Mathematical Logic

Stephen G. Simpson

November 16, 2000

Department of Mathematics
The Pennsylvania State University
University Park, State College PA 16802

http://www.math.psu.edu/simpson/

This is a set of lecture notes for introductory courses in mathematical logic offered at the Pennsylvania State University.

Contents

Chapter 1

Propositional Calculus

1.1 Formulas

Definition 1.1.1. The *propositional connectives* are *negation* ($\neg$), *conjunction* ($\&$), *disjunction* ($\vee$), *implication* ($\Rightarrow$), *biimplication* ($\Leftrightarrow$). They are read as "not", "and", "or", "if-then", "if and only if" respectively. The connectives $\&$, $\vee$, $\Rightarrow$, $\Leftrightarrow$ are designated as *binary*, while $\neg$ is designated as *unary*.

Definition 1.1.2. A *propositional language* L is a set of *propositional atoms* $p, q, r, \ldots$. An *atomic L-formula* is an atom of L.

Definition 1.1.3. The set of *L-formulas* is generated inductively according to the following rules:

1. If p is an atomic L-formula, then p is an L-formula.
2. If A is an L-formula, then $(\neg A)$ is an L-formula.
3. If A and B are L-formulas, then $(A \,\&\, B)$, $(A \vee B)$, $(A \Rightarrow B)$, and $(A \Leftrightarrow B)$ are L-formulas.

Note that rule 3 can be written as follows:

3′. If A and B are L-formulas and b is a binary connective, then (AbB) is an L-formula.

Example 1.1.4. Assume that L contains propositional atoms p, q, r, s. Then

$$(((p \Rightarrow q) \,\&\, (q \vee r)) \Rightarrow (p \vee r)) \Rightarrow \neg\, (q \vee s)$$

is an L-formula.

Definition 1.1.5. If A is a formula, the *degree* of A is the number of occurrences of propositional connectives in A. This is the same as the number of times rules 2 and 3 had to be applied in order to generate A.

Example 1.1.6. The degree of the formula of Example 1.1.4 is 8.

Remark 1.1.7. As in the above example, we omit parentheses when this can be done without ambiguity. In particular, outermost parentheses can always be omitted, so instead of $((\neg A) \Rightarrow B)$ we may write $(\neg A) \Rightarrow B$. But we may not write $\neg A \Rightarrow B$, because this would not distinguish the intended formula from $\neg (A \Rightarrow B)$.

Definition 1.1.8. Let L be a propositional language. A *formation sequence* is finite sequence $A_1, A_2, \ldots, A_n$ such that each term of the sequence is obtained from previous terms by application of one of the rules in Definition 1.1.3. A *formation sequence for* A is a formation sequence whose last term is A. Note that A is an L-formula if and only if there exists a formation sequence for A.

Example 1.1.9. A formation sequence for the L-formula of Example 1.1.4 is

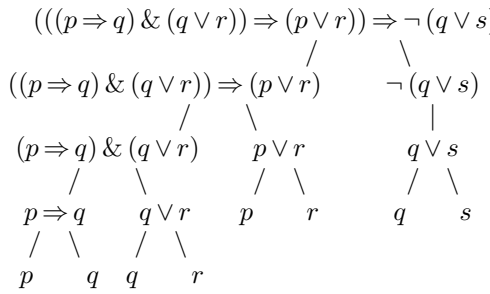
$$p,\ q,\ p \Rightarrow q,\ r,\ q \vee r,\ (p \Rightarrow q) \,\&\, (q \vee r),\ p \vee r,\ ((p \Rightarrow q) \,\&\, (q \vee r)) \Rightarrow (p \vee r),$$

$$s,\ q \vee s,\ \neg (q \vee s),\ (((p \Rightarrow q) \,\&\, (q \vee r)) \Rightarrow (p \vee r)) \Rightarrow \neg (q \vee s)\ .$$

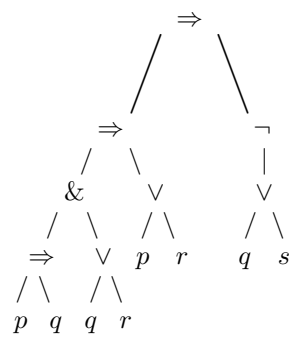
Remark 1.1.10. In contexts where the language L does not need to be specified, an L-formula may be called a *formula*.

Definition 1.1.11. A *formation tree* is a finite rooted dyadic tree where each node carries a formula and each non-atomic formula branches to its immediate subformulas (see the example below). If A is a formula, the *formation tree for* A is the unique formation tree which carries A at its root.

Example 1.1.12. The formation tree for the formula of Example 1.1.4 is



or, in an abbreviated style,



Remark 1.1.13. Note that, if we identify formulas with formation trees in the abbreviated style, then there is no need for parentheses.

Remark 1.1.14. Another way to avoid parentheses is to use Polish notation. In this case the set of L-formulas is generated as follows:

1. If p is an atomic L-formula, then p is an L-formula.
2. If A is an L-formula, then $\neg\, A$ is an L-formula.
3. If A and B are L-formulas and b is a binary connective, then bAB is an L-formula.

For example, the formula of Example 1.1.4 in Polish notation becomes

$$\Rightarrow \Rightarrow \& \Rightarrow p\ q \vee q\ r \vee p\, r \neg \vee q\ s\ .$$

A formation sequence for this formula is

$$p,\ q,\ \Rightarrow pq,\ r,\ \vee qr,\ \& \Rightarrow pq \vee qr,\ \vee pr,\ \Rightarrow \& \Rightarrow pq \vee qr \vee pr,$$
$$s,\ \vee qs,\ \neg \vee qs,\ \Rightarrow \Rightarrow \& \Rightarrow pq \vee qr \vee pr \neg \vee qs\ .$$

Obviously Polish notation is difficult to read, but it has the advantages of being linear and of not using parentheses.

Remark 1.1.15. In our study of formulas, we shall be indifferent to the question of which system of notation is actually used. The only point of interest for us is that each non-atomic formula is uniquely of the form $\neg\, A$ or AbB, where A and B are formulas and b is a binary connective.

1.2 Assignments and Satisfiability

Definition 1.2.1. There are two *truth values*, T and F, denoting truth and falsity.

Definition 1.2.2. Let L be a propositional language. An *L-assignment* is a mapping

$$M : \{p : p \text{ is an atomic } L\text{-formula}\} \to \{\mathrm{T}, \mathrm{F}\}\,.$$

Note that if L has exactly n atoms then there are exactly 2^n different L-assignments.

Lemma 1.2.3. Given an L-assignment M, there is a unique *L-valuation*

$$v_M : \{A : A \text{ is an } L\text{-formula}\} \to \{\mathrm{T}, \mathrm{F}\}$$

given by the following clauses:

1. $v_M(\neg\, A) = \begin{cases} \mathrm{T} & \text{if } v_M(A) = \mathrm{F}\,, \\ \mathrm{F} & \text{if } v_M(A) = \mathrm{T}\,. \end{cases}$

2. $v_M(A \,\&\, B) = \begin{cases} \mathrm{T} & \text{if } v_M(A) = v_M(B) = \mathrm{T}\,, \\ \mathrm{F} & \text{if at least one of } v_M(A), v_M(B) = \mathrm{F}\,. \end{cases}$

3. $v_M(A \vee B) = \begin{cases} \mathrm{T} & \text{if at least one of } v_M(A), v_M(B) = \mathrm{T}\,, \\ \mathrm{F} & \text{if } v_M(A) = v_M(B) = \mathrm{F}\,. \end{cases}$

4. $v_M(A \Rightarrow B) = v_M(\neg\,(A \,\&\, \neg\, B))\,.$

5. $v_M(A \Leftrightarrow B) = \begin{cases} \mathrm{T} & \text{if } v_M(A) = v_M(B)\,, \\ \mathrm{F} & \text{if } v_M(A) \neq v_M(B)\,. \end{cases}$

Proof. The truth value $v_M(A)$ is defined by recursion on L-formulas, i.e., by induction on the degree of A where A is an arbitrary L-formula. □

Remark 1.2.4. The above lemma implies that there is an obvious one-to-one correspondence between L-assignments and L-valuations. If the language L is understood from context, we may speak simply of assignments and valuations.

Remark 1.2.5. Lemma 1.2.3 may be visualized in terms of formation trees. To define $v_M(A)$ for a formula A, one begins with an assignment of truth values to the atoms, i.e., the end nodes of the formation tree for A, and then proceeds upward to the root, assigning truth values to the nodes, each step being given by the appropriate clause.

Example 1.2.6. Consider the formula $(p \Rightarrow q) \Rightarrow (q \Rightarrow r)$ under an assignment M with $M(p) = \mathrm{T}$, $M(q) = \mathrm{F}$, $M(r) = \mathrm{T}$. In terms of the formation tree, this looks like

$$\begin{array}{c} (p \Rightarrow q) \Rightarrow (q \Rightarrow r) \\ \quad / \qquad \backslash \\ p \Rightarrow q \qquad q \Rightarrow r \\ / \quad \backslash \qquad / \quad \backslash \\ p \quad q \qquad q \quad r \\ \mathrm{T} \quad \mathrm{F} \qquad \mathrm{F} \quad \mathrm{T} \end{array}$$

and by applying clause 4 three times we get

$$\begin{array}{c} (p \Rightarrow q) \Rightarrow (q \Rightarrow r) \\ \mathrm{T} \\ \quad / \qquad \backslash \\ p \Rightarrow q \qquad q \Rightarrow r \\ \mathrm{F} \qquad\qquad \mathrm{T} \\ / \quad \backslash \qquad / \quad \backslash \\ p \quad q \qquad q \quad r \\ \mathrm{T} \quad \mathrm{F} \qquad \mathrm{F} \quad \mathrm{T} \end{array}$$

and from this we see that $v_M((p \Rightarrow q) \Rightarrow (q \Rightarrow r)) = \mathrm{T}$.

Remark 1.2.7. The above formation tree with truth values can be compressed and written linearly as

$$\begin{array}{c} (p \Rightarrow q) \Rightarrow (q \Rightarrow r) \\ \mathrm{T\ F\ F\ \ T\ \ F\ T\ T}\ . \end{array}$$

Remark 1.2.8. Note that each clause of Lemma 1.2.3 corresponds to the familiar truth table for the corresponding propositional connective. Thus clause 3 corresponds to the truth table

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

for $\vee$, and clause 4 corresponds to the truth table

A	B	$A \Rightarrow B$
T	T	T
T	F	F
F	T	T
F	F	T

for $\Rightarrow$.

Fix a propositional language L.

Definition 1.2.9. Let M be an assignment. A formula A is said to be *true under* M if $v_M(A) = \mathrm{T}$, and *false under* M if $v_M(A) = \mathrm{F}$.

Definition 1.2.10. A set of formulas S is said to be *satisfiable* if there exists an assignment M which *satisfies* S, i.e., $v_M(A) = \mathrm{T}$ for all $A \in S$.

Definition 1.2.11. Let S be a set of formulas. A formula B is said to be a *logical consequence of* S if it is true under all assignments which satisfy S.

Definition 1.2.12. A formula B is said to be *logically valid* (or a *tautology*) if B is true under all assignments. Equivalently, B is a logical consequence of the empty set.

Remark 1.2.13. B is a logical consequence of $A_1, \ldots, A_n$ if and only if

$$(A_1 \,\&\, \cdots \,\&\, A_n) \Rightarrow B$$

is logically valid. B is logically valid if and only if $\neg\, B$ is not satisfiable.

1.3 Logical Equivalence

Definition 1.3.1. Two formulas A and B are said to be *logically equivalent*, written $A \equiv B$, if each is a logical consequence of the other.

Remark 1.3.2. $A \equiv B$ holds if and only if $A \Leftrightarrow B$ is logically valid.

Exercise 1.3.3. Assume $A_1 \equiv A_2$. Show that

1. $\neg A_1 \equiv \neg A_2$;
2. $A_1 \,\&\, B \equiv A_2 \,\&\, B$;
3. $B \,\&\, A_1 \equiv B \,\&\, A_2$;
4. $A_1 \vee B \equiv A_2 \vee B$;
5. $B \vee A_1 \equiv B \vee A_2$;
6. $A_1 \Rightarrow B \equiv A_2 \Rightarrow B$;
7. $B \Rightarrow A_1 \equiv B \Rightarrow A_2$;
8. $A_1 \Leftrightarrow B \equiv A_2 \Leftrightarrow B$;
9. $B \Leftrightarrow A_1 \equiv B \Leftrightarrow A_2$.

Exercise 1.3.4. Assume $A_1 \equiv A_2$. Show that for any formula C containing A_1 as a part, if we replace one or more occurrences of the part A_1 by A_2, then the resulting formula is logically equivalent to C. (Hint: Use the results of the previous exercise, plus induction on the degree of C.)

Remark 1.3.5. Some useful logical equivalences are:

1. commutative laws:
   (a) $A \,\&\, B \equiv B \,\&\, A$
   (b) $A \vee B \equiv B \vee A$
   (c) $A \Leftrightarrow B \equiv B \Leftrightarrow A$

   Note however that $A \Rightarrow B \not\equiv B \Rightarrow A$.
2. associative laws:
   (a) $A \,\&\, (B \,\&\, C) \equiv (A \,\&\, B) \,\&\, C$
   (b) $A \vee (B \vee C) \equiv (A \vee B) \vee C$

   Note however that $A \Leftrightarrow (B \Leftrightarrow C) \not\equiv (A \Leftrightarrow B) \Leftrightarrow C$, and $A \Rightarrow (B \Rightarrow C) \not\equiv (A \Rightarrow B) \Rightarrow C$.
3. distributive laws:
   (a) $A \,\&\, (B \vee C) \equiv (A \,\&\, B) \vee (A \,\&\, C)$

(b) $A \vee (B \,\&\, C) \equiv (A \vee B) \,\&\, (A \vee C)$

(c) $A \Rightarrow (B \,\&\, C) \equiv (A \Rightarrow B) \,\&\, (A \Rightarrow C)$

(d) $(A \vee B) \Rightarrow C \equiv (A \Rightarrow C) \,\&\, (B \Rightarrow C)$

4. negation laws:

(a) $\neg\,(A \,\&\, B) \equiv (\neg\, A) \vee (\neg\, B)$

(b) $\neg\,(A \vee B) \equiv (\neg\, A) \,\&\, (\neg\, B)$

(c) $\neg \neg\, A \equiv A$

(d) $\neg\,(A \Rightarrow B) \equiv A \,\&\, \neg\, B$

(e) $\neg\,(A \Leftrightarrow B) \equiv (\neg\, A) \Leftrightarrow B$

(f) $\neg\,(A \Leftrightarrow B) \equiv A \Leftrightarrow (\neg\, B)$

5. implication laws:

(a) $A \Rightarrow B \equiv \neg\,(A \,\&\, \neg\, B)$

(b) $A \Rightarrow B \equiv (\neg\, A) \vee B$

(c) $A \Rightarrow B \equiv (\neg\, B) \Rightarrow (\neg\, A)$

(d) $A \Leftrightarrow B \equiv (A \Rightarrow B) \,\&\, (B \Rightarrow A)$

(e) $A \Leftrightarrow B \equiv (\neg\, A) \Leftrightarrow (\neg\, B)$

Definition 1.3.6. A formula is said to be in *disjunctive normal form* if it is of the form $A_1 \vee \cdots \vee A_m$, where each clause A_i, $i = 1, \ldots, m$, is of the form $B_1 \,\&\, \cdots \,\&\, B_n$, and each B_j, $j = 1, \ldots, n$ is either an atom or the negation of an atom.

Example 1.3.7. Writing $\overline{p}$ as an abbreviation for $\neg\, p$, the formula

$$(p_1 \,\&\, \overline{p_2} \,\&\, p_3) \vee (\overline{p_1} \,\&\, p_2 \,\&\, p_3) \vee (p_1 \,\&\, \overline{p_2} \,\&\, \overline{p_3})$$

is in disjunctive normal form.

Exercise 1.3.8. Show that every propositional formula C is logically equivalent to a formula in disjunctive normal form.

Remark 1.3.9. There are two ways to do Exercise 1.3.8.

1. One way is to apply the equivalences of Remark 1.3.5 to subformulas of C via Exercise 1.3.4, much as one applies the commutative and distributive laws in algebra to reduce every algebraic expression to a polynomial.

2. The other way is to use a truth table for C. The disjunctive normal form of C has a clause for each assignment making C true. The clause specifies the assignment.

Example 1.3.10. Consider the formula $(p \Rightarrow q) \Rightarrow r$. We wish to put this in disjunctive normal form.

Method 1. Applying the equivalences of Remark 1.3.5, we obtain

$$\begin{aligned} (p \Rightarrow q) \Rightarrow r &\equiv r \vee \neg (p \Rightarrow q) \\ &\equiv r \vee \neg \neg (p \,\&\, \neg q) \\ &\equiv r \vee (p \,\&\, \neg q) \end{aligned}$$

and this is in disjunctive normal form.

Method 2. Consider the truth table

	p	q	r	$p \Rightarrow q$	$(p \Rightarrow q) \Rightarrow r$
1	T	T	T	T	T
2	T	T	F	T	F
3	T	F	T	F	T
4	T	F	F	F	T
5	F	T	T	T	T
6	F	T	F	T	F
7	F	F	T	T	T
8	F	F	F	T	F

Each line of this table corresponds to a different assignment. From lines 1, 3, 4, 5, 7 we read off the following formula equivalent to $(p \Rightarrow q) \Rightarrow r$ in disjunctive normal form:

$$(p \,\&\, q \,\&\, r) \vee (p \,\&\, \overline{q} \,\&\, r) \vee (p \,\&\, \overline{q} \,\&\, \overline{r}) \vee (\overline{p} \,\&\, q \,\&\, r) \vee (\overline{p} \,\&\, \overline{q} \,\&\, r) \ .$$

1.4 The Tableau Method

Remark 1.4.1. A more descriptive name for tableaux is satisfiability trees. We follow the approach of Smullyan [2].

Definition 1.4.2. A *signed formula* is an expression of the form TA or FA, where A is a formula. An *unsigned formula* is simply a formula.

Definition 1.4.3. A *signed tableau* is a rooted dyadic tree where each node carries a signed formula. An *unsigned tableau* is a rooted dyadic tree where each node carries an unsigned formula. The *signed tableau rules* are presented in Table 1.1. The *unsigned tableau rules* are presented in Table 1.2. If τ is a (signed or unsigned) tableau, an *immediate extension* of τ is a larger tableau τ' obtained by applying a tableau rule to a finite path of τ.

Definition 1.4.4. Let $X_1, \ldots, X_k$ be a finite set of signed or unsigned formulas. A *tableau starting with* $X_1, \ldots, X_k$ is a tableau obtained from

$$\begin{array}{c} X_1 \\ \vdots \\ X_k \end{array}$$

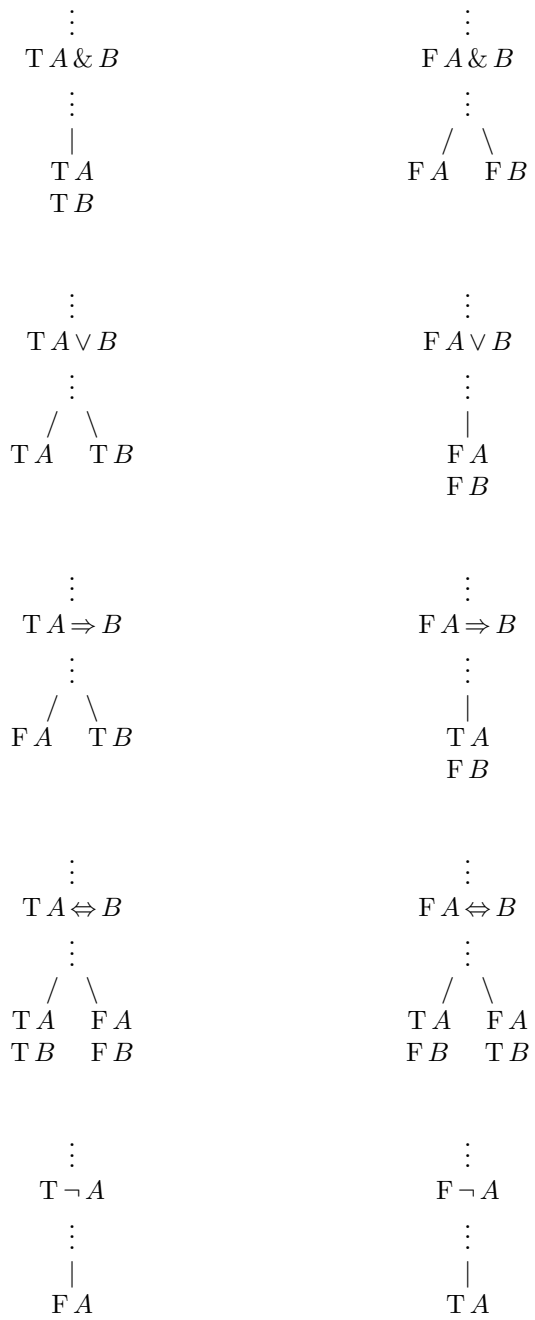


Table 1.1: Signed tableau rules for propositional connectives.

