_{i=1}^{n} \left(\bigwedge_{j=1}^{m} \neg L_{i,j} \right) \right) \equiv \left(\bigvee_{i=1}^{n} \left(\bigwedge_{j=1}^{m} \overline{L_{i,j}} \right) \right),$$

where

$$\overline{L_{i,j}} = \begin{cases} A & \text{if } L_{i,j} = \neg A \\ \neg A & \text{if } L_{i,j} = A \end{cases}$$

and the latter formula is in DNF as required. Analogously, we can obtain from G_2 a CNF formula equivalent to F.
Case 2: F has the form $F = G \vee H$.
Then $F \equiv G_2 \vee H_2$, which is in DNF.
Further,

$$F \equiv G_1 \vee H_1 \equiv \left(\bigwedge_{i=1}^{n} \left(\bigvee_{j=1}^{m} K_{i,j} \right) \right) \vee \left(\bigwedge_{k=1}^{o} \left(\bigvee_{l=1}^{p} L_{k,l} \right) \right) \equiv \left(\bigwedge_{i=1}^{n} \left(\bigwedge_{j=1}^{o} \left(\bigvee_{k=1}^{m} K_{i,j} \vee \bigvee_{l=1}^{p} L_{k,l} \right) \right) \right),$$

which is in CNF.
Case 3: F has the form $F = G \wedge H$.
This case is analogous to Case 2. ■

Exercise 14 (hand-in) Show by structural induction: For any formula F (with all brackets written), we have $b(F) \leq c(F)$, where $b(F)$ is the number of all opening brackets in F, and $c(F)$ is the number of all connectives in F.

01/25/2011

1.31 Remark
Structural induction is a fundamental proof technique, comparable with natural induction.

Exercise 15 (no hand-in) Transform $\neg((A \vee B) \wedge (C \vee D) \wedge (E \vee F))$ into CNF.

1.32 Remark
DNF via truth table.
If, e.g.,

$\mathcal{A}(A)$	$\mathcal{A}(B)$	$\mathcal{A}(C)$	$\mathcal{A}(F)$
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

then a DNF for F is $(\neg A \wedge \neg B \wedge \neg C) \vee (A \wedge \neg B \wedge \neg C) \vee (A \wedge \neg B \wedge C)$.

Exercise 16 (no hand-in) Give a CNF for the formula F in Remark 1.32.

1.33 Definition
Two formulas F and G are *equisatisfiable* if the following holds: F has a model if and only if G has a model.

Exercise 17 (no hand-in) Show the following: For all formulas F_i $(i = 1, 2, 3)$, $F_1 \vee (F_2 \wedge F_3)$ and $(F_1 \vee E) \wedge (E \leftrightarrow (F_2 \wedge F_3))$ are equisatisfiable (E is a propositional variable not occurring in F_1, F_2, F_3).

1.5 Tableaux Algorithm

[Ben-Ari, 1993, Chapter 2.6, strongly modified]

Translating truth tables directly into an algorithm is very expensive.
We take the following approach:
For showing $F_1, \ldots, F_n \models G$, if suffices to show that $F = F_1 \wedge \cdots \wedge F_n \wedge \neg G$ is unsatisfiable (Theorem 1.16).

We attempt to construct a model for F in such a way that, if and only if the construction fails, we know that F is unsatisfiable.

1.34 Definition
Let F be a formula in NNF. A *tableau branch* for F is a set of formulas, defined inductively as follows.

- $\{F\}$ is a tableau branch for F.
- If T is a tableau branch for F and $G \wedge H \in T$, then $T \cup \{G, H\}$ is a tableau branch for F.
- If T is a tableau branch for F and $G \vee H \in T$, then $T \cup \{G\}$ is a tableau branch for F and $T \cup \{H\}$ is a tableau branch for F.

A *tableau* for F is a set of tableau branches for F.
A tableau branch is *closed* if it contains an atomic formula A and the literal $\neg A$. Otherwise, it is *open*.
A tableau branch T is called *complete* if it satisfies the following conditions.

- T is open.
- If $G \wedge H \in T$, then $\{G, H\} \subseteq T$.
- If $G \vee H \in T$, then $G \in T$ or $H \in T$.

A tableau M for F is called *complete* if it satisfies the following conditions.

- If $G \vee H \in T \in M$, and T is open, then there are branches $S_1 \in M$ and $S_2 \in M$ with $\{G\} \cup T \subseteq S_1$ and $\{H\} \cup T \subseteq S_2$.
- All branches of M are complete or closed.

A tableau is *closed* if it is complete and all its branches are closed.

If F is not in NNF, then a tableau (resp., tableau branch) for F is a tableau (resp. tableau branch) for an NNF of F.

1.35 Example
Consider $(\neg I \wedge B) \vee \neg F$, for which a complete (but not closed) tableau is $\{\{(\neg I \wedge B) \vee \neg F, \neg I \wedge B, \neg I, B\}, \{(\neg I \wedge B) \vee \neg F, \neg F\}\}$.

Exercise 18 (hand-in) Give a complete tableau (as set of sets of formulas) for $(\neg A \wedge \neg B \wedge \neg C) \vee (A \wedge \neg B \wedge \neg C)$.