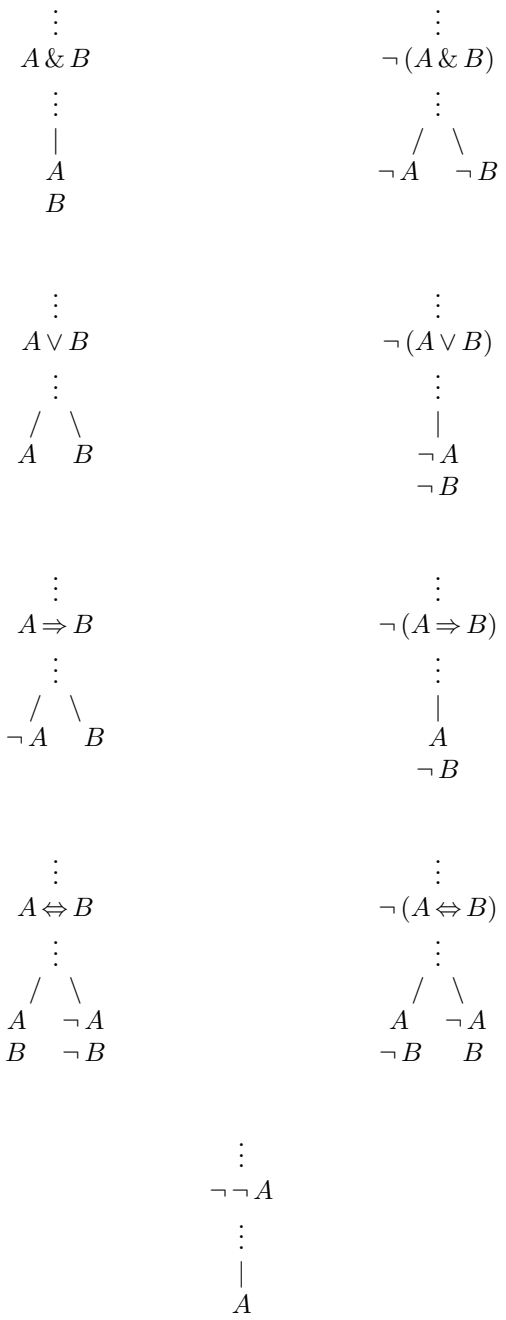


Table 1.2: Unsigned tableau rules for propositional connectives.

by repeatedly applying tableau rules.

Definition 1.4.5. A path of a tableau is said to be *closed* if it contains a conjugate pair of signed or unsigned formulas, i.e., a pair such as $\mathrm{T}A$, $\mathrm{F}A$ in the signed case, or A, $\neg A$ in the unsigned case. A path of a tableau is said to be *open* if it is not closed. A tableau is said to be *closed* if each of its paths is closed.

The tableau method:

1. To test a formula A for validity, form a signed tableau starting with $\mathrm{F}A$, or equivalently an unsigned tableau starting with $\neg A$. If the tableau closes off, then A is logically valid.

2. To test whether B is a logical consequence of $A_1, \ldots, A_k$, form a signed tableau starting with $\mathrm{T}A_1$, ..., $\mathrm{T}A_k$, $\mathrm{F}B$, or equivalently an unsigned tableau starting with $A_1, \ldots, A_k, \neg B$. If the tableau closes off, then B is indeed a logical consequence of $A_1, \ldots, A_k$.

3. To test $A_1, \ldots, A_k$ for satisfiability, form a signed tableau starting with $\mathrm{T}A_1, \ldots, \mathrm{T}A_k$, or equivalently an unsigned tableau starting with $A_1, \ldots, A_k$. If the tableau closes off, then $A_1, \ldots, A_k$ is not satisfiable. If the tableau does not close off, then $A_1, \ldots, A_k$ is satisfiable, and from any open path we can read off an assignment satisfying $A_1, \ldots, A_k$.

The correctness of these tests will be proved in Section 1.5. See Corollaries 1.5.9, 1.5.10, 1.5.11.

Example 1.4.6. Using the signed tableau method to test $(p \,\&\, q) \Rightarrow (q \,\&\, p)$ for logical validity, we have

$$\begin{array}{c} \mathrm{F}\,(p \,\&\, q) \Rightarrow (q \,\&\, p) \\ \mathrm{T}\,p \,\&\, q \\ \mathrm{F}\,q \,\&\, p \\ \mathrm{T}\,p \\ \mathrm{T}\,q \\ \diagup \quad \diagdown \\ \mathrm{F}\,q \qquad \mathrm{F}\,p \end{array}$$

Since (every path of) the tableau is closed, $(p \,\&\, q) \Rightarrow (q \,\&\, p)$ is logically valid.

1.5 The Completeness Theorem

Let $X_1, \ldots, X_k$ be a finite set of signed formulas, or a finite set of unsigned formulas.

Lemma 1.5.1 (the Soundness Theorem). If τ is a finite closed tableau starting with $X_1, \ldots, X_k$, then $X_1, \ldots, X_k$ is not satisfiable.

Proof. Straightforward. □

Definition 1.5.2. A path of a tableau is said to be *replete* if, whenever it contains the top formula of a tableau rule, it also contains at least one of the branches. A *replete tableau* is a tableau in which every path is replete.

Lemma 1.5.3. Any finite tableau can be extended to a finite replete tableau.

Proof. Apply tableau rules until they cannot be applied any more. □

Definition 1.5.4. A tableau is said to be *open* if it is not closed, i.e., it has at least one open path.

Lemma 1.5.5. Let τ be a replete tableau starting with $X_1, \ldots, X_k$. If τ is open, then $X_1, \ldots, X_k$ is satisfiable.

In order to prove Lemma 1.5.5, we introduce the following definition.

Definition 1.5.6. Let S be a set of signed or unsigned formulas. We say that S is a *Hintikka set* if

1. S "obeys the tableau rules", in the sense that if it contains the top formula of a rule then it contains at least one of the branches;
2. S contains no pair of conjugate atomic formulas, i.e., Tp, Fp in the signed case, or $p, \neg p$ in the unsigned case.

Lemma 1.5.7 (Hintikka's Lemma). If S is a Hintikka set, then S is satisfiable.

Proof. Define an assignment M by

$$M(p) \;=\; \begin{cases} \mathrm{T} & \text{if T}p \text{ belongs to } S \\ \mathrm{F} & \text{otherwise} \end{cases}$$

in the signed case, or

$$M(p) \;=\; \begin{cases} \mathrm{T} & \text{if } p \text{ belongs to } S \\ \mathrm{F} & \text{otherwise} \end{cases}$$

in the unsigned case. It is not difficult to see that $v_M(X) = \mathrm{T}$ for all $X \in S$. □

To prove Lemma 1.5.5, it suffices to note that a replete open path is a Hintikka set. Thus, if a replete tableau starting with $X_1, \ldots, X_k$ is open, Hintikka's Lemma implies that $X_1, \ldots, X_k$ is satisfiable.

Combining Lemmas 1.5.1 and 1.5.3 and 1.5.5, we obtain:

Theorem 1.5.8 (the Completeness Theorem). $X_1, \ldots, X_k$ is satisfiable if and only if there is no finite closed tableau starting with $X_1, \ldots, X_k$.

Corollary 1.5.9. $A_1, \ldots, A_k$ is not satisfiable if and only if there exists a finite closed signed tableau starting with TA_1, ..., TA_k, or equivalently a finite closed unsigned tableau starting with $A_1, \ldots, A_k$.

Corollary 1.5.10. A is logically valid if and only if there exists a finite closed signed tableau starting with $\mathrm{F}A$, or equivalently a finite closed unsigned tableau starting with $\neg A$.

Corollary 1.5.11. B is a logical consequence of $A_1, \ldots, A_k$ if and only if there exists a finite closed signed tableau starting with $\mathrm{T}A_1$, ..., $\mathrm{T}A_k$, $\mathrm{F}B$, or equivalently a finite closed unsigned tableau starting with $A_1, \ldots, A_k, \neg B$.

1.6 Trees and König's Lemma

Up to this point, our discussion of trees has been informal. We now pause to make our tree terminology precise.

Definition 1.6.1. A *tree* consists of

1. a set T
2. a function $\ell : T \to \mathbb{N}^+$,
3. a binary relation P on T.

The elements of T are called the *nodes* of the tree. For $x \in T$, $\ell(x)$ is the *level* of x. The relation xPy is read as x *immediately precedes* y, or y *immediately succeeds* x. We require that there is exactly one node $x \in T$ such that $\ell(x) = 1$, called the *root* of the tree. We require that each node other than the root has exactly one immediate predecessor. We require that $\ell(y) = \ell(x) + 1$ for all $x, y \in T$ such that xPy.

Definition 1.6.2. A *subtree of* T is a nonempty set $T' \subseteq T$ such that for all $y \in T'$ and xPy, $x \in T'$. Note that T' is itself a tree, under the restriction of ℓ and P to T'. Moreover, the root of T' is the same as the root of T.

Definition 1.6.3. An *end node* of T is a node with no (immediate) successors. A *path* in T is a set $S \subseteq T$ such that (1) the root of T belongs to S, (2) for each $x \in S$, either x is an end node of T or there is exactly one $y \in S$ such that xPy.

Definition 1.6.4. Let P^* be the transitive closure of P, i.e., the smallest reflexive and transitive relation on T containing P. For $x, y \in T$, we have xP^*y if and only if x *precedes* y, i.e., y *succeeds* x, i.e., there exists a finite sequence $x = x_0 P x_1 P x_2 \cdots x_{n-1} P x_n = y$. Note that the relation P^* is reflexive (xP^*x for all $x \in T$), antisymmetric (xP^*y and yP^*x imply $x = y$), and transitive (xP^*y and yP^*z imply xP^*z). Thus P^* is a partial ordering of T.

Definition 1.6.5. T is *finitely branching* if each node of T has only finitely many immediate successors in T. T is *dyadic* if each node of T has at most two immediate successors in T. Note that a dyadic tree is finitely branching.

Theorem 1.6.6 (König's Lemma). Let T be an infinite, finitely branching tree. Then T has an infinite path.

Proof. Let $\widehat{T}$ be the set of all $x \in T$ such that x has infinitely many successors in T. Note that $\widehat{T}$ is a subtree of T. Since T is finitely branching, it follows by the pigeonhole principle that each $x \in \widehat{T}$ has at least one immediate successor $y \in \widehat{T}$. Now define an infinite path $S = \{x_1, x_2, \ldots, x_n, \ldots\}$ in $\widehat{T}$ inductively by putting $x_1 =$ the root of T, and $x_{n+1} =$ one of the immediate successors of x_n in $\widehat{T}$. Clearly S is an infinite path of T. □

1.7 The Compactness Theorem

Theorem 1.7.1 (the Compactness Theorem, countable case). Let S be a countable set of propositional formulas. If each finite subset of S is satisfiable, the S is satisfiable.

Proof. In brief outline: Form an infinite tableau. Apply König's Lemma to get an infinite path. Apply Hintikka's Lemma.

Details: Let $S = \{A_1, A_2, \ldots, A_i, \ldots\}$. Start with A_1 and generate a finite replete tableau, τ_1. Since A_1 is satisfiable, τ_1 has at least one open path. Append A_2 to each of the open paths of τ_1, and generate a finite replete tableau, τ_2. Since $\{A_1, A_2\}$ is satisfiable, τ_2 has at least one open path. Append A_3 to each of the open paths of τ_2, and generate a finite replete tableau, τ_3. .... Put $\tau = \bigcup_{i=1}^{\infty} \tau_i$. Thus τ is a replete tableau. Note also that τ is an infinite, finitely branching tree. By König's Lemma (Theorem 1.6.6), let S' be an infinite path in τ. Then S' is a Hintikka set containing S. By Hintikka's Lemma, S' is satisfiable. Hence S is satisfiable. □

Theorem 1.7.2 (the Compactness Theorem, uncountable case). Let S be an uncountable set of propositional formulas. If each finite subset of S is satisfiable, the S is satisfiable.

Proof. We present three proofs. The first uses Zorn's Lemma. The second uses transfinite induction. The third uses Tychonoff's Theorem.

Let L be the (necessarily uncountable) propositional language consisting of all atoms occurring in formulas of S. If S is a set of L-formulas, we say that S is *finitely satisfiable* if each finite subset of S is satisfiable. We are trying to prove that, if S is finitely satisfiable, then S is satisfiable.

First proof. Consider the partial ordering $\mathfrak{F}$ of all finitely satisfiable sets of L-formulas which include S, ordered by inclusion. It is easy to see that any chain in $\mathfrak{F}$ has a least upper bound in $\mathfrak{F}$. Hence, by Zorn's Lemma, $\mathfrak{F}$ has a maximal element, S^*. Thus S^* is a set of L-formulas, $S^* \supseteq S$, S^* is finitely satisfiable, and for each L-formula $A \notin S^*$, $S^* \cup \{A\}$ is not finitely satisfiable. From this it is straightforward to verify that S^* is a Hintikka set. Hence, by Hintikka's Lemma, S^* is satisfiable. Hence S is satisfiable.

Second proof. Let A_ξ, $\xi < \alpha$, be a transfinite enumeration of all L-formulas. By transfinite recursion, put $S_0 = S$, $S_{\xi+1} = S_\xi \cup \{A_\xi\}$ if $S_\xi \cup \{A_\xi\}$ is finitely satisfiable, $S_{\xi+1} = S_\xi$ otherwise, and $S_\eta = \bigcup_{\xi<\eta} S_\xi$ for limit ordinals $\eta \leq \alpha$. Using transfinite induction, it is easy to verify that S_ξ is finitely satisfiable for

each $\xi \leq \alpha$. In particular, S_α is finitely satisfiable. It is straightforward to verify that S_α is a Hintikka set. Hence, by Hintikka's Lemma, S_α is satisfiable. Hence S is satisfiable.

Third proof. Let $\mathfrak{M} = \{\mathrm{T}, \mathrm{F}\}^L$ be the space of all L-assignments $M : L \to \{\mathrm{T}, \mathrm{F}\}$. Make $\mathfrak{M}$ a topological space with the product topology where $\{\mathrm{T}, \mathrm{F}\}$ has the discrete topology. Since $\{\mathrm{T}, \mathrm{F}\}$ is compact, it follows by Tychonoff's Theorem that $\mathfrak{M}$ is compact. For each L-formula A, put $\mathfrak{M}_A = \{M \in \mathfrak{M} : v_M(A) = \mathrm{T}\}$. It is easy to check that each $\mathfrak{M}_A$ is a topologically closed set in $\mathfrak{M}$. If S is finitely satisfiable, then the family of sets $\mathfrak{M}_A$, $A \in S$ has the finite intersection property, i.e., $\bigcap_{A \in S_0} \mathfrak{M}_A \neq \emptyset$ for each finite $S_0 \subseteq S$. By compactness of $\mathfrak{M}$ it follows that $\bigcap_{A \in S} \mathfrak{M}_A \neq \emptyset$. Thus S is satisfiable. □

1.8 Combinatorial Applications

In this section we present some combinatorial applications of the Compactness Theorem for propositional calculus.

Definition 1.8.1.

1. A *graph* consists of a set of *vertices* together with a specification of certain pairs of vertices as being *adjacent*. We require that a vertex may not be adjacent to itself, and that u is adjacent to v if and only if v is adjacent to u.

2. Let G be a graph and let k be a positive integer. A *k-coloring* of G is a function $f : \{\text{vertices of } G\} \to \{c_1, \ldots, c_k\}$ such that $f(u) \neq f(v)$ for all adjacent pairs of vertices u, v.

3. G is said to be *k-colorable* if there exists a k-coloring of G. This notion is much studied in graph theory.

Exercise 1.8.2. Let G be a graph and let k be a positive integer. For each vertex v and each $i = 1, \ldots, k$, let p_{vi} be a propositional atom expressing that vertex v receives color c_i. Define $C_k(G)$ to be the following set of propositional formulas: $p_{v1} \vee \cdots \vee p_{vk}$ for each vertex v; $\neg(p_{vi} \,\&\, p_{vj})$ for each vertex v and $1 \leq i < j \leq k$; $\neg(p_{ui} \,\&\, p_{vi})$ for each adjacent pair of vertices u, v and $1 \leq i \leq k$.

1. Show that there is a one-to-one correspondence between k-colorings of G and assignments satisfying $C_k(G)$.

2. Show that G is k-colorable if and only if $C_k(G)$ is satisfiable.

3. Show that G is k-colorable if and only if each finite subgraph of G is k-colorable.

Definition 1.8.3. A *partial ordering* consists of a set P together with a binary relation $\leq_P$ such that

1. $a \leq_P a$ for all $a \in P$ (reflexivity);

2. $a \leq_P b$, $b \leq_P c$ imply $a \leq_P c$ (transitivity);

3. $a \leq_P b$, $b \leq_P a$ imply $a = b$ (antisymmetry).

Example 1.8.4. Let $P = \mathbb{N}^+ = \{1, 2, 3, \ldots, n, \ldots\}$ = the set of positive integers.

1. Let $\leq_P$ be the usual order relation on P, i.e., $m \leq_P n$ if and only if $m \leq n$.

2. Let $\leq_P$ be the divisibility ordering of P, i.e., $m \leq_P n$ if and only if m is a divisor of n.

Definition 1.8.5. Let $P, \leq_P$ be a partial ordering.

1. Two elements $a, b \in P$ are *comparable* if either $a \leq_P b$ or $b \leq_P a$. Otherwise they are *incomparable*.

2. A *chain* is a set $X \subseteq P$ such that any two elements of X are comparable.

3. An *antichain* is a set $X \subseteq P$ such that any two distinct elements of X are incomparable.

Exercise 1.8.6. Let $P, \leq_P$ be a partial ordering, and let k be a positive integer.

1. Use the Compactness Theorem to show that P is the union of k chains if and only if each finite subset of P is the union of k chains.

2. *Dilworth's Theorem* says that P is the union of k chains if and only if every antichain is of size $\leq k$. Show that Dilworth's Theorem for arbitrary partial orderings follows from Dilworth's Theorem for finite partial orderings.

Chapter 2

Predicate Calculus

2.1 Formulas and Sentences

Definition 2.1.1 (languages). A *language* L is a set of *predicates*, each predicate P of L being designated as *n-ary* for some nonnegative[1] integer n.

Definition 2.1.2 (variables and quantifiers). We assume the existence of a fixed, countably infinite set of symbols $x, y, z, \ldots$ known as *variables*. We introduce two new symbols: the *universal quantifier* ($\forall$) and the *existential quantifier* ($\exists$). They are read as "for all" and "there exists", respectively.

Definition 2.1.3 (formulas). Let L be a language, and let U be a set. The set of *L-U-formulas* is generated as follows.

1. An *atomic* L-U-formula is an expression of the form $Pe_1 \cdots e_n$ where P is an n-ary predicate of L and each of $e_1, \ldots, e_n$ is either a variable or an element of U.
2. Each atomic L-U-formula is an L-U-formula.
3. If A is an L-U-formula, then $\neg A$ is an L-U-formula.
4. If A and B are L-U-formulas, then $A \,\&\, B$, $A \vee B$, $A \Rightarrow B$, $A \Leftrightarrow B$ are L-U-formulas.
5. If x is a variable and A is an L-U-formula, then $\forall x\, A$ and $\exists x\, A$ are L-U-formulas.

Definition 2.1.4 (degree). The *degree* of a formula is the number of occurrences of propositional connectives $\neg$, $\&$, $\vee$, $\Rightarrow$, $\Leftrightarrow$ and quantifiers $\forall, \exists$ in it.

Definition 2.1.5. An *L-formula* is an L-$\emptyset$-formula, i.e., an L-U-formula where $U = \emptyset$, the empty set.

[1] It will be seen that 0-ary predicates behave as propositional atoms. Thus predicate calculus is an extension of propositional calculus.

Remark 2.1.6. If U is a subset of U', then every L-U-formula is automatically an L-U'-formula. In particular, every L-formula is automatically an L-U-formula, for any set U.

Definition 2.1.7. In situations where the language L is understood from context, an L-U-formula may be called a *U-formula*, and an L-formula a *formula*.

Definition 2.1.8 (substitution). If A is an L-U-formula and x is a variable and $a \in U$, we define an L-U-formula $A[x/a]$ as follows.

1. If A is atomic, then $A[x/a]$ = the result of replacing each occurrence of x in A by a.
2. $(\neg A)[x/a] = \neg A[x/a]$.
3. $(A \,\&\, B)[x/a] = A[x/a] \,\&\, B[x/a]$.
4. $(A \vee B)[x/a] = A[x/a] \vee B[x/a]$.
5. $(A \Rightarrow B)[x/a] = A[x/a] \Rightarrow B[x/a]$.
6. $(A \Leftrightarrow B)[x/a] = A[x/a] \Leftrightarrow B[x/a]$.
7. $(\forall x\, A)[x/a] = \forall x\, A$.
8. $(\exists x\, A)[x/a] = \exists x\, A$.
9. If y is a variable other than x, then $(\forall y\, A)[x/a] = \forall y\, A[x/a]$.
10. If y is a variable other than x, then $(\exists y\, A)[x/a] = \exists y\, A[x/a]$.

Definition 2.1.9 (free variables). An occurrence of a variable x in an L-U-formula A is said to be *bound in A* if it is within the scope of a quantifier $\forall x$ or $\exists x$ in A. An occurrence of a variable x in an L-U-formula A is said to be *free in A* if it is not bound in A. A variable x is said to *occur freely in A* if there is at least one occurrence of x in A which is free in A.

Exercise 2.1.10.

1. Show that $A[x/a]$ is the result of substituting a for all free occurrences of x in A.
2. Show that x occurs freely in A if and only if $A[x/a] \neq A$.

Definition 2.1.11 (sentences). An *L-U-sentence* is an L-U-formula in which no variables occur freely. An *L-sentence* is an L-$\emptyset$-sentence, i.e., an L-U-sentence where $U = \emptyset$, the empty set.

Remark 2.1.12. If U is a subset of U', then every L-U-sentence is automatically an L-U'-sentence. In particular, every L-sentence is automatically an L-U-sentence, for any set U.

Definition 2.1.13. In situations where the language L is understood from context, an L-U-sentence may be called a *U-sentence*, and an L-sentence a *sentence*.

2.2 Structures and Satisfiability

Definition 2.2.1. Let U be a nonempty set, and let n be a nonnegative[2] integer. U^n is the set of all n-tuples of elements of U, i.e.,

$$U^n = \{\langle a_1, \ldots, a_n\rangle : a_1, \ldots, a_n \in U\} .$$

An *n-ary relation on U* is a subset of U^n.

Definition 2.2.2. Let L be a language. An *L-structure* M consists of a nonempty set U_M, called the *domain* or *universe* of M, together with an n-ary relation P_M on U_M for each n-ary predicate P of L. An L-structure may be called a *structure*, in situations where the language L is understood from context.

Definition 2.2.3. Two L-structures M and M' are said to be *isomorphic* if there exists an *isomorphism* of M onto M', i.e., a one-to-one correspondence $\phi : U_M \cong U_{M'}$ such that for all n-ary predicates P of L and all n-tuples $\langle a_1, \ldots, a_n\rangle \in (U_M)^n$, $\langle a_1, \ldots, a_n\rangle \in P_M$ if and only if $\langle \phi(a_1), \ldots, \phi(a_n)\rangle \in P_{M'}$.

As usual in abstract mathematics, we are mainly interested in properties of structures that are invariant under isomorphism.

Lemma 2.2.4. Given an L-structure M, there is a unique *valuation* or assignment of truth values

$$v_M : \{A : A \text{ is an } L\text{-}U_M\text{-sentence}\} \to \{\mathrm{T}, \mathrm{F}\}$$

defined as follows:

1. $v_M(Pa_1 \cdots a_n) = \begin{cases} \mathrm{T} & \text{if } \langle a_1, \ldots, a_n\rangle \in P_M , \\ \mathrm{F} & \text{if } \langle a_1, \ldots, a_n\rangle \notin P_M . \end{cases}$

2. $v_M(\neg A) = \begin{cases} \mathrm{T} & \text{if } v_M(A) = \mathrm{F} , \\ \mathrm{F} & \text{if } v_M(A) = \mathrm{T} . \end{cases}$

3. $v_M(A \,\&\, B) = \begin{cases} \mathrm{T} & \text{if } v_M(A) = v_M(B) = \mathrm{T} , \\ \mathrm{F} & \text{if at least one of } v_M(A), v_M(B) = \mathrm{F} . \end{cases}$

4. $v_M(A \vee B) = \begin{cases} \mathrm{T} & \text{if at least one of } v_M(A), v_M(B) = \mathrm{T} , \\ \mathrm{F} & \text{if } v_M(A) = v_M(B) = \mathrm{F} . \end{cases}$

5. $v_M(A \Rightarrow B) = v_M(\neg (A \,\&\, \neg B))$.

6. $v_M(A \Leftrightarrow B) = \begin{cases} \mathrm{T} & \text{if } v_M(A) = v_M(B) , \\ \mathrm{F} & \text{if } v_M(A) \neq v_M(B) . \end{cases}$

[2] In the special case $n = 0$ we obtain the notion of a 0-ary relation, i.e., a subset of $\{\langle\rangle\}$. There are only two 0-ary relations, $\{\langle\rangle\}$ and $\{\}$, corresponding to T and F respectively. Thus a 0-ary predicate behaves as a propositional atom.

7. $v_M(\forall x\, A) = \begin{cases} \mathrm{T} & \text{if } v_M(A[x/a]) = \mathrm{T} \text{ for all } a \in U_M\,, \\ \mathrm{F} & \text{if } v_M(A[x/a]) = \mathrm{F} \text{ for at least one } a \in U_M\,. \end{cases}$

8. $v_M(\exists x\, A) = \begin{cases} \mathrm{T} & \text{if } v_M(A[x/a]) = \mathrm{T} \text{ for at least one } a \in U_M\,, \\ \mathrm{F} & \text{if } v_M(A[x/a]) = \mathrm{F} \text{ for all } a \in U_M\,. \end{cases}$

Proof. The truth value $v_M(A)$ is defined by recursion on L-U_M-sentences, i.e., by induction on the degree of A where A is an arbitrary L-U_M-sentence. □

Definition 2.2.5 (truth and satisfaction). Let M be an L-structure.

1. Let A be an L-U_M-sentence. We say that A is *true in* M if $v_M(A) = \mathrm{T}$. We say that A is *false in* M if $v_M(A) = \mathrm{F}$.

2. Let S be a set of L-U_M-sentences. We say that M *satisfies* S if all of the sentences of S are true in M.

Theorem 2.2.6.

1. If M and M' are isomorphic L-structures and $\phi : M \cong M'$ is an isomorphism of M onto M', then for all L-U_M-sentences A we have $v_M(A) = v_{M'}(A')$ where $A' = A[a_1/\phi(a_1), \ldots, a_k/\phi(a_k)]$.[3] Here $a_1, \ldots, a_k$ are the elements of U_M which occur in A.

2. If M and M' are isomorphic L-structures, then they satisfy the same L-sentences.

Proof. We omit the proof of part 1. A more general result will be proved later as Theorem 2.7.3. Part 2 follows immediately from part 1. □

Definition 2.2.7 (satisfiability). Let S be a set of L-sentences. S is said to be *satisfiable*[4] if there exists an L-structure M which satisfies S.

Remark 2.2.8. Satisfiability is one of the most important concepts of mathematical logic. A key result known as the Compactness Theorem[5] states that a set S of L-sentences is satisfiable if and only every finite subset of S is satisfiable.

The following related notion is of technical importance.

Definition 2.2.9 (satisfiability in a domain). Let U be a nonempty set. A set S of L-U-sentences is said to be *satisfiable in the domain* U if there exists an L-structure M such that M satisfies S and $U_M = U$.

Remark 2.2.10. Let S be a set of L-sentences. Then S is satisfiable (according to Definition 2.2.7) if and only if S is satisfiable in some domain U.

[3] We have extended the substitution notation 2.1.8 in an obvious way.

[4] Similarly, the notions of logical validity and logical consequence are defined for L-sentences, in the obvious way, using L-structures. An L-sentence is said to be *logically valid* if it is satisfied by all L-structures. An L-sentence is said to be a *logical consequence* of S if it is satisfied by all L-structures satisfying S.

[5] See Theorems 2.6.1 and 2.6.2 below.

Theorem 2.2.11. Let S be a set of L-sentences. If S is satisfiable in a domain U, then S is satisfiable in any domain of the same cardinality as U.

Proof. Suppose S is satisfiable in a domain U. Let M be an L-structure M satisfying S with $U_M = U$. Let U' be any set of the same cardinality as U. Then there exists a one-to-one correspondence $\phi : U \to U'$. Let M' be the L-structure with $U_{M'} = U'$, $P_{M'} = \{\langle \phi(a_1), \ldots, \phi(a_n)\rangle : \langle a_1, \ldots, a_n\rangle \in P_M\}$ for all n-ary predicates P of L. Then M is isomorphic to M'. Hence, by Theorem 2.2.6, M' satisfies S. Thus S is satisfiable in the domain U'. □

Example 2.2.12. We exhibit a sentence A_∞ which is satisfiable in an infinite domain but not in any finite domain. Our sentence A_∞ is (1) & (2) & (3) with

$$(1) \quad \forall x\, \forall y\, \forall z\, ((Rxy \,\&\, Ryz) \Rightarrow Ryz)$$
$$(2) \quad \forall x\, \forall y\, (Rxy \Rightarrow \neg\, Ryx)$$
$$(3) \quad \forall x\, \exists y\, Rxy$$

See also Example 2.5.9.

2.3 The Tableau Method

Definition 2.3.1. Fix a countably infinite set $V = \{a_1, a_2, \ldots, a_n, \ldots\}$. The elements of V will be called *parameters*. If L is a language, L-V-sentences will be called *sentences with parameters*.

Definition 2.3.2. A (*signed* or *unsigned*) *tableau* is a rooted dyadic tree where each node carries a (signed or unsigned) L-V-sentence. The tableau rules for predicate calculus are the same as those for propositional calculus, with the following additional rules.

Signed:

$$\begin{array}{cc} \vdots & \vdots \\ \mathrm{T}\,\forall x\, A & \mathrm{F}\,\exists x\, A \\ \vdots & \vdots \\ | & | \\ \mathrm{T}\, A[x/a] & \mathrm{F}\, A[x/a] \end{array}$$

where a is an arbitrary parameter

$$\begin{array}{cc} \vdots & \vdots \\ \mathrm{T}\,\exists x\,A & \mathrm{F}\,\forall x\,A \\ \vdots & \vdots \\ | & | \\ \mathrm{T}\,A[x/a] & \mathrm{F}\,A[x/a] \end{array}$$

where a is a new parameter

Unsigned:

$$\begin{array}{cc} \vdots & \vdots \\ \forall x\,A & \neg\,\exists x\,A \\ \vdots & \vdots \\ | & | \\ A[x/a] & \neg\,A[x/a] \end{array}$$

where a is an arbitrary parameter

$$\begin{array}{cc} \vdots & \vdots \\ \exists x\,A & \neg\,\forall x\,A \\ \vdots & \vdots \\ | & | \\ A[x/a] & \neg\,A[x/a] \end{array}$$

where a is a new parameter

Remark 2.3.3. In the above tableau rules, "a is new" means that a does not occur in the path that is being extended. Or, we can insist that a not occur in the tableau that is being extended.

Remark 2.3.4. We are going to prove that the tableau method for predicate calculus is sound (Theorem 2.3.9) and complete (Theorem 2.5.5). In particular, a sentence A of the predicate calculus is logically valid if and only if there exists a finite closed signed tableau starting with $\mathrm{F}\,A$, or equivalently a finite closed unsigned tableau starting with $\neg\,A$.

Example 2.3.5. The signed tableau

$$\begin{array}{c} \mathrm{F}\,(\exists x\,\forall y\,Rxy) \Rightarrow (\forall y\,\exists x\,Rxy) \\ \mathrm{T}\,\exists x\,\forall y\,Rxy \\ \mathrm{F}\,\forall y\,\exists x\,Rxy \\ \mathrm{T}\,\forall y\,Ray \\ \mathrm{F}\,\exists x\,Rxb \\ \mathrm{T}\,Rab \\ \mathrm{F}\,Rab \end{array}$$

is closed. Therefore, by the Soundness Theorem, $(\exists x\,\forall y\,Rxy) \Rightarrow (\forall y\,\exists x\,Rxy)$ is logically valid.

Example 2.3.6. The unsigned tableau

$$\begin{array}{c} \neg\,(\exists x\,(Px \vee Qx)) \Leftrightarrow ((\exists x\,Px) \vee (\exists x\,Qx)) \\ \diagup \qquad\qquad \diagdown \\ \exists x\,(Px \vee Qx) \qquad\qquad \neg\,\exists x\,(Px \vee Qx) \\ \neg\,((\exists x\,Px) \vee (\exists x\,Qx)) \qquad\qquad (\exists x\,Px) \vee (\exists x\,Qx) \\ \neg\,\exists x\,Px \qquad\qquad\qquad\qquad \diagup \qquad \diagdown \\ \neg\,\exists x\,Qx \qquad\qquad\qquad \exists x\,Px \qquad\qquad \exists x\,Qx \\ Pa \vee Qa \qquad\qquad\qquad\quad Pa \qquad\qquad\quad Qa \\ \diagup \quad \diagdown \qquad\qquad\qquad \neg\,(Pa \vee Qa) \qquad \neg\,(Pa \vee Qa) \\ Pa \qquad Qa \qquad\qquad\qquad \neg\,Pa \qquad\qquad \neg\,Pa \\ \neg\,Pa \qquad \neg\,Qa \qquad\qquad\qquad \neg\,Qa \qquad\qquad \neg\,Qa \end{array}$$

is closed. Therefore, by the Soundness Theorem,

$$(\exists x\,(Px \vee Qx)) \Leftrightarrow ((\exists x\,Px) \vee (\exists x\,Qx))$$

is logically valid.

The rest of this section is devoted to proving the Soundness Theorem 2.3.9.

Definition 2.3.7.

1. An *L-V-structure* consists of an L-structure M together with a mapping $\phi : V \to U_M$. If A is an L-V-sentence, we write

   $$A^\phi = A[a_1/\phi(a_1), \ldots, a_k/\phi(a_k)]$$

   where $a_1, \ldots, a_k$ are the parameters occurring in A. Note that A^ϕ is an L-U_M-sentence. Note also that, if A is an L-sentence, then $A^\phi = A$.

2. Let S be a finite or countable set of (signed or unsigned) L-V-sentences. An L-V-structure M, f is said to *satisfy* S if $v_M(A^\phi) = \mathrm{T}$ for all $A \in S$. S is said to be *satisfiable*[6] if there exists an L-V-structure satisfying S. Note that this definition is compatible with Definition 2.2.7.

[6]Similarly, the notions of logical validity and logical consequence are extended to L-V-sentences, in the obvious way, using L-V-structures. An L-V-sentences is said to be *logically valid* if it satisfied by all L-V-structures. An L-V-sentence is said to be a *logical consequence* of S if it is satisfied by all L-V-structures satisfying S.

3. Let τ be an L-tableau. We say that τ is *satisfiable* if at least one path of τ is satisfiable.

Lemma 2.3.8. Let τ and τ' be tableaux such that τ' is an *immediate extension* of τ, i.e., τ' is obtained from τ by applying a tableau rule to a path of τ. If τ is satisfiable, then τ' is satisfiable.

Proof. The proof consists of one case for each tableau rule. We consider some representative cases.

Case 1. Suppose that τ' is obtained from τ by applying the rule

$$\begin{array}{c} \vdots \\ A \vee B \\ \vdots \\ / \quad \backslash \\ A \quad B \end{array}$$

to the path θ in τ, where a is a parameter. Since τ is satisfiable, it has at least one satisfiable path, S. If $S \neq \theta$, then S is a path of τ', so τ' is satisfiable. If $S = \theta$, then θ is satisfiable, so let M and $\phi : V \to U_M$ satisfy θ. In particular $v_M((A \vee B)^\phi) = \mathrm{T}$, so we have at least one of $v_M(A^\phi) = \mathrm{T}$ and $v_M(B^\phi) = \mathrm{T}$. Thus M and f satisfy at least one of θ, A and θ, B. Since these are paths of τ', it follows that τ' is satisfiable.

Case 2. Suppose that τ' is obtained from τ by applying the rule

$$\begin{array}{c} \vdots \\ \forall x\, A \\ \vdots \\ | \\ A[x/a] \end{array}$$

to the path θ in τ. Since τ is satisfiable, it has at least one satisfiable path, S. If $S \neq \theta$, then S is a path of τ', so τ' is satisfiable. If $S = \theta$, then θ is satisfiable, so let M and $\phi : V \to U_M$ satisfy θ. In particular $v_M(\forall x\, (A^\phi)) = v_M((\forall x\, A)^\phi) = \mathrm{T}$, so $v_M(A^\phi[x/c]) = \mathrm{T}$ for all $c \in U_M$. In particular

$$v_M(A[x/a]^\phi) = v_M(A^\phi[x/\phi(a)]) = \mathrm{T}.$$

Thus M and ϕ satisfy $\theta, A[x/a]$. Since this is a path of τ', it follows that τ' is satisfiable.

Case 3. Suppose that τ' is obtained from τ by applying the rule

$$\begin{array}{c} \vdots \\ \exists x\, A \\ \vdots \\ | \\ A[x/a] \end{array}$$

to the path θ in τ, where a is a new parameter. Since τ is satisfiable, it has at least one satisfiable path, S. If $S \neq \theta$, then S is a path of τ', so τ' is satisfiable. If $S = \theta$, then θ is satisfiable, so let M and $\phi : V \to U_M$ satisfy θ. In particular $v_M(\exists x\,(A^\phi)) = v_M((\exists x\,A)^\phi) = \mathrm{T}$, so $v_M(A^\phi[x/c]) = \mathrm{T}$ for at least one $c \in U_M$. Fix such a c and define $\phi' : V \to U_M$ by putting $\phi'(a) = c$, and $\phi'(b) = \phi(b)$ for all $b \neq a$, $b \in V$. Since a is new, we have $B^{\phi'} = B^\phi$ for all $B \in \theta$, and $A^{\phi'} = A^\phi$, hence $A[x/a]^{\phi'} = A^{\phi'}[x/\phi'(a)] = A^\phi[x/c]$. Thus $v_M(B^{\phi'}) = v_M(B^\phi) = \mathrm{T}$ for all $B \in \theta$, and $v_M(A[x/a]^{\phi'}) = v_M(A^\phi[x/c]) = \mathrm{T}$. Thus M and ϕ' satisfy $\theta, A[x/a]$. Since this is a path of τ', it follows that τ' is satisfiable. □

Theorem 2.3.9 (the Soundness Theorem). Let $X_1, \ldots, X_k$ be a finite set of (signed or unsigned) sentences with parameters. If there exists a finite closed tableau starting with $X_1, \ldots, X_k$, then $X_1, \ldots, X_k$ is not satisfiable.

Proof. Let τ be a closed tableau starting with $X_1, \ldots, X_k$. Thus there is a finite sequence of tableaux $\tau_0, \tau_1, \ldots, \tau_n = \tau$ such that

$$\tau_0 = \begin{array}{c} X_1 \\ \vdots \\ X_k \end{array}$$

and each τ_{i+1} is an immediate extension of τ_i. Suppose $X_1, \ldots, X_k$ is satisfiable. Then τ_0 is satisfiable, and by induction on i using Lemma 2.3.8, it follows that all of the τ_i are satisfiable. In particular $\tau_n = \tau$ is satisfiable, but this is impossible since τ is closed. □

2.4 Logical Equivalence

Definition 2.4.1. Given a formula A, let $A' = A[x_1/a_1, \ldots, x_k/a_k]$, where $x_1, \ldots, x_k$ are the variables which occur freely in A, and $a_1, \ldots, a_k$ are parameters not occurring in A. Note that A' has no free variables, i.e., it is a sentence. We define A to be *satisfiable* if and only if A' is satisfiable, in the sense of Definition 2.3.7. We define A to be *logically valid* if and only if A' is logically valid, in the sense of Definition 2.3.7.