1.36 Remark
Tableaux can be represented graphically (blackboard).

1.37 Theorem (Soundness)
A formula F is satisfiable if there is a complete tableau branch for F.

1.38 Theorem (Completeness)
If a formula F is satisfiable, then there is a complete tableau branch for F.

01/27/11

1.39 Theorem
A formula F is

1. unsatisfiable if and only if there is a closed tableau for F,
2. a tautology if and only if there is a closed tableau for $\neg F$.

1.40 Example
Modus Ponens holds if $(P \wedge (P \to Q)) \to Q$ is a tautology. We construct a complete tableau (blackboard) for $\neg((P \wedge (P \to Q)) \to Q)$, which turns out to be closed.

Exercise 19 (no hand-in) Do the same as in Example 1.40 for Modus Tollens.

Exercise 20 (hand-in) Show $\{A \to (B \to C)\} \models (A \to B) \to (A \to C)$ using the tableaux algorithm.

1.41 Lemma
Let F be a formula, T be a complete tableau branch for F, and $L_1, \ldots, L_n$ be all the literals contained in T. Then any assignment $\mathcal{A}$ with $\mathcal{A}(L_1 \wedge \cdots \wedge L_n) = 1$ is a model for F.

Proof: We show by structural induction, that $\mathcal{A}$ is a model for each formula F' in T.
Induction Base: Let $F' = L$ be a literal. Then by definition $\mathcal{A}(F') = 1$.
Induction Hypothesis: $\mathcal{A}(G) = \mathcal{A}(H) = 1$ for $G, H \in T$.
Induction Step: (1) Let $F' = G \wedge H \in T$. Then $G \in T$ and $H \in T$. By IH, $\mathcal{A}(F') = \mathcal{A}(G \wedge H) = 1$. (2) Let $F' = G \vee H$. Then $G \in T$ or $H \in T$. By IH, $\mathcal{A}(G) = 1$ or $\mathcal{A}(H) = 1$, hence $\mathcal{A}(F') = 1$. (3) The case $F' = \neg G \in T$ cannot happen since all formulas are in NNF, and the literal case was dealt with in the induction base. ■

Proof of Theorem 1.37: By Lemma 1.41, we obtain that F has a model, hence it is satisfiable. ■

02/08/2011

1.42 Example
Is the following formula valid? satisfiable? unsatisfiable?

$$(((A \to B) \to A) \to A)$$

(done on whiteboard)

02/17/2011

Proof of Theorem 1.38: First note the following, for any assignment M and all formulas G and H:

- If $M \models G \wedge H$, then $M \models G$ and $M \models H$.
- if $M \models G \vee H$, then $M \models G$ or $M \models H$.

Since F is satisfiable, it has a model M. Construct a tableau branch T for F recursively as follows.

- If $G \wedge H \in T$, set $T := T \cup \{G, H\}$.
- If $G \vee H \in T$ with $M \models G$, set $T := T \cup \{G\}$, otherwise set $T := T \cup \{H\}$.

The recursion terminates since only subformulas of F are added and sets cannot contain duplicate elements. The resulting T is a complete tableau branch, and $M \models T$, by definition. ■

Proof of Theorem 1.39:
We prove Statement 1. Statement 2 is shown in Exercise 21.
Let A be the statement "F is unsatisfiable", and let B be the statement "F has a closed tableau".
We need to show: $A \equiv B$, for which it suffices to show that $A \leftrightarrow B \equiv (A \to B) \wedge (B \to A)$ is valid.
By the contraposition principle, it therefore suffices to show that $(\neg B \to \neg A) \wedge (\neg A \to \neg B) \equiv (\neg B \leftrightarrow \neg A)$ is valid, i.e., that $\neg A \equiv \neg B$.
$\neg A$ is the statement "F is not unsatisfiable", i.e. "F is satisfiable".
$\neg B$ is the statment "F does not have a closed tableau". Since, every formula has a complete tableau, this is equivalent to the statement "F has a complete tableau branch".
It thus remains to show: *F is satisfiable if and only if F has a complete tableau branch.* This was shown in Theorems 1.37 and 1.38. ■

1.43 Remark
In short, Statement 1 of Theorem 1.39 holds because it expresses the contrapositions of Theorem 1.37 and 1.38.

Exercise 21 (hand-in) Show Theorem 1.39 2.

Exercise 22 (hand-in) For any formula F, let F' be the formula obtained from F by replacing all $\vee$ by $\wedge$, and by replacing all $\wedge$ by $\vee$. Furthermore, let $\overline{F}$ be obtained from F by replacing each occurrence of an atomic formula A in F by $\neg A$.
Example: For $F = (A \wedge B) \vee \neg C$, we have $F' = (A \vee B) \wedge \neg C$ and $\overline{F} = (\neg A \wedge \neg B) \vee \neg\neg C$; and $\overline{F}' = (\neg A \vee \neg B) \wedge \neg\neg C$.
Show by structural induction: $F \equiv \neg\overline{F}'$ for each formula F.

2 First-order Predicate Logic

2.1 Example
Difficult/impossible to model in propositional logic:

- For all $n \in \mathbb{N}$, $n! \geq n$.

2.2 Example
Difficult/impossible to model in propositional logic:

1. Healthy beings are not dead.
2. Every cat is alive or dead.
3. If somebody owns something, (s)he cares for it.
4. A happy cat owner owns a cat and all beings he cares for are healthy.
5. Schrödinger is a happy cat owner.

2.1 Syntax

[Schöning, 1989, Chapter 2.1]

2.3 Definition

- *Variables*: $x_1, x_2, \ldots$ (also y, z, $\ldots$).
- *Function symbols*: $f_1, f_2, \ldots$ (also g, h, $\ldots$), each with an *arity* ($\in\mathbb{N}$) (number of parameters).
  Constants are function symbols with arity 0.
- *Predicate symbols*: $P_1, P_2, \ldots$ (also Q, R, $\ldots$, each with an *arity* ($\in\mathbb{N}$) (number of parameters).

Terms are inductively defined:

- Each variable is a term.
- If f is a function symbol of arity k, and if $t_1, \ldots, t_k$ are terms, then $f(t_1, \ldots, t_k)$ is a term.

Formulas are inductively defined:

- If P is a predicate symbol of arity k, and if $t_1, \ldots, t_k$ are erms, then $P(t_1, \ldots, t_k)$ is a formula (called *atomic*).
- For each formula F, $\neg F$ is a formula.
- For all formulas F and G, $(F \wedge G)$ and $(F \vee G)$ are formulas.
- If x is a variable and F is a formula, then $\exists x F$ and $\forall x F$ are formulas.

2.4 Definition
$F \to G$ (respectively, $F \leftrightarrow G$) is shorthand for $\neg F \vee G$ (respectively, $(F \to G) \wedge (G \to F)$).
We also use other notational variants from propositional logic freely.

2.5 Example
The following are formulas (s is a constant).

1. $\forall x(H(x) \to \neg D(x))$
2. $\forall x(C(x) \to (A(x) \vee D(x)))$
3. $\forall x \forall y(O(x, y) \to R(x, y))$

4. $\forall x(P(x) \to (\exists y(O(x,y) \wedge C(y)) \wedge (\forall y(R(x,y) \to H(y)))))$
5. $P(s)$

In 1, predicate symbols are D and H, and x is a term.

Exercise 23 (no hand-in) Identify all predicate symbols and all terms in Example 2.5 3.

2.6 Example
Example 2.1 could be written as

$$\forall n(n \in \mathbb{N} \to n! \geq n),$$

where (with abuse of our introduced formal notation), "$\in \mathbb{N}$" is a unary predicate symbol, "$\geq$" is a binary predicate symbol, and "!" is a unary function symbol, written postfix.

Exercise 24 (no hand-in) Determine all predicate symbols and all function symbols, with arities, of the formula

$$\forall \varepsilon \exists \delta \forall x((\varepsilon > 0 \wedge \delta > 0) \to (|x-2| < \delta \to |x^3 - 2^3| < \varepsilon)).$$

02/22/2011

2.7 Definition
If a formula F is part of a formula G, then it is called a *subformula* of G.
An occurrence of a variable x in a formula F is *bound* if it occurs within a subformula of F of the form $\exists xG$ or $\forall xG$. Otherwise it is *free*.
A formula without free variables is *closed*. A formula with free variables is *open*.
$\exists$, $\forall$ are *quantifiers*, $\vee$, $\wedge$, $\neg$, $\to$, $\leftrightarrow$ are *connectives*.

2.8 Example
All subformulas of $\forall x(C(x) \to (A(x) \vee D(x)))$:
$C(x)$, $A(x)$, $D(x)$, $A(x) \vee D(x)$, $C(x) \to (A(x) \vee D(x))$, $\forall x(C(x) \to (A(x) \vee D(x)))$.

2.9 Example
In the formula $P(x) \wedge \forall x(P(x) \to Q(f(x)))$, the first occurrence of x is free, the others are bound.

Exercise 25 (no hand-in) Give all subformulas of Exercise 2.5 4. Which of them are closed? Which of them are open?

2.2 Semantics

[Schöning, 1989, Chapter 2.1 cont.]

2.10 Definition
A *structure* is a pair $\mathcal{A} = (U_\mathcal{A}, I_\mathcal{A})$, with $U_\mathcal{A} \neq \emptyset$ a set (*ground set* or *universe*) and $I_\mathcal{A}$ a mapping which maps

- each k-ary predicate symbol P to a k-ary predicate (relation) on $U_\mathcal{A}$ (if $I_\mathcal{A}$ is defined for P)

- each k-ary function symbol f to a k-ary function on $U_{\mathcal{A}}$ (if $I_{\mathcal{A}}$ is defined for f)
- each variable x to an alement of $U_{\mathcal{A}}$ (if $I_{\mathcal{A}}$ is defined for x).

Write $P^{\mathcal{A}}$ for $I_{\mathcal{A}}(P)$ etc. $\mathcal{A}$ is *suitable* for a formula F if $I_{\mathcal{A}}$ is defined for all predicate and function symbols in F and for all free variables in F.