Exercises 2.4.2. Let A be a formula.

1. Show that A is logically valid if and only if $\neg\, A$ is not satisfiable. Show that A is satisfiable if and only if $\neg\, A$ is not logically valid.

2. Let x be a variable. Show that A is logically valid if and only if $\forall x\, A$ is logically valid. Show that A is satisfiable if and only if $\exists x\, A$ is satisfiable.

3. Let x be a variable, and let a be a parameter not occurring in A. Show that A is logically valid if and only if $A[x/a]$ is logically valid. Show that A is satisfiable if and only if $A[x/a]$ is satisfiable.

Definition 2.4.3. Let A and B be formulas. A and B are said to be *logically equivalent*, written $A \equiv B$, if $A \Leftrightarrow B$ is logically valid.

Exercise 2.4.4. Assume $A \equiv B$. Show that for any variable x, $\forall x\, A \equiv \forall x\, B$ and $\exists x\, A \equiv \exists x\, B$. Show that for any variable x and parameter a, $A[x/a] \equiv B[x/a]$.

Exercise 2.4.5. For a formula A, it is not in general true that $A \equiv A'$, where A' is as in Definition 2.4.1. Also, it is not in general true that $A \equiv \forall x\, A$, or that $A \equiv \exists x\, A$, or that $A \equiv A[x/a]$. Give examples illustrating these remarks.

Exercise 2.4.6. If A and B are formulas, put $A' = A[x_1/a_1, \ldots, x_k/a_k]$ and $B' = B[x_1/a_1, \ldots, x_k/a_k]$, where $x_1, \ldots, x_k$ are the variables occurring freely in A and B, and $a_1, \ldots, a_k$ are parameters not occurring in A or in B. Show that $A \equiv B$ if only if $A' \equiv B'$.

Remark 2.4.7. The results of Exercises 1.3.3 and 1.3.4 and Remark 1.3.5 for formulas of the propositional calculus, also hold for formulas of the predicate calculus.

In particular, if $A_1 \equiv A_2$, then for any formula C containing A_1 as a part, if we replace one or more occurrences of the part A_1 by A_2, then the resulting formula is logically equivalent to C.

Remark 2.4.8. Some useful logical equivalences are:

1. (a) $\forall x\, A \equiv A$, provided x does not occur freely in A
   (b) $\exists x\, A \equiv A$, provided x does not occur freely in A
   (c) $\forall x\, A \equiv \forall y\, A[x/y]$, provided y does not occur in A
   (d) $\exists x\, A \equiv \exists y\, A[x/y]$, provided y does not occur in A
2. (a) $\forall x\, (A \,\&\, B) \equiv (\forall x\, A) \,\&\, (\forall x\, B)$
   (b) $\exists x\, (A \vee B) \equiv (\exists x\, A) \vee (\exists x\, B)$
   (c) $\exists x\, (A \Rightarrow B) \equiv (\forall x\, A) \Rightarrow (\exists x\, B)$

   Note however that, in general, $\exists x\, (A \,\&\, B) \not\equiv (\exists x\, A) \,\&\, (\exists x\, B)$, and $\forall x\, (A \vee B) \not\equiv (\forall x\, A) \vee (\forall x\, B)$, and $\forall x\, (A \Rightarrow B) \not\equiv (\exists x\, A) \Rightarrow (\forall x\, B)$.

   On the other hand, we have:

3. (a) $\exists x\, (A \,\&\, B) \equiv A \,\&\, (\exists x\, B)$, provided x does not occur freely in A
   (b) $\exists x\, (A \,\&\, B) \equiv (\exists x\, A) \,\&\, B$, provided x does not occur freely in B
   (c) $\forall x\, (A \vee B) \equiv A \vee (\forall x\, B)$, provided x does not occur freely in A
   (d) $\forall x\, (A \vee B) \equiv (\forall x\, A) \vee B$, provided x does not occur freely in B
   (e) $\forall x\, (A \Rightarrow B) \equiv A \Rightarrow (\forall x\, B)$, provided x does not occur freely in A
   (f) $\forall x\, (A \Rightarrow B) \equiv (\exists x\, A) \Rightarrow B$, provided x does not occur freely in B
4. (a) $\exists x \neg A \equiv \neg \forall x\, A$

(b) $\forall x \neg A \equiv \neg \exists x\, A$

(c) $\forall x\, A \equiv \neg \exists x \neg A$

(d) $\exists x\, A \equiv \neg \forall x \neg A$

Definition 2.4.9 (prenex form). A formula is said to be *quantifier-free* if it contains no quantifiers. A formula is said to be *in prenex form* if it is of the form $Q_1 x_1 \cdots Q_n x_n\, B$, where each Q_i is a quantifier ($\forall$ or $\exists$), each x_i is a variable, and B is quantifier-free.

Example 2.4.10. The sentence

$$\forall x\, \forall y\, \exists z\, \forall w\, (Rxy \Rightarrow (Rxz \,\&\, Rzy \,\&\, \neg\, (Rzw \,\&\, Rwy)))$$

is in prenex form.

Exercise 2.4.11. Show that every formula is logically equivalent to a formula in prenex form.

Example 2.4.12. Consider the sentence $(\exists x\, Px) \,\&\, (\exists x\, Qx)$. We wish to put this into prenex form. Applying the equivalences of Remark 2.4.8, we have

$$\begin{aligned} (\exists x\, Px) \,\&\, (\exists x\, Qx) &\equiv (\exists x\, Px) \,\&\, (\exists y\, Qy) \\ &\equiv \exists x\, (Px \,\&\, (\exists y\, Qy)) \\ &\equiv \exists x\, \exists y\, (Px \,\&\, Qy) \end{aligned}$$

and this is in prenex form.

Exercises 2.4.13. Let A and B be quantifier-free formulas. Put the following into prenex form.

1. $(\exists x\, A) \,\&\, (\exists x\, B)$
2. $(\forall x\, A) \Leftrightarrow (\forall x\, B)$
3. $(\forall x\, A) \Leftrightarrow (\exists x\, B)$

Definition 2.4.14 (universal closure). Let A be a formula. The *universal closure* of A is the sentence $A^* = \forall x_1 \cdots \forall x_k\, A$, where $x_1, \ldots, x_k$ are the variables which occur freely in A. Note that $A^{**} = A^*$.

Exercises 2.4.15. Let A be a formula.

1. Show that A is logically valid if and only if A^*, the universal closure of A, is logically valid.
2. It is not true in general that $A \equiv A^*$. Give an example illustrating this.
3. It is not true in general that A is satisfiable if and only if A^* is satisfiable. Give an example illustrating this.
4. For formulas A and B, it is not true in general that $A \equiv B$ if and only if $A^* \equiv B^*$. Give an example illustrating this.

For completeness we state the following definition.

Definition 2.4.16. Let $A_1, \ldots, A_k, B$ be formulas. We say that B is a *logical consequence* of $A_1, \ldots, A_k$ if $(A_1 \,\&\, \cdots \,\&\, A_k) \Rightarrow B$ is logically valid. This is equivalent to saying that the universal closure of $(A_1 \,\&\, \cdots \,\&\, A_k) \Rightarrow B$ is logically valid.

Remark 2.4.17. A and B are logically equivalent if and only if each is a logical consequence of the other. A is logically valid if and only if A is a logical consequence of the empty set. $\exists x\, A$ is a logical consequence of $A[x/a]$, but the converse does not hold in general. $A[x/a]$ is a logical consequence of $\forall x\, A$, but the converse does not hold in general.

2.5 The Completeness Theorem

Let U be a nonempty set, and let S be a set of (signed or unsigned) L-U-sentences.

Definition 2.5.1. S is *closed* if S contains a conjugate pair of L-U-sentences. In other words, for some L-U-sentence A, S contains $\mathrm{T}\, A$, $\mathrm{F}\, A$ in the signed case, A, $\neg\, A$ in the unsigned case. S is *open* if it is not closed.

Definition 2.5.2. S is *U-replete* if S "obeys the tableau rules" with respect to U. We list some representative clauses of the definition.

1. If S contains $\mathrm{T} \neg\, A$, then S contains $\mathrm{F}\, A$. If S contains $\mathrm{F} \neg\, A$, then S contains $\mathrm{T}\, A$. If S contains $\neg \neg\, A$, then S contains A.

2. If S contains $\mathrm{T}\, A \,\&\, B$, then S contains both $\mathrm{T}\, A$ and $\mathrm{T}\, B$. If S contains $\mathrm{F}\, A \,\&\, B$, then S contains at least one of $\mathrm{F}\, A$ and $\mathrm{F}\, B$. If S contains $A \,\&\, B$, then S contains both A and B. If S contains $\neg\, (A \,\&\, B)$, then S contains at least one of $\neg\, A$ and $\neg\, B$.

3. If S contains $\mathrm{T}\, \exists x\, A$, then S contains $\mathrm{T}\, A[x/a]$ for at least one $a \in U$. If S contains $\mathrm{F}\, \exists x\, A$, then S contains $\mathrm{F}\, A[x/a]$ for all $a \in U$. If S contains $\exists x\, A$, then S contains $A[x/a]$ for at least one $a \in U$. If S contains $\neg\, \exists x\, A$, then S contains $\neg\, A[x/a]$ for all $a \in U$.

4. If S contains $\mathrm{T}\, \forall x\, A$, then S contains $\mathrm{T}\, A[x/a]$ for all $a \in U$. If S contains $\mathrm{F}\, \forall x\, A$, then S contains $\mathrm{F}\, A[x/a]$ for at least one $a \in U$. If S contains $\forall x\, A$, then S contains $A[x/a]$ for all $a \in U$. If S contains $\neg\, \forall x\, A$, then S contains $\neg\, A[x/a]$ for at least one $a \in U$.

Lemma 2.5.3 (Hintikka's Lemma). If S is U-replete and open[7], then S is satisfiable. In fact, S is satisfiable in the domain U.

[7] See also Exercise 2.5.7.

Proof. Assume S is U-replete and open. We define an L-structure M by putting $U_M = U$ and, for each n-ary predicate P of L,

$$P_M = \{(a_1, \ldots, a_n) \in U^n : \mathrm{T}\, Pa_1 \cdots a_n \in S\}$$

in the signed case, and

$$P_M = \{(a_1, \ldots, a_n) \in U^n : Pa_1 \cdots a_n \in S\}$$

in the unsigned case.

We claim that for all L-U-sentences A,

(a) if S contains $\mathrm{T}\, A$, then $v_M(A) = \mathrm{T}$

(b) if S contains $\mathrm{F}\, A$, then $v_M(A) = \mathrm{F}$

in the signed case, and

(c) if S contains A, then $v_M(A) = \mathrm{T}$

(d) if S contains $\neg A$, then $v_M(A) = \mathrm{F}$

in the unsigned case.

In both cases, the claim is easily proved by induction on the degree of A. We give the proof for some representative cases.

1. $\deg(A) = 0$. In this case A is atomic, say $A = Pa_1 \cdots a_n$.
   (a) If S contains $\mathrm{T}\, Pa_1 \cdots a_n$, then by definition of M we have $(a_1, \ldots, a_n) \in P_M$, so $v_M(Pa_1 \cdots a_n) = \mathrm{T}$.
   (b) If S contains $\mathrm{F}\, Pa_1 \cdots a_n$, then S does not contain $\mathrm{T}\, Pa_1 \cdots a_n$ since S is open. Thus by definition of M we have $(a_1, \ldots, a_n) \notin P_M$, so $v_M(Pa_1 \cdots a_n) = \mathrm{F}$.
   (c) If S contains $Pa_1 \cdots a_n$, then by definition of M we have $(a_1, \ldots, a_n) \in P_M$, so $v_M(Pa_1 \cdots a_n) = \mathrm{T}$.
   (d) If S contains $\neg\, Pa_1 \cdots a_n$, then S does not contain $Pa_1 \cdots a_n$ since S is open. Thus by definition of M we have $(a_1, \ldots, a_n) \notin P_M$, so $v_M(Pa_1 \cdots a_n) = \mathrm{F}$.
2. $\deg(A) > 0$ and $A = \neg\, B$. Note that $\deg(B) < \deg(A)$ so the inductive hypothesis applies to B.
3. $\deg(A) > 0$ and $A = B \,\&\, C$. Note that $\deg(B)$ and $\deg(C)$ are $< \deg(A)$ so the inductive hypothesis applies to B and C.
   (a) If S contains $\mathrm{T}\, B \,\&\, C$, then by repleteness of S we see that S contains both $\mathrm{T}B$ and $\mathrm{T}C$. Hence by inductive hypothesis we have $v_M(B) = v_M(C) = \mathrm{T}$. Hence $v_M(B \,\&\, C) = \mathrm{T}$.

(b) If S contains $\mathrm{F}\, B \,\&\, C$, then by repleteness of S we see that S contains at least one of $\mathrm{F}B$ and $\mathrm{F}C$. Hence by inductive hypothesis we have at least one of $v_M(B) = \mathrm{F}$ and $v_M(C) = \mathrm{F}$. Hence $v_M(B \,\&\, C) = \mathrm{F}$.

(c) If S contains $B \,\&\, C$, then by repleteness of S we see that S contains both B and C. Hence by inductive hypothesis we have $v_M(B) = v_M(C) = \mathrm{T}$. Hence $v_M(B \,\&\, C) = \mathrm{T}$.

(d) If S contains $\neg\,(B \,\&\, C)$, then by repleteness of S we see that S contains at least one of $\neg\, B$ and $\neg\, C$. Hence by inductive hypothesis we have at least one of $v_M(B) = \mathrm{F}$ and $v_M(C) = \mathrm{F}$. Hence $v_M(B \,\&\, C) = \mathrm{F}$.

4. $\deg(A) > 0$ and $A = \exists x\, B$. Note that for all $a \in U$ we have $\deg(B[x/a]) < \deg(A)$, so the inductive hypothesis applies to $B[x/a]$.

5. $\deg(A) > 0$ and $A = \forall x\, B$. Note that for all $a \in U$ we have $\deg(B[x/a]) < \deg(A)$, so the inductive hypothesis applies to $B[x/a]$.

□

We shall now use Hintikka's Lemma to prove the completeness of the tableau method. As in Section 2.3, Let $V = \{a_1, \ldots, a_n, \ldots\}$ be the set of parameters. Recall that a tableau is a tree whose nodes carry L-V-sentences.

Lemma 2.5.4. Let τ_0 be a finite tableau. By applying tableau rules, we can extend τ_0 to a (possibly infinite) tableau τ with the following properties: every closed path of τ is finite, and every open path of τ is V-replete.

Proof. The idea is to start with τ_0 and use tableau rules to construct a sequence of finite extensions $\tau_0, \tau_1, \ldots, \tau_i, \ldots$. If some τ_i is closed, then the construction halts, i.e., $\tau_j = \tau_i$ for all $j \geq i$, and we set $\tau = \tau_i$. In any case, we set $\tau = \tau_\infty = \bigcup_{i=0}^{\infty} \tau_i$. In the course of the construction, we apply tableau rules systematically to ensure that τ_∞ will have the desired properties, using the fact that $V = \{a_1, a_2, \ldots, a_n, \ldots\}$ is countably infinite.

Here are the details of the construction. Call a node X of τ_i *quasiuniversal* if it is of the form $\mathrm{T}\,\forall x\, A$ or $\mathrm{F}\,\exists x\, A$ or $\forall x\, A$ or $\neg\,\exists x\, A$. Our construction begins with τ_0. Suppose we have constructed τ_{2i}. For each quasiuniversal node X of τ_{2i} and each $n \leq 2i$, apply the appropriate tableau rule to extend each open path of τ_{2i} containing X by $\mathrm{T}\, A[x/a_n]$ or $\mathrm{F}\, A[x/a_n]$ or $A[x/a_n]$ or $\neg\, A[x/a_n]$ as the case may be. Let τ_{2i+1} be the finite tableau so obtained. Next, for each non-quasiuniversal node X of τ_{2i+1}, extend each open path containing X by applying the appropriate tableau rule. Again, let τ_{2i+2} be the finite tableau so obtained.

In this construction, a closed path is never extended, so all closed paths of τ_∞ are finite. In addition, the construction ensures that each open path of τ_∞ is V-replete. Thus τ_∞ has the desired properties. This proves our lemma. □

Theorem 2.5.5 (the Completeness Theorem). Let $X_1, \ldots, X_k$ be a finite set of (signed or unsigned) sentences with parameters. If $X_1, \ldots, X_k$ is not

satisfiable, then there exists a finite closed tableau starting with $X_1, \ldots, X_k$. If $X_1, \ldots, X_k$ is satisfiable, then $X_1, \ldots, X_k$ is satisfiable in the domain V.

Proof. By Lemma 2.5.4 there exists a (possibly infinite) tableau τ starting with $X_1, \ldots, X_k$ such that every closed path of τ is finite, and every open path of τ is V-replete. If τ is closed, then by König's Lemma (Theorem 1.6.6), τ is finite. If τ is open, let S be an open path of τ. Then S is V-replete. By Hintikka's Lemma 2.5.3, S is satisfiable in V. Hence $X_1, \ldots, X_k$ is satisfiable in V. □

Definition 2.5.6. Let L, U, and S be as in Definition 2.5.1. S is said to be *atomically closed* if S contains a conjugate pair of atomic L-U-sentences. In other words, for some n-ary L-predicate P and $a_1, \ldots, a_n \in U$, S contains $\mathrm{T}\, Pa_1 \cdots a_n$, $\mathrm{F}\, Pa_1 \cdots a_n$ in the signed case, and $Pa_1 \cdots a_n$, $\neg\, Pa_1 \cdots a_n$ in the unsigned case. S is *atomically open* if it is not atomically closed.

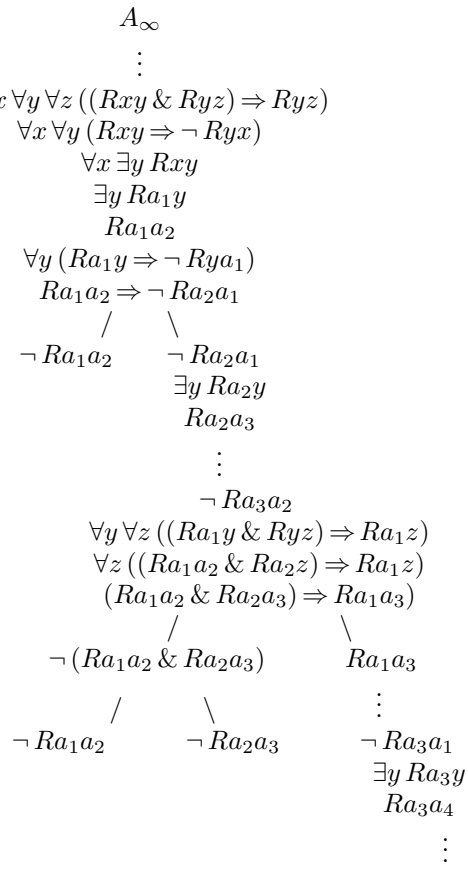
Exercise 2.5.7. Show that Lemmas 2.5.3 and 2.5.4 and Theorem 2.5.5 continue to hold with "closed" ("open") replaced by "atomically closed" ("atomically open").

Remark 2.5.8. Corollaries 1.5.9, 1.5.10, 1.5.11 carry over from the propositional calculus to the predicate calculus. In particular, the tableau method provides a test for logical validity of sentences of the predicate calculus.

Note however that the test is only partially effective. If a sentence A is logically valid, we will certainly find a finite closed tableau starting with $\neg\, A$. But if A is not logically valid, we will not necessarily find a finite tableau which demonstrates this. See the following example.

Example 2.5.9. In 2.2.12 we have seen an example of a sentence A_∞ which is satisfiable in a countably infinite domain but not in any finite domain. It is

instructive to generate a tableau starting with A_∞.



An infinite open path gives rise (via the proof of Hintikka's Lemma) to an infinite L-structure M with $U_M = \{a_1, a_2, \ldots, a_n, \ldots\}$, $R_M = \{\langle a_m, a_n\rangle : 1 \leq m < n\}$. Clearly M satisfies A_∞.