2.11 Example

$$F = \forall x \forall y (P(a) \wedge (P(x) \rightarrow (P(s(x)) \wedge Q(x,x) \wedge ((P(y) \wedge Q(x,y)) \rightarrow Q(x,s(y))))))$$

Structure $(U_{\mathcal{A}}, I_{\mathcal{A}})$:

$$\begin{aligned} U_{\mathcal{A}} &= \mathbb{N} \\ a^{\mathcal{A}} &= 0 (\in \mathbb{N}) \\ s^{\mathcal{A}} &: n \mapsto n+1 \\ P^{\mathcal{A}} &= \mathbb{N} \qquad (= U_{\mathcal{A}}) \\ Q^{\mathcal{A}} &= \{(n,k) \mid n \leq k\} \end{aligned}$$

Another structure $(U_{\mathcal{B}}, I_{\mathcal{B}})$:

$$\begin{aligned} U_{\mathcal{B}} &= \{\text{☺}, \text{☹}\} \\ a^{\mathcal{B}} &= \text{☺} \\ s^{\mathcal{B}} &: \text{☺} \mapsto \text{☹}; \text{☹} \mapsto \text{☹} \\ P^{\mathcal{B}} &= U_{\mathcal{B}} \\ Q^{\mathcal{B}} &= \{(\text{☹}, \text{☹})\} \end{aligned}$$

Exercise 26 (hand-in) Give a structure for the formula

$$\forall x \forall y (Q(x,y) \rightarrow Q(y,x)).$$

2.12 Definition
F a formula. $\mathcal{A} = (U_{\mathcal{A}}, I_{\mathcal{A}})$ a suitable structure for F.
Define for each term t in F its *value* $t^{\mathcal{A}}$:

1. If $t = x$ is a variable, $t^{\mathcal{A}} = x^{\mathcal{A}}$.
2. If $t = f(t_1, \ldots, t_k)$, then $t^{\mathcal{A}} = f^{\mathcal{A}}(t_1^{\mathcal{A}}, \ldots, t_k^{\mathcal{A}})$.

Define for F its *truth value* $\mathcal{A}(F)$ as follows, where $\mathcal{A}_{[x/u]}$ is identical to $\mathcal{A}$ except $x^{\mathcal{A}_{[x/u]}} = u$.

1. $\mathcal{A}(P(t_1, \ldots, t_k)) = \begin{cases} 1, & \text{if } (\mathcal{A}(t_1), \ldots, \mathcal{A}(t_k)) \in P^{\mathcal{A}} \\ 0, & \text{otherwise} \end{cases}$

2. $\mathcal{A}(H \wedge G) = \begin{cases} 1, & \text{if } \mathcal{A}(H) = 1 \text{ and } \mathcal{A}(G) = 1 \\ 0, & \text{otherwise} \end{cases}$

3. $\mathcal{A}(H \vee G) = \begin{cases} 1, & \text{if } \mathcal{A}(H) = 1 \text{ or } \mathcal{A}(G) = 1 \\ 0, & \text{otherwise} \end{cases}$

4. $\mathcal{A}(\neg G) = \begin{cases} 1, & \text{if } \mathcal{A}(G) = 0 \\ 0, & \text{otherwise} \end{cases}$

5. $\mathcal{A}(\forall x G) = \begin{cases} 1, & \text{if for all } u \in U_{\mathcal{A}}, \mathcal{A}_{[x/u]}(G) = 1 \\ 0, & \text{otherwise} \end{cases}$

6. $\mathcal{A}(\exists x G) = \begin{cases} & \text{if there exists some } u \in U_{\mathcal{A}} \text{ s.t. } \mathcal{A}_{[x/u]}(G) = 1 \\ 0, & \text{otherwise} \end{cases}$

If $\mathcal{A}(F) = 1$, we write $\mathcal{A} \models F$ and say F *is true in* $\mathcal{A}$ or $\mathcal{A}$ *is a model for* F.
F is *valid* (or a *tautology*, written $\models F$) if $\mathcal{A} \models F$ for every suitable structure $\mathcal{A}$ for F. F is *satisfiable* if there is $\mathcal{A}$ with $\mathcal{A} \models F$, and otherwise it is *unsatisfiable*.

2.13 Example
Consider the formula $F = \exists x \forall y Q(x, y)$ under the structure $\mathcal{A} = (U_{\mathcal{A}}, I_{\mathcal{A}})$ from Example 2.11. We show $\mathcal{A}(F) = 1$.
First note that $0 \leq n$ for all $n \in \mathbb{N}$, i.e. $\mathcal{A}_{[x/0][y/n]}(Q(x, y)) = 1$ for all $n \in \mathbb{N} = U_{\mathcal{A}}$. Thus, $\mathcal{A}_{[x/0]}(\forall y Q(x, y)) = 1$ and therefore $\mathcal{A}(\exists x \forall y Q(x, y)) = 1$ as desired.

Exercise 27 (hand-in) Show that $(U_{\mathcal{B}}, I_{\mathcal{B}})$ as in Example 2.11 is a model for

$$\forall x \exists y (P(x) \wedge Q(s(x), y)).$$

2.14 Remark
Many notions and results carry over directly from propositional logic: *logical consequence*, *equivalence of formulas*, Theorem 1.16, Theorem 1.22, etc.

2.15 Remark
Predicate logic "degenerates" to propositional logic if either all predicate symbols have arity 0, or if no variables are used. For the latter, a formula like $(Q(a) \wedge \neg R(f(b), c)) \wedge P(a, b)$ can be written as the propositional formula $(A \wedge \neg B) \wedge C$ with A for $Q(a)$, B for $R(f(b), c)$, and C for $P(a, b)$.

02/24/2011

2.16 Remark
We deal with *first-order* predicate logic. Second-order predicate logic also allows to quantify over predicate symbols.

Exercise 28 (hand-in) Sentence 1 of Example 2.2 can be written as.

$$\forall x (\text{Healthy}(x) \rightarrow \neg \text{Dead}(x)).$$

Translate all other sentences from Example 2.2. Use schroedinger as a constant symbol and use only the following predicate symbols:

unary: Healthy, Dead, Cat, Alive, HappyCatOwner
binary: owns, cares

Exercise 29 (no hand-in) Sketch, how you would formally prove, using Exercise 28, that Schrödinger's cat is alive.

2.3 Equivalence

[Schöning, 1989, Chapter 2.2]

2.17 Theorem
The following hold for arbitrary formulas F and G.

$$\neg\forall xF \equiv \exists x\neg F \qquad \neg\exists xF \equiv \forall x\neg F$$
$$\forall xF \wedge \forall xG \equiv \forall x(F \wedge G) \qquad \exists xF \vee \exists xG \equiv \exists x(F \vee G)$$
$$\forall x\forall yF \equiv \forall y\forall xF \qquad \exists x\exists yF \equiv \exists y\exists xF$$

If x does not occur free in G, then

$$\forall xF \wedge G \equiv \forall x(F \wedge G) \qquad \forall xF \vee G \equiv \forall x(F \vee G)$$
$$\exists xF \wedge G \equiv \exists x(F \wedge G) \qquad \exists xF \vee G \equiv \exists x(F \vee G)$$

Proof: We show only $\forall xF \wedge \forall xG \equiv \forall x(F \wedge G)$:
$\mathcal{A}(\forall xF \wedge \forall xG) = 1$
iff $\mathcal{A}(\forall xF) = 1$ and $\mathcal{A}(\forall xG) = 1$
iff for all $u \in U_{\mathcal{A}}$, $\mathcal{A}_{[x/u]}(F) = 1$ and for all $v \in U_{\mathcal{A}}$, $\mathcal{A}_{[x/v]}(G) = 1$
iff for all $u \in U_{\mathcal{A}}$, $\mathcal{A}_{[x/u]}(F) = 1$ and $\mathcal{A}_{[x/u]}(G) = 1$
iff $\mathcal{A}(\forall x(F \wedge G)) = 1$ ■

Exercise 30 (hand-in) Show, that the first statement of Theorem 2.17, $\neg\forall xF \equiv \exists x\neg F$, holds.

Exercise 31 (hand-in) Show, that $\forall x\exists yP(x, y) \not\equiv \exists u\forall vP(v, u)$.

Exercise 32 (no hand-in) Show, that $\forall x\exists y(P(x) \wedge Q(y)) \equiv \exists y\forall x(P(x) \wedge Q(y))$.

Exercise 33 (no hand-in) Show, that

$$\forall x(P(x) \rightarrow (\exists y(O(x, y) \wedge C(y)) \wedge (\forall z(R(x, z) \rightarrow H(z)))))$$

and

$$\forall z\forall x\exists y((P(x) \rightarrow (O(x, y) \wedge C(y))) \wedge ((P(x) \wedge R(x, z)) \rightarrow H(z)))$$

are equivalent.

2.18 Definition
A *substitution* $[x/t]$, where x is a variable and t a term, is a mapping which maps each formula G to the formula $G[x/t]$, which is obtained from G by replacing all free occurrences of x by t.

2.19 Example
$(P(x,y) \wedge \forall y Q(x,y))[x/a][y/f(x)] = P(a,f(x)) \wedge \forall y Q(a,y)$

Exercise 34 (no hand-in) What is $(\forall x(Q(x,y,z)[y/a])[x/b] \wedge \forall x(P(x,y)[y/x][x/a]))[z/x]$?

Exercise 35 (no hand-in) Show, that, for any formula F in which y does not occur as free variable, $\forall x F \equiv \forall y F[x/y]$.

2.4 Normal Forms

[Schöning, 1989, Chapter 2.2 cont.]

2.20 Definition
A *literal* is an atomic formula (a *positive* literal) or the negation of an atomic formula (a *negative* literal).
A formula F is in *negation normal form* (NNF) if the negation symbol $\neg$ occurs only in literals (and $\rightarrow$, $\leftrightarrow$ don't appear in it).