Remark 2.5.10. In the course of applying a tableau test, we will sometimes find a finite open path which is U-replete for some finite set of parameters $U \subseteq V$. In this case, the proof of Hintikka's Lemma provides a finite L-structure with domain U.

Example 2.5.11. Let A be the sentence $(\forall x\,(Px \vee Qx)) \Rightarrow ((\forall x\,Px) \vee (\forall x\,Qx))$.

Testing A for logical validity, we have:

$$
\begin{array}{c}
\neg\, A \\
\forall x\,(Px \vee Qx) \\
\neg\,((\forall x\, Px) \vee (\forall x\, Qx)) \\
\neg \forall x\, Px \\
\neg \forall x\, Qx \\
\neg\, Pa \\
\neg\, Qb \\
Pa \vee Qa \\
Pb \vee Qb \\
\quad / \qquad \backslash \\
Pa \qquad\quad Qa \\
\qquad\quad / \quad \backslash \\
\qquad Pb \quad Qb
\end{array}
$$

This tableau has a unique open path, which gives rise (via the proof of Hintikka's Lemma) to a finite L-structure M with $U_M = \{a, b\}$, $P_M = \{b\}$, $Q_M = \{a\}$. Clearly M falsifies A.

2.6 The Compactness Theorem

Theorem 2.6.1 (the Compactness Theorem, countable case). Let S be a countably infinite set of sentences of the predicate calculus. S is satisfiable if and only if each finite subset of S is satisfiable.

Proof. FIXME □

Theorem 2.6.2 (the Compactness Theorem, uncountable case). Let S be an uncountable set of sentences of the predicate calculus. S is satisfiable if and only if each finite subset of S is satisfiable.

Proof. FIXME □

Exercise 2.6.3. Let L be a language consisting of a binary predicate R and some additional predicates. Let $M = (U_M, R_M, \ldots)$ be an L-structure such that (U_M, R_M) is isomorphic to $(\mathbb{N}, <_{\mathbb{N}})$. Note that M contains no infinite R-descending sequence. Show that there exists an L-structure M' such that:

1. M and M' satisfy the same L-sentences.
2. M' contains an infinite R-descending sequence. In other words, there exist elements $a'_1, a'_2, \ldots, a'_n, \ldots \in U_{M'}$ such that $\langle a'_{n+1}, a'_n \rangle \in R_{M'}$ for all $n = 1, 2, \ldots$.

Hint: Use the Compactness Theorem.

Exercise 2.6.4. Generalize Exercise 2.6.3 replacing $(\mathbb{N}, <_{\mathbb{N}})$ by an arbitrary infinite linear ordering with no infinite descending sequence.

2.7 Satisfiability in a Domain

The notion of satisfiability in a domain was introduced in Definition 2.2.9.

Theorem 2.7.1. Let S be a set of L-sentences.

1. Assume that S is finite or countably infinite. If S is satisfiable, then S is satisfiable in some finite or countably infinite domain.

2. Assume that S is of cardinality $\aleph_\alpha$. If S is satisfiable, then S is satisfiable in some domain of cardinality less than or equal to $\aleph_\alpha$.

Proof. Parts 1 and 2 follow easily from the proofs of Compactness Theorems 2.6.1 and 2.6.2, respectively.
FIXME □

In Example 2.2.12 we have seen a sentence A_∞ which is satisfiable in a countably infinite domain but not in any finite domain. Regarding satisfiability in finite domains, we have:

Example 2.7.2. Given a positive integer n, we exhibit a sentence A_n which is satisfiable in a domain of cardinality n but not in any domain of smaller cardinality. Our sentence A_n is (1) & (2) & (3) with

$$(1) \quad \forall x\, \forall y\, \forall z\, ((Rxy \,\&\, Ryz) \Rightarrow Ryz)$$
$$(2) \quad \forall x\, \forall y\, (Rxy \Rightarrow \neg\, Ryx)$$
$$(3) \quad \exists x_1 \cdots \exists x_n\, (Rx_1x_2 \,\&\, Rx_2x_3 \,\&\, \cdots \,\&\, Rx_{n-1}x_n)$$

On the other hand, we have:

Theorem 2.7.3. Let M and M' be L-structures. Assume that there exists an onto mapping $\phi : U_M \to U_{M'}$ such that for all n-ary predicates P of L and all n-tuples $\langle a_1, \ldots, a_n\rangle \in (U_M)^n$, $\langle a_1, \ldots, a_n\rangle \in P_M$ if and only if $\langle \phi(a_1), \ldots, \phi(a_n)\rangle \in P_{M'}$. Then as in Theorem 2.2.6 we have $v_M(A) = v_{M'}(A')$ for all L-U_M-sentences A, where $A' = A[a_1/\phi(a_1), \ldots, a_k/\phi(a_k)]$. In particular, M and M' satisfy the same L-sentences.

Proof. The proof is by induction on the degree of A. Suppose for example that $A = \forall x\, B$. Then by definition of v_M we have that $v_M(A) = \mathrm{T}$ if and only if $v_M(B[x/a]) = \mathrm{T}$ for all $a \in U_M$. By inductive hypothesis, this holds if and only if $v_{M'}(B[x/a]') = \mathrm{T}$ for all $a \in U_M$. But for all $a \in U_M$ we have $B[x/a]' = B'[x/\phi(a)]$. Thus our condition is equivalent to $v_{M'}(B'[x/\phi(a)]) = \mathrm{T}$ for all $a \in U_M$. Since $\phi : U_M \to U_{M'}$ is onto, this is equivalent to $v_{M'}(B'[x/b]) = \mathrm{T}$ for all $b \in U_{M'}$. By definition of $v_{M'}$ this is equivalent to $v_{M'}(\forall x\, B') = \mathrm{T}$. But $\forall x\, B' = A'$, so our condition is equivalent to $v_{M'}(A') = \mathrm{T}$. □

Corollary 2.7.4. Let S be a set of L-sentences. If S is satisfiable in a domain U, then S is satisfiable in any domain of the same or larger cardinality.

Proof. Suppose S is satisfiable in domain U. Let U' be a set of cardinality greater than or equal to that of U. Let $\phi : U' \to U$ be onto. If M is any L-structure with $U_M = U$, we can define an L-structure M' with $U_{M'} = U'$ by putting $P_{M'} = \{\langle a_1, \ldots, a_n\rangle : \langle \phi(a_1), \ldots, \phi(a_n)\rangle \in P_M\}$ for all n-ary predicates P of L. By Theorem 2.7.3, M and M' satisfy the same L-sentences. In particular, if M satisfies S, then M' satisfies S. □

Remark 2.7.5. We shall see later[8] that Theorem 2.7.3 and Corollary 2.7.4 fail for normal satisfiability.

[8] See Section 4.1.

Chapter 3

Proof Systems for Predicate Calculus

3.1 Introduction to Proof Systems

Definition 3.1.1. An *abstract proof system* consists of a set $\mathfrak{X}$ together with a relation $\mathfrak{R} \subseteq \bigcup_{k=0}^{\infty} \mathfrak{X}^{k+1}$. Elements of $\mathfrak{X}$ are called *objects*. Elements of $\mathfrak{R}$ are called *rules of inference*. An object $X \in \mathfrak{X}$ is said to be *derivable*, or *provable*, if there exists a finite sequence of objects $X_1, \ldots, X_n$ such that $X_n = X$ and, for each $i \leq n$, there exist $j_1, \ldots, j_k < i$ such that $\langle X_{j_1}, \ldots, X_{j_k}, X_i \rangle \in \mathfrak{R}$. The sequence $X_1, \ldots, X_n$ is called a *derivation* of X, or a *proof* of X.

Notation 3.1.2. For $k \geq 1$ it is customary to write

$$\frac{X_1 \ \cdots \ X_k}{Y}$$

indicating that $\langle X_1, \ldots, X_k, Y \rangle \in \mathfrak{R}$. This is to be understood as "from the premises $X_1, \ldots, X_k$ we may immediately infer the conclusion Y". For $k = 0$ we may write

$$\frac{}{Y}$$

or simply Y, indicating that $\langle Y \rangle \in \mathfrak{R}$. This is to be understood as "we may infer Y from no premises", or "we may assume Y".

Definition 3.1.3. Let L be a language. Recall that V is the set of parameters. A *Hilbert-style proof system* for L is a proof system with the following properties:

1. The objects are sentences with parameters. In other words,

$$\mathfrak{X} = \{A : A \text{ is an } L\text{-}V\text{-sentence}\}.$$

2. For each rule of inference

$$\frac{A_1 \ \cdots \ A_k}{B}$$

(i.e., $\langle A_1, \ldots, A_k, B\rangle \in \mathfrak{R}$), we have that B is a logical consequence of $A_1, \ldots, A_k$. This property is known as *soundness*. It implies that every L-V-sentence which is derivable is logically valid.

3. For all L-V-sentences A, B, we have a rule of inference $\langle A, A \Rightarrow B, B\rangle \in \mathfrak{R}$, i.e.,
$$\frac{A \quad A \Rightarrow B}{B}.$$
In other words, from A and $A \Rightarrow B$ we immediately infer B. This collection of inference rules is known as *modus ponens*.

4. An L-V-sentence A is logically valid if and only if A is derivable. This property is known as *completeness*.

Remark 3.1.4. In Section 3.3 we shall explicitly exhibit a particular Hilbert-style proof system, LH. The soundness of LH will be obvious. In order to verify the completeness of LH, we shall first prove a result known as the Companion Theorem, which is also of interest in its own right.

3.2 The Companion Theorem

In this section we shall comment on the notion of logical validity for sentences of the predicate calculus. We shall analyze logical validity into two components: a propositional component (quasitautologies), and a quantificational component (companions).

Definition 3.2.1 (quasitautologies).

1. A *tautology* is a propositional formula which is logically valid.

2. A *quasitautology* is an L-V-sentence of the form $F[p_1/A_1, \ldots, p_k/A_k]$, where F is a tautology, $p_1, \ldots, p_k$ are the atoms occurring in F, and $A_1, \ldots, A_k$ are L-V-sentences.

For example, $p \Rightarrow (q \Rightarrow p)$ is a tautology. This implies that, for all L-V-sentences A and B, $A \Rightarrow (B \Rightarrow A)$ is a quasitautology.

Remarks 3.2.2.

1. Obviously, every quasitautology is logically valid.

2. There is a decision procedure[1] for quasitautologies. One such decision procedure is based on truth tables. Another is based on propositional tableaux.

[1] In other words, there is a Turing algorithm which, given an L-V-sentence A as input, will eventually halt with output 1 if A is a quasitautology, 0 if A is not a quasitautology.

3. It can be shown that there is no decision procedure for logical validity. (This result is known as Church's Theorem.) Therefore, in relation to the problem of characterizing logical validity, we regard the quasitautologies as trivial.

Let A be an L-V-sentence.

Definition 3.2.3 (companions). A *companion* of A is any L-V-sentence of one of the forms

$$(1) \qquad (\forall x\, B) \Rightarrow B[x/a]$$

$$(2) \qquad B[x/a] \Rightarrow (\forall x\, B)$$

$$(3) \qquad (\exists x\, B) \Rightarrow B[x/a]$$

$$(4) \qquad B[x/a] \Rightarrow (\exists x\, B)$$

where, in (2) and (3), the parameter a may not occur in A or in B.

Lemma 3.2.4. Let C be a companion of A.

1. A is satisfiable if and only if $C \,\&\, A$ is satisfiable.

2. A is logically valid if and only if $C \Rightarrow A$ is logically valid.

Proof. Let C be a companion of A.

For part 1, assume that A is satisfiable. In accordance with Definition 2.3.7, let M, ϕ be an L-V-structure satisfying A. If C is of the form 3.2.3(1) or 3.2.3(4), then C is logically valid, hence M, ϕ satisfies $C \,\&\, A$. Next, consider the case when C is of the form 3.2.3(2). If M, ϕ satisfies $\forall x\, B$, then M, f satisfies C. Otherwise we have $v_M(\forall x\, B^\phi) = \mathrm{F}$, so let $c \in U_M$ be such that $v_M(B^\phi[x/c]) = \mathrm{F}$. Define $\phi' : V \to U_M$ by putting $\phi'(a) = c$, $\phi'(b) = \phi(b)$ for $b \neq a$. Since a does not occur in A, we have that M, ϕ' satisfies A. Also, since a does not occur in B, we have $B[x/a]^{\phi'} = B^{\phi'}[x/c] = B^\phi[x/c]$, hence $v_M(B[x/a]^{\phi'}) = v_M(B^\phi[x/c]) = \mathrm{F}$, i.e., M, ϕ' satisfies $\neg\, B[x/a]$. Thus M, ϕ' satisfies $C \,\&\, A$. The case when C is of the form 3.2.3(3) is handled similarly.

For part 2 note that, since C is a companion of A, C is a companion of $\neg\, A$. Thus we have that A is logically valid if and only if $\neg\, A$ is not satisfiable, if and only if $C \,\&\, \neg\, A$ is not satisfiable (by part 1), if and only if $\neg\, (C \,\&\, \neg\, A)$ is logically valid, i.e., $C \Rightarrow A$ is logically valid. □

Definition 3.2.5 (companion sequences). A *companion sequence* of A is a finite sequence $C_1, \ldots, C_n$ such that, for each $i < n$, C_{i+1} is a companion of

$$(C_1 \,\&\, \cdots \,\&\, C_i) \Rightarrow A.$$

Lemma 3.2.6. If $C_1, \ldots, C_n$ is a companion sequence of A, then A is logically valid if and only if $(C_1 \,\&\, \cdots \,\&\, C_n) \Rightarrow A$ is logically valid.

Proof. Note that $(C_1 \,\&\, \cdots \,\&\, C_n) \Rightarrow A$ is quasitautologically equivalent to

$$C_n \Rightarrow (C_{n-1} \Rightarrow \cdots \Rightarrow (C_1 \Rightarrow A)).$$

Our lemma follows by n applications of part 2 of Lemma 3.2.4. $\square$

Theorem 3.2.7 (the Companion Theorem). A is logically valid if and only if there exists a companion sequence $C_1, \ldots, C_n$ of A such that

$$(C_1 \,\&\, \cdots \,\&\, C_n) \Rightarrow A$$

is a quasitautology.

Proof. The "if" part is immediate from Lemma 3.2.6. For the "only if" part, assume that A is logically valid. By Theorem 2.5.5 let τ be a finite closed unsigned tableau starting with $\neg\, A$. Thus we have a finite sequence of tableaux $\tau_0, \tau_1, \ldots, \tau_n$ where $\tau_0 = \neg\, A$, $\tau_n = \tau$, and each τ_{i+1} is obtained by applying a tableau rule R_i to τ_i. If R_i is a quantifier rule, let C_i be an appropriate companion. Thus $C_1, \ldots, C_n$ is a companion sequence for A, and we can easily transform τ into a closed tableau τ' starting with

$$\begin{array}{c} \neg\, A \\ C_1 \\ \vdots \\ C_n \end{array}$$

in which only propositional tableau rules are applied. Thus $(C_1 \,\&\, \cdots \,\&\, C_n) \Rightarrow A$ is a quasitautology. This proves our theorem.

For instance, if R_i is the tableau rule

$$(*) \qquad \begin{array}{c} \vdots \\ \forall x\, B \\ \vdots \\ | \\ B[x/a]\ , \end{array}$$

where a is an arbitrary parameter, let C_i be the companion $(\forall x\, B) \Rightarrow B[x/a]$, and replace the application of $(*)$ by

$$\begin{array}{c} \vdots \\ (\forall x\, B) \Rightarrow B[x/a] \\ \vdots \\ \forall x\, B \\ \vdots \\ / \quad \backslash \\ \neg\, \forall x\, B \qquad B[x/a] \end{array}$$

noting that the left-hand path is closed.

Similarly, if R_i is the tableau rule

$$(**)\qquad \begin{array}{c} \vdots \\ \neg\,\forall x\,B \\ \vdots \\ | \\ \neg\,B[x/a] \end{array}$$

where a is a new parameter, let C_i be the companion $B[x/a] \Rightarrow (\forall x\,B)$, and replace the application of $(**)$ by

$$\begin{array}{c} \vdots \\ B[x/a] \Rightarrow (\forall x\,B) \\ \vdots \\ \neg\,\forall x\,B \\ \vdots \\ \diagup \quad \diagdown \\ \neg\,B[x/a] \qquad \forall x\,B \end{array}$$

noting that the right-hand path is closed. □

Example 3.2.8. As an example illustrating Theorem 3.2.7, let A be the sentence $(\exists x\,(Px \vee Qx)) \Rightarrow ((\exists x\,Px) \vee (\exists x\,Qx))$. Let τ be the closed tableau

$$\begin{array}{c} \neg\,A \\ \exists x(Px \vee Qx) \\ \neg\,((\exists x\,Px) \vee (\exists x\,Qx)) \\ \neg\,\exists x\,Px \\ \neg\,\exists x\,Qx \\ Pa \vee Qa \\ \neg\,Pa \\ \neg\,Qa \\ \diagup \quad \diagdown \\ Pa \qquad Qa \end{array}$$

which shows that A is logically valid. Examining the applications of quantifier rules in τ, we obtain the companion sequence C_1, C_2, C_3 for A, where C_1 is $(\exists x\,(Px \vee Qx)) \Rightarrow (Pa \vee Qa)$, C_2 is $Pa \Rightarrow \exists x\,Px$, C_3 is $Qa \Rightarrow \exists x\,Qx$. Clearly $(C_1 \,\&\, C_2 \,\&\, C_3) \Rightarrow A$ is a quasitautology.

3.3 A Hilbert-Style Proof System

Let L be a language. Recall that V is the set of parameters.

Definition 3.3.1 (the system LH). Our Hilbert-style proof system LH for the predicate calculus is as follows:

1. The objects are L-V-sentences.
2. For each quasitautology A, $\langle A \rangle$ is a rule of inference.
3. $\langle (\forall x\, B) \Rightarrow B[x/a] \rangle$ and $\langle B[x/a] \Rightarrow (\exists x\, B) \rangle$ are rules of inference.
4. $\langle A, A \Rightarrow B, B \rangle$ is a rule of inference.
5. $\langle A \Rightarrow B[x/a], A \Rightarrow (\forall x\, B) \rangle$ and $\langle B[x/a] \Rightarrow A, (\exists x\, B) \Rightarrow A \rangle$ are rules of inference, provided the parameter a does not occur in A or in B.

Schematically, LH consists of:

1. A, where A is any quasitautology
2. (a) $(\forall x\, B) \Rightarrow B[x/a]$ (universal instantiation)
   (b) $B[x/a] \Rightarrow (\exists x\, B)$ (existential instantiation)
3. $\dfrac{A \quad A \Rightarrow B}{B}$ (modus ponens)
4. (a) $\dfrac{A \Rightarrow B[x/a]}{A \Rightarrow (\forall x\, B)}$ (universal generalization)
   (b) $\dfrac{B[x/a] \Rightarrow A}{(\exists x\, B) \Rightarrow A}$ (existential generalization),
   where a does not occur in A, B.

Lemma 3.3.2 (soundness of LH). LH is sound. In other words, for all L-V-sentences A, if A is derivable, then A is logically valid.

Proof. The proof is straightforward by induction on the length of a derivation. The induction step is similar to the proof of Lemma 3.2.4. □

Example 3.3.3. In LH we have the following derivation:

1. $(\forall x\, A) \Rightarrow A[x/a]$ (universal instantiation)
2. $A[x/a] \Rightarrow (\exists x\, A)$ (existential instantiation)
3. $((\forall x\, A) \Rightarrow A[x/a]) \Rightarrow ((A[x/a] \Rightarrow (\exists x\, A)) \Rightarrow ((\forall x\, A) \Rightarrow (\exists x\, A)))$
   (This is a quasitautology, obtained from the tautology $(p \Rightarrow q) \Rightarrow ((q \Rightarrow r) \Rightarrow (p \Rightarrow r))$.)
4. $(A[x/a] \Rightarrow (\exists x\, A)) \Rightarrow ((\forall x\, A) \Rightarrow (\exists x\, A))$ (1, 3, modus ponens)
5. $(\forall x\, A) \Rightarrow (\exists x\, A)$ (2, 4, modus ponens)

Thus, by Lemma 3.3.2, $(\forall x\, A) \Rightarrow (\exists x\, A)$ is logically valid.