2.21 Theorem
For every formula F, there is a formula $G \equiv F$ which is in NNF.

Proof: Apply de Morgan, double negation, and $\neg\forall x F \equiv \exists x \neg F$ and $\neg\exists x F \equiv \forall x \neg F$ exhaustively. ■

2.22 Example
$$\begin{aligned}
&\neg(\exists x P(x,y) \vee \forall z Q(z)) \wedge \neg\exists w P(f(a,w)) \\
&\quad \equiv (\neg\exists x P(x,y) \wedge \neg\forall z Q(z)) \wedge \forall w \neg P(f(a,w)) \\
&\quad \equiv (\forall x \neg P(x,y) \wedge \exists z \neg Q(z)) \wedge \forall w \neg P(f(a,w))
\end{aligned}$$

Exercise 36 (no hand-in) Transform all formulas from Example 2.5 into NNF.

2.5 Tableaux Algorithm

[Ben-Ari, 1993, Chapter 5.5, strongly modified]

2.23 Definition
Let F be a formula in NNF. A *tableau branch* for F is a set of formulas, defined inductively as follows.

- $\{F\}$ is a tableau branch for F.

- If T is a tableau branch for F and $G \wedge H \in T$, then $T \cup \{G, H\}$ is a tableau branch for F.
- If T is a tableau branch for F and $G \vee H \in T$, then $T \cup \{G\}$ is a tableau branch for F and $T \cup \{H\}$ is a tableau branch for F.
- If T is a tableau branch for F and $\forall x G \in T$, then $T \cup \{G[x/t]\}$ is a tableau branch for F, where t is any term. 03/08/2011
- If T is a tableau branch for F and $\exists x G \in T$, then $T \cup \{G[x/a]\}$ is a tableau branch for F, where a is a constant symbol which does not occur in T (or in the tableau curently constructed).

A *tableau* for F is a set of tableau branches for F.
A tableau branch is *closed* if it contains an atomic formula A and its negation $\neg A$. Otherwise, it is *open*.
A tableau M for F is called *closed* if for each $T \in M$ there is a closed $T' \in M$ with $T \subseteq T'$.

If F is not in NNF, then a tableau (resp., tableau branch) for F is a tableau (resp. tableau branch) for an NNF of F.

2.24 Theorem (Soundness)
If a closed formula F has a closed tableau, then F is unsatisfiable.

2.25 Theorem (Completeness)
If a closed formula F is unsatisfiable, then there is a closed tableau for F.

2.26 Example
We show $\exists u \forall v P(v, u) \models \forall x \exists y P(x, y)$. I.e. we make a tableau for

$$\exists u \forall v P(v, u) \wedge \exists x \forall y \neg P(x, y),$$

see Figure 1 (left).

2.27 Remark
The (predicate logic) tableaux algorithm does *not* in general provide a means to find out if a formula is satisfiable or falsifiable.
Consider $\forall x \exists y P(x, y) \overset{?}{\models} \exists u \forall v P(v, u)$. If we attempt to make a tableau for

$$\forall x \exists y P(x, y) \wedge \forall u \exists v \neg P(v, u),$$

see for example Figure 1, then the search for closing the tableau does not stop. The reason for this is that the tableau cannot close, but the occurrence of the quantifiers in the formula prompts the algorithm to ever explore new terms for the bound variables.

Exercise 37 (no hand-in) Show, using a tableau, that $\exists x(P(x) \wedge Q(x)) \models \exists x P(x) \wedge \exists y Q(y)$.

Exercise 38 (no hand-in) Show, using a tableau, that $\exists x(O(s, x) \wedge A(x))$ is a logical consequence of the formulas in Exercise 2.5.

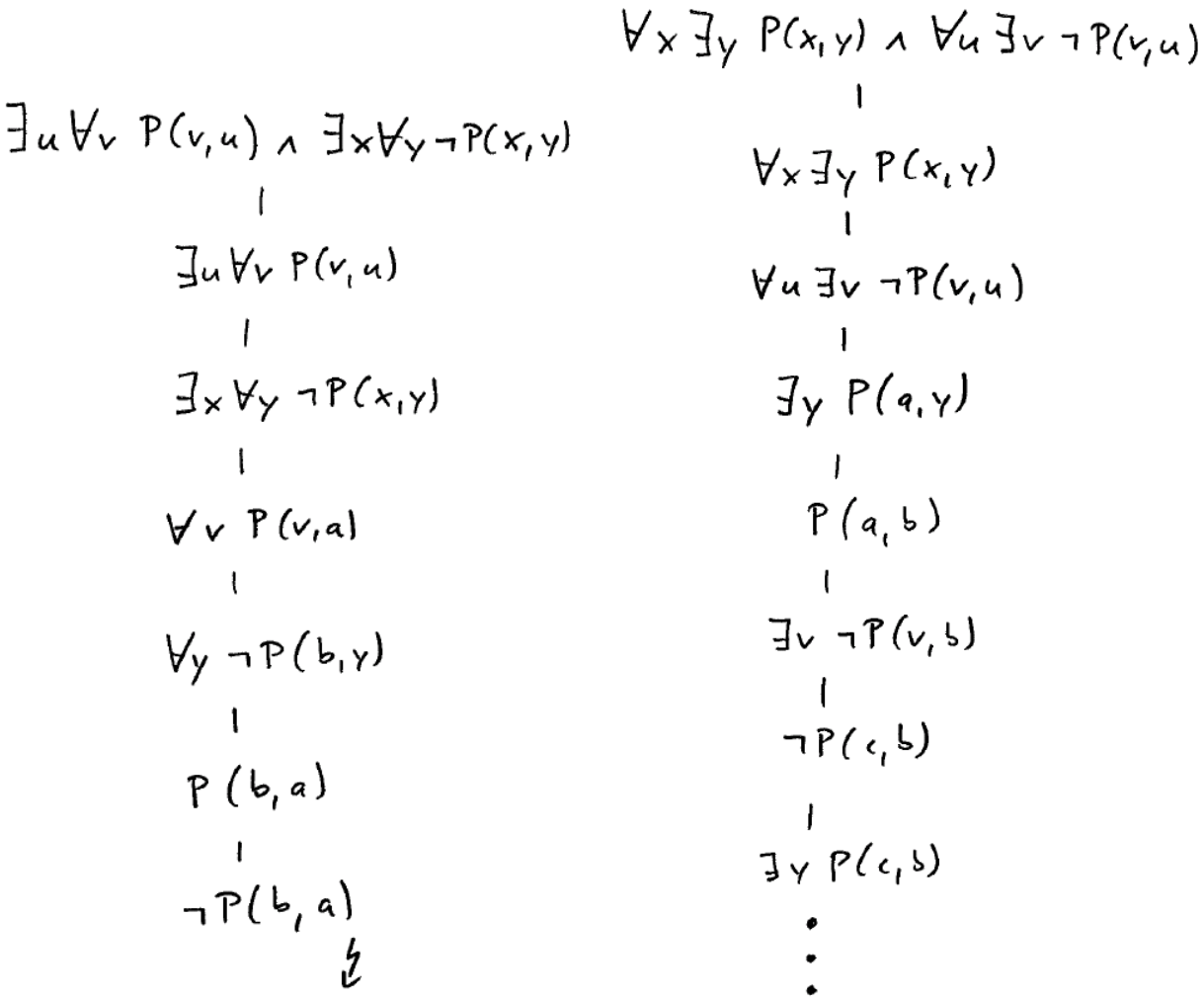

Figure 1: Tableaux for Example 2.26 (left) and Remark 2.27 (right).

Exercise 39 (no hand-in) Show, using a tableau, that $Q(a) \wedge Q(b) \wedge \forall x(P(x) \wedge (Q(x) \rightarrow \neg P(x)))$ is unsatisfiable.

2.28 Remark
While the propositional tableaux algorithm always terminates, this is not the case for the predicate logic tableaux algorithm.

Appendix

1.6 Theoretical Aspects (Propositional Logic)

[Schöning, 1989, Part of Chapter 1.4 plus some more]

1.44 Theorem (monotonicity of propositional logic)
Let M, N be sets of formulas. If $M \subseteq N$ then $\{F \mid M \models F\} \subseteq \{F \mid N \models F\}$.

Proof: Let F be such that $M \models F$.
Let $\mathcal{A}$ be a model for N. Then all formulas in N, and hence all formulas in M, are true under $\mathcal{A}$. Hence $\mathcal{A} \models F$. This holds for all models of N, and hence $N \models F$. ■

Exercise 40 (hand-in) Is the following true or false?
Let M, N be sets of formulas. If $\{F \mid M \models F\} \subseteq \{F \mid N \models F\}$ then $M \subseteq N$.
Prove that your answer is correct.

1.45 Theorem (compactness of propositional logic)
A set M of formulas is satisfiable if and only if every finite subset of it is satisfiable.

Proof:
$\Rightarrow$: Every model for M is also a model for each finite subset of M.
$\Leftarrow$: Assume every finite subset of M is satisfiable.
Let $\{A_1, A_2, \dots\}$ be all propositional variables.
Define M_n to be the set of all elements of M which contains only the propositional variables $A_1, \dots, A_n$.
M_n contains at most 2^{2^n} many formulas with different truth tables.
Thus, there is a set $\mathcal{F}_n = \{F_1, \dots, F_k\} \subseteq M_n$ ($k \leq 2^{2^n}$), such that for every $F \in M$, $F \equiv F_i$ for some i.
Hence, every model for $\mathcal{F}_n$ is a model for M_n.
By assumption, $\mathcal{F}_n$ is satisfiable, say with model $\mathcal{A}_n$.
$\mathcal{A}_n$ is also a model for $M_1, \dots, M_{n-1}$. [$M_i \subseteq M_{i+1}$for all i]
For all $k \in \mathbb{N}$, define $\mathcal{A}(A_k) = \limsup_{n\to\infty} \mathcal{A}_n(A_k)$.
Note: For each $k \in \mathbb{N}$ there exists $n_k \in \mathbb{N}$ s.t. for all $n \geq n_k$ we have $\mathcal{A}_n(A_k) = \mathcal{A}_{n+1}(A_k)$.
It remains to show: $\mathcal{A} \models M$:
Let $F \in M$. Then $F \in M_k$ for some k.
With $n' = \max\{n_1, \dots, n_k\}$ we have that $\mathcal{A}$ and all $\mathcal{A}_n$ with $n \geq n'$ agree on all propositional variables in F.
We have $\mathcal{A}_m \models F$ for all $m \geq \max\{k, n'\}$.
Hence $\mathcal{A} \models F$ as required. ■

Exercise 41 (hand-in) Show: A set M of formulas is unsatisfiable if and only if some finite subset of it is unsatisfiable.