Example 3.3.4. In LH we have the following derivation:

1. $B[x/a] \Rightarrow (\exists x\, B)$ (existential instantiation)
2. $(B[x/a] \Rightarrow (\exists x\, B)) \Rightarrow ((A \,\&\, B)[x/a] \Rightarrow (\exists x\, B))$ (quasitautology)
3. $(A \,\&\, B)[x/a] \Rightarrow (\exists x\, B)$ (1, 2, modus ponens)
4. $(\exists x\, (A \,\&\, B)) \Rightarrow (\exists x\, B)$ (3, existential generalization)

Thus, by Lemma 3.3.2, $(\exists x\, (A \,\&\, B)) \Rightarrow (\exists x\, B)$ is logically valid.

We now turn to the proof that LH is complete.

Lemma 3.3.5. LH is closed under quasitautological consequence. In other words, if $A_1, \ldots, A_k$ are derivable, and if B is a quasitautological consequence of $A_1, \ldots, A_k$, then B is derivable.

Proof. We are assuming that B is a quasitautological consequence of $A_1, \ldots, A_k$. Thus $A_1 \Rightarrow (A_2 \Rightarrow \cdots \Rightarrow (A_k \Rightarrow B))$ is a quasitautology, hence derivable. We are also assuming that $A_1, \ldots, A_k$ are derivable. Thus we obtain B by k applications of modus ponens. □

Lemma 3.3.6. If C is a companion of A, and if $C \Rightarrow A$ is derivable in LH, then A is derivable in LH.

Proof. First, suppose C is of the form 3.2.3(1), namely $(\forall x\, B) \Rightarrow B[x/a]$. By universal instantiation, C is derivable. In addition, we are assuming that $C \Rightarrow A$ is derivable. Hence, by modus ponens, A is derivable.

Next, suppose C is of the form 3.2.3(2), namely $B[x/a] \Rightarrow (\forall x\, B)$, where a does not occur in A, B. We are assuming that $C \Rightarrow A$ is derivable, i.e., $(B[x/a] \Rightarrow (\forall x\, B)) \Rightarrow A$ is derivable. It follows by Lemma 3.3.5 that both (i) $(\neg\, A) \Rightarrow B[x/a]$ and (ii) $(\neg\, A) \Rightarrow (\neg\, \forall x\, B)$ are derivable. Applying universal generalization to (i), we see that $(\neg\, A) \Rightarrow (\forall x\, B)$ is derivable. From this plus (ii), it follows by Lemma 3.3.5 that A is derivable.

The other cases, where C is of the form 3.2.3(3) or 3.2.3(4), are handled similarly. □

Theorem 3.3.7 (completeness of LH). LH is sound and complete. In other words, for all L-V-sentences A, A is derivable if and only if A is logically valid.

Proof. The "only if" part is Lemma 3.3.2. For the "if" part, assume that A is logically valid. By Theorem 3.2.7, there exists a companion sequence $C_1, \ldots, C_n$ for A such that $(C_1 \,\&\, \cdots \,\&\, C_n) \Rightarrow A$ is a quasitautology. Hence $C_n \Rightarrow (C_{n-1} \Rightarrow \cdots \Rightarrow (C_1 \Rightarrow A))$ is a quasitautology, hence derivable. From this and n applications of Lemma 3.3.6, we obtain derivability of A. □

Exercise 3.3.8. Consider the following proof system LH', which is a "stripped down" version of LH. The objects of LH' are L-V-sentences containing only $\forall$, $\Rightarrow$, $\neg$ (i.e., not containing $\exists$, $\Leftrightarrow$, $\&$, $\vee$). The rules of LH' are:

1. quasitautologies
2. $(\forall x\, B) \Rightarrow B[x/a]$
3. $(\forall x\, (A \Rightarrow B)) \Rightarrow (A \Rightarrow \forall x\, B)$
4. $\dfrac{A \;\; A \Rightarrow B}{B}$ (modus ponens)
5. $\dfrac{B[x/a]}{\forall x\, B}$ (generalization), where a does not occur in B.

Show that LH' is sound and complete.

Exercise 3.3.9.

1. Let S be a set of L-sentences. Consider a proof system $LH(S)$ consisting of LH with additional rules of inference $\langle A \rangle$, $A \in S$. Show that an L-V-sentence B is derivable in $LH(S)$ if and only if B is a logical consequence of S.
2. Indicate the modifications needed when S is a set of L-V-sentences.

Notation 3.3.10. We write $S \vdash B$ to indicate that B is derivable in $LH(S)$.

3.4 Gentzen-Style Proof Systems

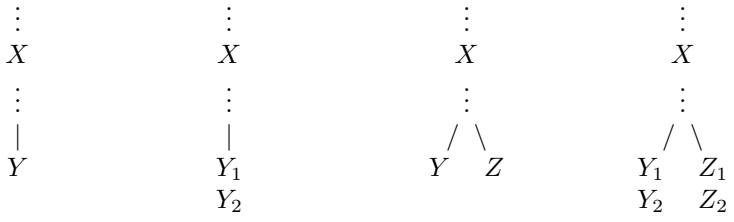
Throughout this section, let L be a language. As usual, V is the set of parameters.

Before presenting our Gentzen-style proof system for L, we first discuss the block tableau method, a trivial variant of the signed tableau method.

Definition 3.4.1. A *block* is a finite set of signed L-V-sentences. A block is said to be *closed* if it contains $\mathrm{T}\, A$ and $\mathrm{F}\, A$ for some L-V-sentence A.

Notation 3.4.2. If S is a block and X is a signed L-V-sentence, we write S, X instead of $S \cup \{X\}$, etc.

Definition 3.4.3. A *block tableau* is a rooted dyadic tree where each node carries a block. A block tableau is said to be *closed* if each of its end nodes is closed. Given a block S, a *block tableau starting with* S is a block tableau generated from S by means of block tableau rules. The *block tableau rules* are obtained from the signed tableau rules (pages 11 and 23) as follows. Corresponding to signed tableau rules of the form



we have block tableau rules

$$\begin{array}{c} S, X \\ | \\ S, X, Y \end{array} \qquad \begin{array}{c} S, X \\ | \\ S, X, Y_1, Y_2 \end{array} \qquad \begin{array}{c} S, X \\ / \quad \backslash \\ S, X, Y \quad S, X, Z \end{array} \qquad \begin{array}{c} S, X \\ / \quad \backslash \\ S, X, Y_1, Y_2 \quad S, X, Z_1, Z_2 \end{array}$$

respectively.

For example, we have the following block tableau rules:

$$\begin{array}{c} S, \mathrm{T}\, A \,\&\, B \\ | \\ S, \mathrm{T}\, A \,\&\, B, \mathrm{T}\, A, \mathrm{T}\, B \end{array} \qquad \begin{array}{c} S, \mathrm{F}\, A \,\&\, B \\ / \quad \backslash \\ S, \mathrm{F}\, A \,\&\, B, \mathrm{F}\, A \quad S, \mathrm{F}\, A \,\&\, B, \mathrm{F}\, B \end{array}$$

$$\begin{array}{c} S, \mathrm{T}\, A \Rightarrow B \\ / \quad \backslash \\ S, \mathrm{T}\, A \Rightarrow B, \mathrm{F} A \quad S, \mathrm{T}\, A \Rightarrow B, \mathrm{T} B \end{array} \qquad \begin{array}{c} S, \mathrm{F}\, A \Rightarrow B \\ | \\ S, \mathrm{F}\, A \Rightarrow B, \mathrm{T}\, A, \mathrm{F}\, B \end{array}$$

$$\begin{array}{c} S, \mathrm{T}\, \forall x\, A \\ | \\ S, \mathrm{T}\, \forall x\, A, \mathrm{T}\, A[x/a] \end{array} \qquad \begin{array}{c} S, \mathrm{F}\, \forall x\, A \\ | \\ S, \mathrm{F}\, \forall x\, A, \mathrm{F}\, A[x/a] \end{array}$$

where a is new.

Example 3.4.4. We exhibit a closed block tableau demonstrating that $\exists x\, A$ is a logical consequence of $\forall x\, A$.

$$\begin{array}{c} \mathrm{T}\, \forall x\, A,\ \mathrm{F}\, \exists x\, A \\ | \\ \mathrm{T}\, \forall x\, A,\ \mathrm{F}\, \exists x\, A,\ \mathrm{T}\, A[x/a] \\ | \\ \mathrm{T}\, \forall x\, A,\ \mathrm{F}\, \exists x\, A,\ \mathrm{T}\, A[x/a],\ \mathrm{F}\, A[x/a] \end{array}$$

This block tableau is of course similar to the signed tableau

$$\begin{array}{c} \mathrm{T}\, \forall x\, A \\ \mathrm{F}\, \exists x\, A \\ \mathrm{T}\, A[x/a] \\ \mathrm{F}\, A[x/a] \end{array}$$

which demonstrates the same thing.

We now define our Gentzen-style system, LG.

Definition 3.4.5. A *sequent* is an expression of the form $\Gamma \to \Delta$ where Γ and Δ are finite sets of L-V-sentences. If $S = \mathrm{T}\, A_1, \ldots, \mathrm{T}\, A_n, \mathrm{F}\, B_1, \ldots, \mathrm{F}\, B_n$ is a block, let $|S|$ be the sequent $A_1, \ldots, A_m \to B_1, \ldots, B_n$. This gives a one-to-one correspondence between blocks and sequents.

Definition 3.4.6 (the system LG). Our Gentzen-style proof system LG for the predicate calculus is as follows.

1. The objects of LG are sequents.[2]
2. For each closed block S, we have a rule of inference $\langle |S| \rangle$. In other words, for all finite sets of L-V-sentences Γ and Δ and all L-V-sentences A, we assume the sequent $\Gamma, A \to A, \Delta$.
3. For each non-branching block tableau rule

$$\begin{array}{c} S \\ | \\ S' \end{array}$$

we have a rule of inference $\langle |S_1|, |S| \rangle$, i.e., $\dfrac{|S'|}{|S|}$.

4. For each branching block tableau rule

$$\begin{array}{ccc} & S & \\ / & & \backslash \\ S' & & S'' \end{array}$$

we have a rule of inference $\langle |S'|, |S''|, |S| \rangle$, i.e., $\dfrac{|S'| \quad |S''|}{|S|}$.

Thus LG includes the following rules of inference:

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

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

$$\frac{\Gamma, A \,\&\, B, A, B \to \Delta}{\Gamma, A \,\&\, B \to \Delta} \qquad \frac{\Gamma \to A \,\&\, B, A, \Delta \quad \Gamma \to A \,\&\, B, B, \Delta}{\Gamma \to A \,\&\, B, \Delta}$$

$$\frac{\Gamma, A \Rightarrow B \to A, \Delta \quad \Gamma, A \Rightarrow B, B \to \Delta}{\Gamma, A \Rightarrow B \to \Delta} \qquad \frac{\Gamma, A \to A \Rightarrow B, B, \Delta}{\Gamma \to A \Rightarrow B, \Delta}$$

$$\frac{\Gamma, \forall x\, A, A[x/a] \to \Delta}{\Gamma, \forall x\, A \to \Delta} \qquad \frac{\Gamma \to \forall x\, A, A[x/a], \Delta}{\Gamma \to \forall x\, A, \Delta}$$

where a does not occur in the conclusion.

[2] For this reason, LG is sometimes called a *sequent calculus*.

Exercise 3.4.7. Explicitly display the remaining inference rules of LG.

Definition 3.4.8. A sequent $A_1, \ldots, A_m \to B_1, \ldots, B_n$ is said to be *logically valid* if and only if the L-V-sentence

$$(A_1 \,\&\, \cdots \,\&\, A_m) \Rightarrow (B_1 \vee \cdots \vee B_n)$$

is logically valid.

Theorem 3.4.9 (soundness and completeness of LG). LG is sound and complete. In other words, a sequent $\Gamma \to \Delta$ is logically valid if and only if it is derivable in LG. In particular, an L-V-sentence A is logically valid if and only if the sequent

$$\to A$$

is derivable in LG.

Proof. Note that the sequent $A_1, \ldots, A_m \to B_1, \ldots, B_n$ is logically valid if and only if the block $\mathrm{T}\, A_1, \ldots, \mathrm{T}\, A_m, \mathrm{F}\, B_1, \ldots, \mathrm{F}\, B_n$ is not satisfiable. Thus, soundness and completeness of LG is equivalent to soundness and completeness of the block tableau method. The latter is in turn easily seen to be equivalent to soundness and completeness of the signed tableau method, as presented in Theorems 2.3.9 and 2.5.5. □

Exercise 3.4.10. Let LG(atomic) be a variant of LG in which $\Gamma, A \to A, \Delta$ is assumed only for atomic L-V-sentences A. Show that LG(atomic) is sound and complete. (Hint: Use the result of Exercise 2.5.7.)

Exercise 3.4.11.

1. The *modified block tableau rules* are a variant of the block tableau rules of Definition 3.4.3, replacing each non-branching rule of the form

$$\begin{array}{c} S, X \\ | \\ S, X, Y_1, Y_2 \end{array}$$

by a pair of rules

$$\begin{array}{c} S, X \\ | \\ S, X, Y_1 \end{array} \qquad\qquad \begin{array}{c} S, X \\ | \\ S, X, Y_2 \end{array}$$

Show that the modified block tableau rules are sound and complete.

2. Let LG' be the variant of LG corresponding to the modified block tableau rules. Write out all the rules of LG' explicitly. Show that LG' is sound and complete.

3.5 The Interpolation Theorem

As usual, let L be a language and let V be the set of parameters.

Theorem 3.5.1 (the Interpolation Theorem). Let A and B be L-V-sentences. If $A \Rightarrow B$ is logically valid, we can find an L-V-sentence I such that:

1. $A \Rightarrow I$ and $I \Rightarrow B$ are logically valid.
2. Each predicate and parameter occurring in I occurs in both A and B.

Such an I is called an *interpolant* for $A \Rightarrow B$. We indicate this by writing $A \overset{I}{\Rightarrow} B$.

Remark 3.5.2. If A and B have no predicates in common, then obviously the theorem is incorrect as stated, because all L-V-sentences necessarily contain at least one predicate. In this case, we modify the conclusion of the theorem to say that at least one of $\neg A$ and B is logically valid.[3] The conclusion is obvious in this case.

In order to prove the Interpolation Theorem, we introduce a "symmetric" variant of LG, wherein sentences do not move from one side of $\rightarrow$ to the other.

Definition 3.5.3. A *signed sequent* is an expression of the form $M \rightarrow N$ where M and N are finite sets of signed L-V-sentences. A *variant* of $M \rightarrow N$ is a signed sequent obtained from $M \rightarrow N$ by transferring sentences from one side of $\rightarrow$ to the other, changing signs. In particular, $M, X \rightarrow N$ and $M \rightarrow \overline{X}, N$ are variants of each other, where we use an overline to denote conjugation, i.e., $\overline{\mathrm{T}\,A} = \mathrm{F}\,A$, $\overline{\mathrm{F}\,A} = \mathrm{T}\,A$.

Definition 3.5.4. Let

$$C_1, \ldots, C_m \rightarrow D_1, \ldots, D_n$$

be an unsigned sequent[4]. A *signed variant* of $C_1, \ldots, C_m \rightarrow D_1, \ldots, D_n$ is any variant of the signed sequent

$$\mathrm{T}\,C_1, \ldots, \mathrm{T}\,C_m \rightarrow \mathrm{T}\,D_1, \ldots, \mathrm{T}\,D_n\,.$$

Note that each signed sequent is a signed variant of one and only one unsigned sequent. We define a signed sequent to be *logically valid* if and only if the corresponding unsigned sequent is logically valid.

Definition 3.5.5. LG(symmetric) is the following proof system.

1. The objects are signed sequents.

[3] This amounts to saying that at least one of the truth values T and F is an interpolant for $A \Rightarrow B$.

[4] An *unsigned sequent* is just what we have previously called a sequent.

2. We have

$$\frac{}{M, X \to X, N}$$

and

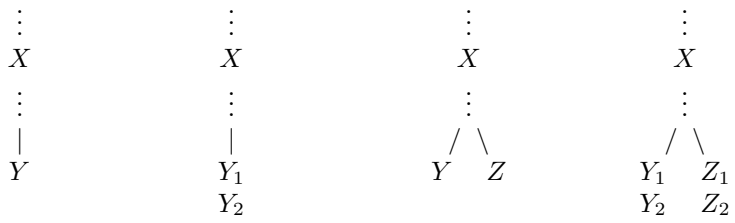
$$\frac{}{M, X, \overline{X} \to N}$$

and

$$\frac{}{M \to X, \overline{X}, N}$$

for all X.

3. For each signed tableau rule of the form



we have a corresponding pair of signed sequent rules

$$\frac{M, X, Y \to N}{M, X \to N} \qquad \frac{M \to \overline{X}, \overline{Y}, N}{M \to \overline{X}, N}$$

$$\frac{M, X, Y_1, Y_2 \to N}{M, X \to N} \qquad \frac{M \to \overline{X}, \overline{Y_1}, \overline{Y_2}, N}{M \to \overline{X}, N}$$

$$\frac{M, X, Y \to N \quad M, X, Z \to N}{M, X \to N} \qquad \frac{M \to \overline{X}, \overline{Y}, N \quad M \to \overline{X}, \overline{Z}, N}{M \to \overline{X}, N}$$

$$\frac{M, X, Y_1, Y_2 \to N \quad M, X, Z_1, Z_2 \to N}{M, X \to N} \qquad \frac{M \to \overline{X}, \overline{Y_1}, \overline{Y_2}, N \quad M \to \overline{X}, \overline{Z_1}, \overline{Z_2}, N}{M \to \overline{X}, N}$$

respectively.

Lemma 3.5.6. An unsigned sequent is derivable in LG if and only if all of its signed variants are derivable in LG(symmetric).

Proof. The proof is by induction on the length of derivations in LG. The base step consists of noting that all signed variants of $\Gamma, A \to A, \Delta$ are of the form $M, X \to X, N$ or $M, X, \overline{X} \to N$ or $M \to X, \overline{X}, N$, hence derivable in LG(symmetric). The inductive step consists of checking that, for each rule of inference of LG, if all signed variants of the premises are derivable in LG(symmetric), then so are all signed variants of the conclusion. This is straightforward. □

Theorem 3.5.7. LG(symmetric) is sound and complete. In other words, a signed sequent is logically valid if and only if it is derivable in LG(symmetric). In particular, an L-V-sentence $A \Rightarrow B$ is logically valid if and only if the signed sequent $\mathrm{T}\,A \to \mathrm{T}\,B$ is derivable in LG(symmetric).

Proof. Soundness and completeness of LG(symmetric) follows from Theorem 3.4.9, soundness and completeness of LG, using Lemma 3.5.6. □

We now prove the Interpolation Theorem.

Definition 3.5.8. Let $M \to N$ be a signed sequent. An *interpolant* for $M \to N$ is an L-V-sentence I such that the signed sequents $M \to \mathrm{T}\,I$ and $\mathrm{T}\,I \to N$ are logically valid, and all predicates and parameters occurring in I occur in both M and N.[5] We indicate this by writing $M \xrightarrow{I} N$.

In order to prove the Interpolation Theorem, it suffices by Theorem 3.5.7 to prove that every signed sequent derivable in LG(symmetric) has an interpolant. We prove this by induction on the length of derivations.

For the base step, we note that X is an interpolant for $M, X \to X, N$, and that $M, X, \overline{X} \to$ and $\to X, \overline{X}, N$ are logically valid. Thus we have $M, X \xrightarrow{X} X, N$ and $M, X, \overline{X} \xrightarrow{\mathrm{F}} N$ and $M \xrightarrow{\mathrm{T}} X, \overline{X}, N$.

For the induction step we show that, for each rule of LG(symmetric), given interpolants for the premises of the rule, we can find an interpolant for the conclusion. We present some representative special cases.

$$\frac{M, \mathrm{T}\,A \,\&\, B, \mathrm{T}\,A, \mathrm{T}\,B \xrightarrow{I} N}{M, \mathrm{T}\,A \,\&\, B \xrightarrow{I} N} \qquad \frac{M \xrightarrow{I} \mathrm{F}\,A \,\&\, B, \mathrm{F}\,A, \mathrm{F}B, N}{M \xrightarrow{I} \mathrm{F}\,A \,\&\, B, N}$$

$$\frac{M, \mathrm{F}\,A \,\&\, B, \mathrm{F}\,A \xrightarrow{I} N \qquad M, \mathrm{F}\,A \,\&\, B, \mathrm{F}\,B \xrightarrow{J} N}{M, \mathrm{F}\,A \,\&\, B \xrightarrow{I \vee J} N}$$

$$\frac{M \xrightarrow{I} \mathrm{T}\,A \,\&\, B, \mathrm{T}\,A, N \qquad M \xrightarrow{J} \mathrm{T}\,A \,\&\, B, \mathrm{T}\,B, N}{M \xrightarrow{I \,\&\, J} \mathrm{T}\,A \,\&\, B, N}$$

$$\frac{M, \mathrm{T} \neg A, \mathrm{F}\,A \xrightarrow{I} N}{M, \mathrm{T} \neg A \xrightarrow{I} N} \qquad \frac{M \xrightarrow{I} \mathrm{F} \neg A, \mathrm{T}\,A, N}{M \xrightarrow{I} \mathrm{F} \neg A, N}$$

$$\frac{M, \mathrm{F} \neg A, \mathrm{T}\,A \xrightarrow{I} N}{M, \mathrm{F} \neg A \xrightarrow{I} N} \qquad \frac{M \xrightarrow{I} \mathrm{T} \neg A, \mathrm{F}\,A, N}{M \xrightarrow{I} \mathrm{T} \neg A, N}$$

[5] In the special case when M and N have no predicates in common, we require instead that at least one of the signed sequents $M \to$ and $\to N$ be logically valid. This amounts to requiring that at least one of T, F be an interpolant for $M \to N$.

$$\frac{M, \mathrm{F}\,\forall x\, A, \mathrm{F}\, A[x/a] \xrightarrow{I} N}{M, \mathrm{F}\,\forall x\, A \xrightarrow{I} N}$$

where a does not occur in the conclusion.

$$\frac{M \xrightarrow{I} \mathrm{T}\,\forall x\, A, \mathrm{T}\, A[x/a], N}{M \xrightarrow{I} \mathrm{T}\,\forall x\, A, N}$$

where a does not occur in the conclusion.

$$\frac{M, \mathrm{T}\,\forall x\, A, \mathrm{T}\, A[x/a] \xrightarrow{I} N}{M, \mathrm{T}\,\forall x\, A \xrightarrow{K} N}$$

where $K = I$ if a occurs in $M, \mathrm{T}\,\forall x\, A$, otherwise $K = \forall z\, I[a/z]$ where z is a new variable.

$$\frac{M \xrightarrow{I} \mathrm{F}\,\forall x\, A, \mathrm{F}\, A[x/a], N}{M \xrightarrow{K} \mathrm{F}\,\forall x\, A, N}$$

where $K = I$ if a occurs in $\mathrm{F}\,\forall x\, A, N$, otherwise $K = \exists z\, I[a/z]$ where z is a new variable.

This completes the proof.

Chapter 4

Extensions of Predicate Calculus

4.1 Predicate Calculus with Identity

Definition 4.1.1. A *language with identity* consists of a language L with a particular binary predicate, I, designated as the *identity predicate*.

Definition 4.1.2. Let L be a language with identity. The *identity axioms for* L are the following sentences:

1. $\forall x\, Ixx$ (reflexivity)
2. $\forall x \forall y\, (Ixy \Leftrightarrow Iyx)$ (symmetry)
3. $\forall x \forall y \forall z\, ((Ixy \,\&\, Iyz) \Rightarrow Ixz)$ (transitivity)
4. For each n-ary predicate P of L, we have an axiom
   $\forall x_1 \cdots \forall x_n \forall y_1 \cdots \forall y_n\, ((Ix_1y_1 \,\&\, \cdots \,\&\, Ix_ny_n) \Rightarrow (Px_1 \cdots x_n \Leftrightarrow Py_1 \cdots y_n))$
   (congruence).

Exercise 4.1.3. Show that the identity predicate is unique in the following sense. If L contains two identity predicates I_1 and I_2, then $\forall x \forall y\, (I_1xy \Leftrightarrow I_2xy)$ is a logical consequence of the identity axioms for I_1 and I_2.

Let L be a language with identity.

Definition 4.1.4. An L-structure M is said to be *normal* if the identity predicate denotes the identity relation, i.e., $I_M = \{\langle a, a\rangle : a \in U_M\}$.

Note that any normal L-structure automatically satisfies the identity axioms for L. Conversely, we have:

Theorem 4.1.5. Let M be an L-structure satisfying the identity axioms for L. For each $a \in U_M$ put $\overline{a} = \{b \in U_M : v_M(Iab) = \mathrm{T}\}$. Then we have a normal L-structure $\overline{M}$ and an onto mapping $\phi : U_M \to U_{\overline{M}}$ as in Theorem 2.7.3, defined by putting $U_{\overline{M}} = \{\overline{a} : a \in U_M\}$, and $P_{\overline{M}} = \{\langle \overline{a}_1, \ldots, \overline{a}_n \rangle : \langle a_1, \ldots, a_n \rangle \in P_M\}$ for all n-ary predicates P.

Proof. This is straightforward, using the fact that I_M is a congruence with respect to each of the relations P_M, $P \in L$. □

Theorem 4.1.6. If M is an L-structure satisfying the identity axioms for L, then we have a normal L-structure $\overline{M}$ satisfying the same sentences as M.

Proof. This is immediate from Theorems 4.1.5 and 2.7.3. □

Let S be a set of L-sentences.

Definition 4.1.7. S is *normally satisfiable* if there exists a normal L-structure which satisfies S.

Corollary 4.1.8. S is normally satisfiable if and only if

$$S \cup \{\text{identity axioms for } L\}$$

is satisfiable.

We also have the Compactness Theorem for normal satisfiability:

Corollary 4.1.9. S is normally satisfiable if and only if each finite subset of S is normally satisfiable.

Proof. This is immediate from Corollary 4.1.8 plus the Compactness Theorem for predicate calculus without identity (Theorems 2.6.1 and 2.6.2). □

Regarding normal satisfiability in particular domains, we have:

Example 4.1.10. Given a positive integer n, we exhibit a sentence E_n which is normally satisfiable in domains of cardinality n but not in domains of any other cardinality. The sentence

$$\exists x_1 \cdots \exists x_n \left(\forall y \left(Ix_1y \vee \cdots \vee Ix_ny\right) \& \neg \left(Ix_1x_2 \vee Ix_1x_3 \vee \cdots \vee Ix_{n-1}x_n\right)\right)$$

has this property. Intuitively, E_n says that there exist exactly n things.

On the other hand, we have:

Theorem 4.1.11.

1. If S is normally satisfiable in arbitrarily large finite domains, then S is normally satisfiable in some infinite domain.

2. If S is normally satisfiable in some infinite domain, then S is normally satisfiable in all infinite domains of cardinality $\geq$ the cardinality of S.

Proof. Use the Compactness Theorem.
FIXME □

Definition 4.1.12. Let A be a sentence of the predicate calculus with identity. The *spectrum of* A is the set of positive integers n such that A is normally satisfiable in a domain of cardinality n. A *spectrum* is a set X of positive integers, such that $X = \text{spectrum}(A)$ for some A.

Remark 4.1.13. The *spectrum problem* is the problem of characterizing the spectra, among all sets of positive integers. This is a famous and apparently difficult open problem.[1] In particular, it is unknown whether the complement of a spectrum is necessarily a spectrum.

Exercises 4.1.14. Prove the following.

1. If X is a finite or cofinite[2] set of positive integers, then X is a spectrum.
2. The set of even numbers is a spectrum.
3. The set of odd numbers is a spectrum.
4. If r and m are positive integers, $\{n \geq 1 : n \equiv r \bmod m\}$ is a spectrum.
5. If X and Y are spectra, $X \cup Y$ and $X \cap Y$ are spectra.

Exercise 4.1.15. Prove that, for any sentence A of the predicate calculus with identity, at least one of spectrum(A) and spectrum($\neg A$) is cofinite. (Hint: Use part 1 of Theorem 4.1.11.)

Example 4.1.16. We show that the set $\{n \geq 1 : n \text{ is even}\}$ is a spectrum.

Let U be a nonempty set. A binary relation $R \subseteq U^2$ is said to be an *equivalence relation* on U if it is reflexive, symmetric, and transitive, i.e., if the structure (U, R) satisfies (1) & (2) & (3):

$$\begin{aligned} &(1) \quad \forall x\, Rxx \\ &(2) \quad \forall x \forall y\, (Rxy \Leftrightarrow Ryx) \\ &(3) \quad \forall x \forall y \forall z\, ((Rxy \,\&\, Ryz) \Rightarrow Rxz) \end{aligned}$$

In this situation, the equivalence classes $[a]_R = \{b \in U : \langle a, b\rangle \in R\}$, $a \in U$, form a *partition of* U, i.e., a decomposition of the set U into pairwise disjoint nonempty subsets.

Let A be the following sentence of the predicate calculus with identity:

$$(1)\ \&\ (2)\ \&\ (3)\ \&\ \forall x \exists y\, ((\neg\, Ixy) \,\&\, \forall z\, (Rxz \Leftrightarrow (Ixz \vee Iyz)))$$

[1] Jones/Selman [1] show that X is a spectrum if and only if there exists a nondeterministic Turing machine which accepts X in time 2^{ck}, where k is the length of the input. Since the input is a positive integer n, we have $k = [\log_2 n]$, as usual in computational number theory.

[2] A set of positive integers is said to be *cofinite* if its complement is finite.

Intuitively, A says that R is an equivalence relation with the property that each equivalence class consists of exactly two elements. Obviously, a finite set U admits an equivalence relation with this property if and only if the cardinality of U is even. Thus the spectrum of A is the set of even numbers.

Example 4.1.17. We show that the set of composite numbers[3] is a spectrum.

Let L be a language consisting of two binary predicates, R and S, as well as the identity predicate, I. Let A be an L-sentence saying that R and S are equivalence relations, each with more than one equivalence class, and $\forall x\,\forall y\,(\exists$ exactly one $z)(Rxz\,\&\,Syz)$. Thus, for any normal L-structure $M = (U_M, R_M, S_M, I_M)$ satisfying A, we have that R_M and S_M partition U_M into "rows" and "columns", respectively, in such a way that the intersection of any "row" with any "column" consists of exactly one element of U_M. Thus, if U_M is finite, the elements of U_M are arranged in an $m \times n$ "matrix", where $m, n \geq 2$. Therefore, the number of elements in U_M is mn, a composite number. Conversely, for any $m, n \geq 2$, there is an L-structure M as above, which satisfies A. Thus spectrum(A) is the set of composite numbers.

Exercise 4.1.18. Let L be a finite language with identity, and let M be a finite normal L-structure. Construct an L-sentence A such that, for all normal L-structures M', M' satisfies A if and only if M' is isomorphic to M.

Exercise 4.1.19. Let L be the following language:

Ox: $x = 1$

$Pxyz$: $x + y = z$

$Qxyz$: $x \times y = z$

Rxy: $x < y$

Sxy: $x + 1 = y$

Ixy: $x = y$ (identity predicate)

For each positive integer n, let M_n be the normal L-structure

$$M_n = (U_n, O_n, P_n, Q_n, R_n, S_n, I_n)$$

where

$U_n = \{1, \ldots, n\}$

$O_n = \{1\}$

$P_n = \{\langle i, j, k\rangle \in (U_n)^3 : i + j = k\}$

$Q_n = \{\langle i, j, k\rangle \in (U_n)^3 : i \times j = k\}$

[3] A *composite number* is an integer greater than 1 which is not prime.

$$R_n = \{\langle i, j\rangle \in (U_n)^2 : i < j\}$$

$$S_n = \{\langle i, j\rangle \in (U_n)^2 : i + 1 = j\}$$

$$I_n = \{\langle i, j\rangle \in (U_n)^2 : i = j\}$$

Exhibit an L-sentence A such that, for all finite normal L-structures M', M' satisfies A if and only if M' is isomorphic to M_n for some n.

Exercise 4.1.20. Let L and M_n be as in Exercise 4.1.19. Show that there exists an infinite normal L-structure $M = M_\infty$ with the following property: for all L-sentences A, if M_p satisfies A for all sufficiently large primes p, then M_∞ satisfies A. (Hint: Use the Compactness Theorem.)

Exercise 4.1.21. Use the result of Exercise 4.1.19 to prove the following:

1. The set of prime numbers and its complement are spectra.
2. The set of squares $\{1, 4, 9, \ldots\}$ and its complement are spectra.
3. The set of powers of 2, $\{2^n : n = 1, 2, 3, \ldots\}$, and its complement, are spectra.
4. The set of prime powers $\{p^n : p \text{ prime}, n = 1, 2, \ldots\}$ and its complement are spectra.

Exercise 4.1.22.

1. The Fibonacci numbers are defined recursively by $F_1 = 1$, $F_2 = 2$, $F_n = F_{n-1} + F_{n-2}$ for $n \geq 3$. Show that the set of Fibonacci numbers

   $$\{F_n : n = 1, 2, \ldots\} = \{1, 2, 3, 5, 8, 13, 21, 34, 55, \ldots\}$$

   and its complement are spectra.
2. Show that $\{x^y : x, y \geq 2\}$ and its complement are spectra.

4.2 Predicate Calculus With Operations

In this section we extend the syntax and semantics of the predicate calculus, to encompass operations. As examples of operations, we may cite the familiar mathematical operations of addition ($+$) and multiplication ($\times$). Such operations are considered binary, because they take two arguments. More generally, we consider n-ary operations.

Definition 4.2.1 (languages). A *language* is a set of predicates $P, Q, R, \ldots$ and *operations* $f, g, h, \ldots$. Each predicate and each operation is designated as *n-ary* for some nonnegative[4] integer n.

[4] A 0-ary operation is sometimes known as a *constant*. Syntactically, constants behave as parameters.

Definition 4.2.2 (terms, formulas, sentences). Let L be a language, and let U be a nonempty set. The set of *L-U-terms* is generated as follows.

1. Each variable is an L-U-term.
2. Each element of U is an L-U-term.
3. If f is an n-ary operation of L, and if $t_1, \ldots, t_n$ are L-U-terms, then $ft_1 \cdots t_n$ is an L-U-term.

An L-U-term is said to be *variable-free* if no variables occur in it. An *atomic L-U-formula* is an expression of the form

$$Pt_1 \cdots t_n$$

where P is an n-ary predicate of L, and $t_1, \ldots, t_n$ are L-U-terms. The set of *L-U-formulas* is generated as in clauses 2, 3, 4 and 5 of Definition 2.1.3. The notions of substitution, free variables, and L-U-sentences are defined as in Section 2.1. Note that $Pt_1 \cdots t_n$ is a sentence if and only if it is variable-free.

Definition 4.2.3 (structures). An *L-structure M* consists of a nonempty set U_M, an n-ary relation $P_M \subseteq (U_M)^n$ for each n-ary predicate P of L, and an n-ary function $f_M : (U_M)^n \to U_M$ for each n-ary operation f of L.

Definition 4.2.4 (isomorphism). Two L-structures M and M' are said to be *isomorphic* if there exists an *isomorphism* of M onto M', i.e., a one-to-one correspondence $\phi : U_M \cong U_{M'}$ such that:

1. for all n-ary predicates P of L and all n-tuples $\langle a_1, \ldots, a_n \rangle \in (U_M)^n$, $\langle a_1, \ldots, a_n \rangle \in P_M$ if and only if $\langle \phi(a_1), \ldots, \phi(a_n) \rangle \in P_{M'}$.
2. for all n-ary operations f of L and all n-tuples $\langle a_1, \ldots, a_n \rangle \in (U_M)^n$, $\phi(f_M(a_1, \ldots, a_n) = f_{M'}(\phi(a_1), \ldots, \phi(a_n))$.

Lemma 4.2.5 (valuations). Let M be an L-structure.

1. There is a unique valuation

   $$v_M : \{t : t \text{ is a variable-free } L\text{-}U_M\text{-term}\} \to U_M$$

   defined as follows:

   (a) $v_M(a) = a$ for all $a \in U_M$.

   (b) $v_M(ft_1 \cdots t_n) = f_M(v_M(t_1), \ldots, v_M(t_n))$ for all n-ary operations f of L and all variable-free L-U_M-terms $t_1, \ldots, t_n$.

2. There is a unique valuation

   $$v_M : \{A : A \text{ is an } L\text{-}U_M\text{-sentence}\} \to \{\mathrm{T}, \mathrm{F}\}$$

defined as follows. For atomic L-U-sentences, we have

$$v_M(Pt_1 \cdots t_n) = \begin{cases} \mathrm{T} & \text{if } \langle v_M(t_1), \ldots, v_M(t_n)\rangle \in P_M, \\ \mathrm{F} & \text{if } \langle v_M(t_1), \ldots, v_M(t_n)\rangle \notin P_M. \end{cases}$$

For non-atomic L-U_M-sentences, $v_M(A)$ is defined as in clauses 2 through 8 of Lemma 2.2.4.

Proof. The proof is as for Lemma 2.2.4. □

Definition 4.2.6 (tableau method). The signed and unsigned tableau methods carry over to predicate calculus with operations. We modify the tableau rules as follows.

Signed:

$$\begin{array}{cc} \vdots & \vdots \\ \mathrm{T}\,\forall x\,A & \mathrm{F}\,\exists x\,A \\ \vdots & \vdots \\ | & | \\ \mathrm{T}\,A[x/t] & \mathrm{F}\,A[x/t] \end{array}$$

where t is a variable-free term

$$\begin{array}{cc} \vdots & \vdots \\ \mathrm{T}\,\exists x\,A & \mathrm{F}\,\forall x\,A \\ \vdots & \vdots \\ | & | \\ \mathrm{T}\,A[x/a] & \mathrm{F}\,A[x/a] \end{array}$$

where a is a new parameter

Unsigned:

$$\begin{array}{cc} \vdots & \vdots \\ \forall x\,A & \neg\,\exists x\,A \\ \vdots & \vdots \\ | & | \\ A[x/t] & \neg\,A[x/t] \end{array}$$

where t is a variable-free term

$$\begin{array}{cc} \vdots & \vdots \\ \exists x\, A & \neg\, \forall x\, A \\ \vdots & \vdots \\ | & | \\ A[x/a] & \neg\, A[x/a] \end{array}$$

where a is a new parameter

Remark 4.2.7 (soundness and completeness). With the tableau rules as above, the Soundness Theorem 2.3.9 carries over unchanged to the context of predicate calculus with operations. The results of Section 2.4 on logical equivalence also carry over. The notion of U-repleteness (Definition 2.5.2) is modified to say that, for example, if S contains $\forall x\, A$ then S contains $A[x/t]$ for all variable-free L-U-terms t. The conclusion of Hintikka's Lemma 2.5.3 is modified to say that S is satisfiable in the domain of variable-free L-U-terms. The conclusion of the Completeness Theorem 2.5.5 is modified to say that $X_1, \ldots, X_k$ is satisfiable in the domain of variable-free L-V-terms. The Compactness Theorems 2.6.1 and 2.6.2 carry over unchanged.

Remark 4.2.8 (satisfiability in a domain). The notion of satisfiability in a domain carries over unchanged to the context of predicate calculus with operations. Theorems 2.2.6 and 2.2.11 on isomorphism, and Theorem 2.7.1 on satisfiability in infinite domains, also carry over. Theorem 2.7.3 carries over in an appropriately modified form. See Theorem 4.2.9 and Exercise 4.2.10 below.

Theorem 4.2.9. Let M and M' be L-structures. Assume that $\phi : U_M \to U_{M'}$ is an onto mapping such that conditions 1 and 2 of Definition 4.2.4 hold. Then as in Theorem 2.2.6 we have $v_M(A) = v_{M'}(A')$ for all L-U_M-sentences A, where $A' = A[a_1/\phi(a_1), \ldots, a_k/\phi(a_k)]$. In particular, M and M' satisfy the same L-sentences.

Proof. The proof of is similar to that of Theorem 2.7.3. □

Exercise 4.2.10. Use Theorem 4.2.9 to show that Corollary 2.7.4 carries over to the context of predicate calculus with operations.

Remark 4.2.11 (companions and proof systems). In our notion of companion (Definition 3.2.3), clauses (1) and (4) are modified as follows:

$$\begin{array}{ll} (1) & (\forall x\, B) \Rightarrow B[x/t] \\ (4) & B[x/t] \Rightarrow (\exists x\, B) \end{array}$$

where t is any variable-free term. In our Hilbert-style proof system LH, the instantiation rules are modified as follows:

(a) $(\forall x\, B) \Rightarrow B[x/t]$ (universal instantiation)

(b) $B[x/t] \Rightarrow (\exists x\, B)$ (existential instantiation)

where t is any variable-free term. Also, our Gentzen-style proof system LG is modified in accordance with the modified tableau rules. With these changes, the soundness and completeness of LG and LH carry over.