1.46 Definition
A problem with a yes/no answer is *decidable* if there exists an algorithm which terminates on any allowed input of the problem and, upon termination, outputs the correct answer.

1.47 Example
"Is n an even number?" is decidable (allowed input: any $n \in \mathbb{N}$).
[

```
1. If n=1, terminate with output 'No'.
2. If n=0, terminate with output 'Yes'.
3. Set n := n-2.
4. Go to 1.
```

]

1.48 Theorem (decidability of finite entailment)
The problem of deciding whether a finite set M of formulas entails some other formula F is decidable.

Proof: M contains only a finite number of propositional variables. Use truth tables to check whether all models of M are models of F. ■

1.49 Definition
A problem with a yes/no answer is *semi-decidable* if there exists an algorithm which, on any allowed input of the problem, terminates if the answer is 'yes' and outputs the correct answer.

1.50 Theorem (semi-decidability of infinite entailment)
The problem of deciding whether a countably infinite set M of formulas entails some other formula F is semi-decidable.

Proof: $M \models F$ if and only if $M \cup \{\neg F\}$ is unsatisfiable. [Exercise 42]
By the compactness theorem, $M \cup \{\neg F\}$ is unsatisfiable if and only if one of its finite subsets is unsatisfiable. Now use an enumeration $M_1, M_2, \dots$ of all these finite subsets and check satisfiability of each of them in turn, using truth tables. If one of the sets is unsatisfiable, terminate and output that $M \models F$. ■

Exercise 42 (Proof by Contradiction – hand-in) Show: $M \models F$ if and only if $M \cup \{\neg F\}$ is unsatisfiable.

Exercise 43 (no hand-in) Let $\{F_1, F_2, F_3, \dots\}$ be a (countably) infinite set. Give an algorithm with enumerates all its finite subsets.

1.51 Theorem (complexity of finite satisfiability)
The problem of deciding whether a finite set of formulas is satisfiable, is NP-complete.

Proof: See CS740 (or any book on computational complexity theory). ■

1.52 Theorem (complexity of finite entailment)
The problem of deciding whether a finite set of formulas entails some other formula is NP-complete.

Proof: Because of Exercise 42, finite entailment and finite satisfiability can be reduced to each other, hence they have the same complexity. ■

2.6 Theoretical Aspects (Predicate Logic)

[Schöning, 1989, Chapter 2.3 and other sources]

2.29 Theorem (monotonicity of propositional logic)
Let M, N be sets of formulas. If $M \subseteq N$ then $\{F \mid M \models F\} \subseteq \{F \mid N \models F\}$.

Proof: Similar as for propositional logic. ■

2.30 Theorem (compactness of propositional logic)
A set M of formulas is satisfiable if and only if every finite subset of it is satisfiable.

2.31 Theorem (undecidability of predicate logic)
The problem "*Given a formula F, is F valid?*" is undecidable.

Exercise 44 (hand-in) Show, that the problem "*Given a formula F and a finite set of formulas M, is $M \models F$?*" is undecidable. [use Theorem 2.31]

2.32 Theorem (semi-decidability of predicate logic)
The problem "*Given a formula F, is F valid?*" is semi-decidable.

Proof: We have, e.g., the tableaux calculus for this. ■

2.33 Remark
The formula

$$F = \forall x \forall y \forall u \forall v \forall w (P(x, f(x)) \wedge \neg P(y, y) \wedge ((P(u, v) \wedge P(v, w)) \rightarrow P(u, w)$$

is satisfiable but has no finite model (with $U_{\mathcal{A}}$ finite).
$\mathcal{A} = (U_{\mathcal{A}}, I_{\mathcal{A}})$ is a model, where

$$\begin{aligned} U_{\mathcal{A}} &= \mathbb{N} \\ P^{\mathcal{A}} &= \{(m, n) \mid m < n\} \\ f^{\mathcal{A}}(n) &= n + 1 \end{aligned}$$

Assume $B = (U_{\mathcal{B}}, I_{\mathcal{B}})$ is a finite model for F. Let $u_0 \in U_{\mathcal{B}}$ and consider the sequence $(u_i)_{i \in \mathbb{N}}$ with $u_{i+1} = f^{\mathcal{B}}(u_i)$. Since $U_{\mathcal{B}}$ is finite, there exist $i < j$ with $u_i = u_j$. F enforces transitivity of F, hence $(u_i, u_j) \in P^{\mathcal{B}}$. But since $u_i = u_j$ this contradicts $\forall y \neg P(y, y)$.

2.34 Theorem (Löwenheim-Skolem)
If a (finite or) countable set of formulas is satisfiable, then it is satisfiable in a countable domain.

2.35 Remark
According to Theorem 2.34, it is impossible to axiomatize the real numbers in first-order predicate logic.

3 Application: Knowledge Representation for the World Wide Web

[See [Hitzler et al., 2009] for further reading.]

[**Slideset 2**]