Exercise 4.2.12 (the Interpolation Theorem). Strengthen the Interpolation Theorem 3.5.1 to say that each operation, predicate and parameter occurring in I occurs in both A and B. (Hint: The version with operations can be deduced from the version without operations.)

Remark 4.2.13 (predicate calculus with identity). We augment the identity axioms (Definition 4.1.2) as follows:

5. For each n-ary operation f of L, we have an axiom

$\forall x_1 \cdots \forall x_n \forall y_1 \cdots \forall y_n ((Ix_1y_1 \,\&\, \cdots \,\&\, Ix_ny_n) \Rightarrow Ifx_1 \cdots x_nfy_1 \cdots y_n)$.

The notions of normal structure and normal satisfiability are defined as before. The results of Section 4.1 on predicate calculus with identity carry over unchanged to predicate calculus with identity and operations.

Remark 4.2.14 (predicate calculus with equality). The predicate calculus with identity and operations is well suited for the study of algebraic structures such as number systems, groups, rings, etc. In such a context, one often writes $t_1 = t_2$ instead of It_1t_2, and one refers to *predicate calculus with equality* rather than predicate calculus with identity. One also uses customary algebraic notation, e.g., $t_1 + t_2$ instead of $+t_1t_2$, $t_1 \times t_2$ or t_1t_2 instead of $\times t_1t_2$, etc. To avoid ambiguity, parentheses are used.

Examples 4.2.15 (groups and rings). Using predicate calculus with identity and operations, a group may be viewed as an L-structure

$$G = (U_G, f_G, i_G, e_G, I_G).$$

Here U_G is the underlying set of the group, and L is the language $\{f, i, e, I\}$, where f is the group composition law (a binary operation), i is group inversion (a unary operation), e is the group identity element (a 0-ary operation, i.e., a constant), and I is the identity predicate (a binary predicate). We could refer to L as *the language of groups*. It is customary to write G instead of U_G, $t_1 \cdot t_2$ or t_1t_2 instead of ft_1t_2, t^{-1} instead of it, 1 instead of e, and $t_1 = t_2$ instead of It_1t_2. Thus

$$G = (G, \cdot_G, {}^{-1}{}_G, 1_G, =_G)$$

and G is required to satisfy the *group axioms*, consisting of the *equality axioms* (i.e., the identity axioms for L) plus $\forall x \forall y \forall z\, ((xy)z = x(yz))$, $\forall x\, (x^{-1}x = xx^{-1} = 1)$, $\forall x\, (1x = x1 = x)$.

Similarly, a ring may be viewed as a structure

$$R = (R, +_R, \cdot_R, -_R, 0_R, 1_R, =_R)$$

where $+$ and $\cdot$ are binary operations, $-$ is a unary operation, 0 and 1 are constants, and $=$ is the equality predicate. We could refer to the language $\{+, \cdot, -, 0, 1, =\}$ as *the language of rings*. R is required to satisfy the *ring axioms*, consisting of the equality axioms (i.e., the identity axioms) plus $\forall x \forall y \forall y\,((x + y) + z = x + (y + z))$, $\forall x \forall y\,(x + y = y + x)$, $\forall x\,(x + 0 = x)$, $\forall x\,(x + (-x) = 0)$, $\forall x \forall y \forall z\,((x \cdot y) \cdot z = x \cdot (y \cdot z))$, $\forall x\,(x \cdot 1 = 1 \cdot x = x)$, $\forall x \forall y \forall z\,(x \cdot (y + z) = (x \cdot y) + (x \cdot z))$, $\forall x \forall y \forall z\,((x + y) \cdot z = (x \cdot z) + (y \cdot z))$.

Exercise 4.2.16. Let G be a group. For $a \in G$ write $a^n = a \cdot \cdots \cdot a$ (n times). Thus $a^1 = a$ and $a^{n+1} = a^n \cdot a$. We say that G is a *torsion group* if for all $a \in G$ there exists a positive integer n such that $a^n = 1$. We say that G is *torsion-free* if for all $a \in G$, if $a \neq 1$ then $a^n \neq 1$ for all positive integers n.

1. Show that the class of torsion-free groups can be characterized by a set of sentences. I.e., there is a set of sentences S such that, for all groups G, G is torsion-free if and only if G satisfies S.

2. Show that the class of torsion-free groups cannot be characterized by a finite set of sentences.

3. Show that the class of torsion groups cannot be characterized by a set of sentences. I.e., there is no set of sentences S with the property that, for all groups G, G is a torsion group if and only if G satisfies S.

Exercise 4.2.17. Show that the spectrum problem for predicate calculus with identity and operations is equivalent to the spectrum problem for predicate calculus with identity and without operations, as previously discussed in Section 4.1. In other words, given a sentence A involving some operations, construct a sentence A' involving no operations, such that $\text{spectrum}(A) = \text{spectrum}(A')$.

Hint: Replace each n-ary operation f by an $(n+1)$-ary predicate P_f where $P_f x_1 \ldots x_n y$ expresses $I f x_1 \ldots x_n y$.

4.3 Many-Sorted Predicate Calculus

Definition 4.3.1 (many-sorted languages). A *many-sorted language* consists of

1. a set of *sorts* $\sigma, \tau, \ldots$,

2. a set of predicates $P, Q, \ldots$, each designated as *n-ary of type*

$$\sigma_1 \times \cdots \times \sigma_n$$

   for some nonnegative integer n and sorts $\sigma_1, \ldots, \sigma_n$,

3. a set of operations $f, g, \ldots$, each designated as *n-ary of type*

$$\sigma_1 \times \cdots \times \sigma_n \to \sigma_{n+1}$$

for some nonnegative integer n and sorts $\sigma_1, \ldots, \sigma_n, \sigma_{n+1}$.

Let L be a many-sorted language.

Definition 4.3.2 (terms, formulas, sentences). For each sort σ, we assume a fixed, countably infinite set of *variables of sort* σ, denoted $x^\sigma, y^\sigma, z^\sigma, \ldots$. Let $U = (U^\sigma, U^\tau, \ldots)$ consist of a set U^σ for each sort σ of L. The *L-U-terms* are generated as follows.

1. Each variable of sort σ is a term of sort σ.
2. Each element of U^σ is a term of sort σ.
3. If f is an n-ary operaton of type $\sigma_1 \times \cdots \times \sigma_n \to \sigma_{n+1}$, and if $t_1, \ldots, t_n$ are terms of sort $\sigma_1, \ldots, \sigma_n$ respectively, then $ft_1 \ldots t_n$ is a term of sort σ_{n+1}.

An *atomic L-U-formula* is an expression of the form $Pt_1 \ldots t_n$, where P is an n-ary predicate of type $\sigma_1 \times \cdots \times \sigma_n$, and $t_1, \ldots, t_n$ are terms of sort $\sigma_1, \ldots, \sigma_n$ respectively. The *L-U-formulas* are generated as in Definition 2.1.3, with clause 5 modified as follows:

5'. If x^σ is a variable of sort σ, and if A is an L-U-formula, then $\forall x^\sigma\, A$ and $\exists x^\sigma\, A$ are L-U-formulas.

Our notions of substitution, free and bound variables, sentences, etc. are extended in the obvious way to the many-sorted context. Naturally, the substitution $A[x^\sigma/t]$ makes sense only when t is a term of sort σ. An *L-formula* is an L-U-formula where $U_\sigma = \emptyset$ for each sort σ.

Definition 4.3.3 (many-sorted structures). An *L-structure* M consists of

1. a nonempty set U_M^σ for each sort σ of L,
2. an n-ary relation $P_M \subseteq U_M^{\sigma_1} \times \cdots \times U_M^{\sigma_n}$ for each n-ary predicate P of type $\sigma_1 \times \cdots \times \sigma_n$ belonging to L,
3. an n-ary function $f_M : U_M^{\sigma_1} \times \cdots \times U_M^{\sigma_n} \to U_M^{\sigma_{n+1}}$ for each n-ary operation of type $\sigma_1 \times \cdots \times \sigma_n \to \sigma_{n+1}$ belonging to L.

Notions such as isomorphism, valuation, truth, satisfiability, and results such as Theorem 4.2.9 on onto mappings, carry over to the many-sorted context in the obvious way.

Definition 4.3.4 (many-sorted domains). We define a *domain* for L to be an indexed family of nonempty sets $U = (U^\sigma, U^\tau, \ldots)$, where $\sigma, \tau, \ldots$ are the sorts of L. In this way, the notion of satisfiability in a domain generalizes to the many-sorted context.

Remark 4.3.5 (tableau method, proof systems). For each sort σ of L, fix a countably infinite set $V^\sigma = \{a^\sigma, b^\sigma, \ldots\}$, the set of *parameters of sort* σ. Then the tableau method carries over in the obvious way, generalizing Remark 4.2.7. In the Completeness Theorem for the tableau method, we obtain satisfiability in the domain $U = (U^\sigma, U^\tau, \ldots)$, where U^σ is the set of variable-free L-V-terms, with $V = (V^\sigma, V^\tau, \ldots)$. The soundness and completeness of our proof systems LH and LG and the Interpolation Theorem also carry over, just as in Section 4.2.

Remark 4.3.6 (identity predicates). For each sort σ of L, L may or may not contain a binary predicate I^σ of type $\sigma \times \sigma$ designated as the *identity predicate* for σ. As *identity axioms* we may take the universal closures of all L-formulas of the form

$$\forall x^\sigma \, \forall y^\sigma \, (I^\sigma xy \Rightarrow (A \Leftrightarrow A[x/y]))$$

where A is atomic. An L-structure M is said to be *normal* if $I^\sigma_M = \{\langle a, a\rangle : a \in U^\sigma_M\}$ for all σ such that I^σ belongs to L. The results of Section 4.1 concerning normal satisfiability carry over to the many-sorted context. If A is an L-sentence and $\sigma_1, \ldots, \sigma_k$ are the sorts occurring in A, the *spectrum* of A is the set of k-tuples of positive integers $(n_1, \ldots, n_k)$ such that there exists a normal L-structure M with $U^{\sigma_i}_M$ of cardinality n_i, for $i = 1, \ldots, k$. In this way, the spectrum problem carries over to many-sorted predicate calculus.

Remark 4.3.7. Our reasons for including many-sorted predicate calculus in this course are as follows:

1. it is more useful ....

   FIXME

Chapter 5

Theories, Models, Definability

5.1 Theories and Models

Definition 5.1.1. A *theory* T consists of a language L, called the *language of* T, together with a set of L-sentences called the *axioms of* T.

Definition 5.1.2. If T is a theory, a *model of* T is a structure M for the language of T such that all of the axioms of T are true in M.

Definition 5.1.3. Let T be a theory with language L. A *theorem of* T is any L-sentence which is a logical consequence of the axioms of T. Equivalently, a theorem of T is any L-sentence which is is true in all models of T. Two theories are said to be *equivalent* if they have the same language and the same theorems.

Definition 5.1.4. A theory T is said to be *consistent* if there exists at least one model of T. Equivalently, T is consistent if the sentence F is not a theorem of T.

Definition 5.1.5. theories with identity, normal models

5.2 Mathematical Theories

Example 5.2.1 (groups).

Example 5.2.2 (rings).

Example 5.2.3 (fields).

Example 5.2.4 (vector spaces).

Example 5.2.5 (ordered structures).

5.3 Foundational Theories

Example 5.3.1 (first order arithmetic).

Example 5.3.2 (second order arithmetic).

Example 5.3.3 (set theory).

Remark 5.3.4 (practical completeness).

5.4 Definability over a Model

Definition 5.4.1. explicit definability over M

Definition 5.4.2. implicit definability over M

Exercise 5.4.3. Show that any relation which is explicitly definable over M is implicitly definable over M.

Exercise 5.4.4. Let $\mathbb{R} = (\mathbb{R}, +, -, \cdot, 0, 1, <, =)$ be the ordered field of real numbers. Show that the relations $y = \sin x$ and $y = e^x$ are implicitly definable over $\mathbb{R}$. It can be shown that these relations are not explicitly definable over $\mathbb{R}$.

Exercise 5.4.5. Show that if M is saturated, then any relation which is implicitly definable over M is explicitly definable over M.

Remark 5.4.6. Beth Definability Theorem

5.5 Definitional Extensions of Theories

Theorem 5.5.1 (adding a new predicate).

Theorem 5.5.2 (adding a new operation).

Definition 5.5.3. We say that T_1 is *interpretable in* T_2 if T_1 is a subtheory of a definitional extension of T_2.

Example 5.5.4. First order arithmetic is interpretable in second order arithmetic. First and second order arithmetic are interpretable in set theory.

5.6 The Beth Definability Theorem

Theorem 5.6.1 (the Beth Definability Theorem).

Proof. This is an application of the Interpolation Theorem, Section 3.5. □

Chapter 6

Arithmetization of Predicate Calculus

6.1 Primitive Recursive Arithmetic

Definition 6.1.1. To each natural number n we associate a PRA-term $\underline{n}$ as follows: $\underline{0} = 0$, $\underline{n+1} = \underline{S}(\underline{n})$. These terms are known as *numerals*.

Theorem 6.1.2. Let f be a k-ary primitive recursive function. Then for all k-tuples of natural numbers $m_1, \ldots, m_k$ we have

$$\mathsf{PRA} \vdash \underline{f}\,\underline{m_1}\cdots\underline{m_k} = \underline{f(m_1, \ldots, m_k)}\ .$$

Proof. □

6.2 Interpretability of PRA in Z_1

6.3 Gödel Numbers

Let L be a countable language. Assume that to all the sorts σ, predicates P, and operations f of L have been assigned distinct positive integers $\#(\sigma)$, $\#(P)$, $\#(f)$ respectively.

Definition 6.3.1 (Gödel numbers). To each term t and formula A of L we assign a positive integer, the *Gödel number* of t or of A, denoted $\#(t)$ or $\#(A)$,

respectively.

$$
\begin{array}{lcll}
\#(v_i^\sigma) & = & 2 \cdot 3^{\#(\sigma)} \cdot 5^i & \\
\#(a_i^\sigma) & = & 2^2 \cdot 3^{\#(\sigma)} \cdot 5^i & \\
\#(f t_1 \cdots t_n) & = & 2^3 \cdot 3^{\#(f)} \cdot p_2^{\#(t_1)} \cdots p_{n+1}^{\#(t_n)} & \text{if } f \text{ is an } n\text{-ary operation} \\
\#(P t_1 \cdots t_n) & = & 2^4 \cdot 3^{\#(P)} \cdot p_2^{\#(t_1)} \cdots p_{n+1}^{\#(t_n)} & \text{if } P \text{ is an } n\text{-ary predicate} \\
\#(\neg A) & = & 2^5 \cdot 3^{\#(A)} & \\
\#(A \,\&\, B) & = & 2^6 \cdot 3^{\#(A)} \cdot 5^{\#(B)} & \\
\#(A \vee B) & = & 2^7 \cdot 3^{\#(A)} \cdot 5^{\#(B)} & \\
\#(A \Rightarrow B) & = & 2^8 \cdot 3^{\#(A)} \cdot 5^{\#(B)} & \\
\#(A \Leftrightarrow B) & = & 2^9 \cdot 3^{\#(A)} \cdot 5^{\#(B)} & \\
\#(\forall v\, A) & = & 2^{10} \cdot 3^{\#(v)} \cdot 5^{\#(A)} & \\
\#(\exists v\, A) & = & 2^{11} \cdot 3^{\#(v)} \cdot 5^{\#(A)} &
\end{array}
$$

Definition 6.3.2. The language L is said to be *primitive recursive* if the following items are primitive recursive.

$$
\begin{array}{lcll}
\mathrm{Sort}(x) & \equiv & x = \#(\sigma) & \text{for some sort } \sigma \\
\mathrm{Pred}(x) & \equiv & x = \#(P) & \text{for some predicate } P \\
\mathrm{Op}(x) & \equiv & x = \#(f) & \text{for some operation } f \\
\mathrm{arity}(\#(P)) & = & n & \text{if } P \text{ is an } n\text{-ary predicate} \\
\mathrm{arity}(\#(f)) & = & n & \text{if } f \text{ is an } n\text{-ary operation} \\
\mathrm{sort}(\#(P), i) & = & \#(\sigma_i) & \text{if } 1 \le i \le n \text{ and } P \text{ is an } n\text{-ary predicate of type } \sigma_1 \times \cdots \times \sigma_n \\
\mathrm{sort}(\#(f), i) & = & \#(\sigma_i) & \text{if } 1 \le i \le n+1 \text{ and } f \text{ is an } n\text{-ary operation of type } \sigma_1 \times \cdots \times \sigma_n \to \sigma_{n+1}
\end{array}
$$

Lemma 6.3.3. If L is primitive recursive, then the following are primitive recursive.

$$
\begin{array}{lcll}
\mathrm{Var}(x) & \equiv & x = \#(v) & \text{for some variable } v \\
\mathrm{Param}(x) & \equiv & x = \#(a) & \text{for some parameter } a \\
\mathrm{Term}(x) & \equiv & x = \#(t) & \text{for some term } t \\
\mathrm{ClTerm}(x) & \equiv & x = \#(t) & \text{for some closed term } t \\
\mathrm{AtFml}(x) & \equiv & x = \#(A) & \text{for some atomic formula } A \\
\mathrm{Fml}(x) & \equiv & x = \#(A) & \text{for some formula } A \\
\mathrm{sort}(\#(t)) & = & \#(\sigma) & \text{if } t \text{ is a term of sort } \sigma
\end{array}
$$

Proof. We have

$$
\mathrm{Var}(x) \quad \equiv \quad (x)_0 = 1 \;\&\; x = 2^{(x)_0} \cdot 3^{(x)_1} \cdot 5^{(x)_2} \;\&\; \mathrm{Sort}((x)_1)
$$

and

$$\mathrm{Param}(x) \quad \equiv \quad (x)_0 = 2 \;\&\; x = 2^{(x)_0} \cdot 3^{(x)_1} \cdot 5^{(x)_2} \;\&\; \mathrm{Sort}((x)_1) \; .$$

To show that the predicate $\mathrm{Term}(x)$ is primitive recursive, we first show that the function $\mathrm{sort}(x)$ is primitive recursive, where $\mathrm{sort}(\#(t)) = \#(\sigma)$ if t is a term of sort σ, $\mathrm{sort}(x) = 0$ otherwise. Put $\mathrm{lh}(x) =$ least $w < x$ such that $(x)_w = 0$. We then have

$$\mathrm{sort}(x) \quad = \quad \begin{cases} (x)_1 & \text{if } \mathrm{Var}(x) \vee \mathrm{Param}(x) \, , \\ \mathrm{sort}((x)_1, \mathrm{lh}(x) \dot{-} 1) & \text{if } (x)_0 = 3 \;\&\; \mathrm{Op}((x)_1) \;\&\; (+) \, , \\ 0 & \text{otherwise} \, , \end{cases}$$

where

$$(+) \qquad \mathrm{arity}((x)_1) = \mathrm{lh}(x) \dot{-} 2 \;\; \& \;\; x = \textstyle\prod_{i=0}^{\mathrm{lh}(x) \dot{-} 1} p_i^{(x)_i}$$
$$\& \;\; (\forall i < \mathrm{lh}(x) \dot{-} 2) \; (\mathrm{sort}((x)_{i+2}) = \mathrm{sort}((x)_1, i+1)) \; .$$

Then

$$\mathrm{Term}(x) \quad \equiv \quad \mathrm{sort}(x) > 0 \; .$$

For closed terms, define $\mathrm{clsort}(x)$ like $\mathrm{sort}(x)$ replacing $\mathrm{Var}(x) \vee \mathrm{Param}(x)$ by $\mathrm{Param}(x)$. We then have

$$\mathrm{ClTerm}(x) \quad \equiv \quad \mathrm{clsort}(x) > 0 \; .$$

For formulas we have

$$\mathrm{AtFml}(x) \quad \equiv \quad ((x)_0 = 4 \;\&\; \mathrm{Pred}((x)_1) \;\&\; (+))$$

and

$$\begin{aligned} \mathrm{Fml}(x) \quad \equiv \quad & \mathrm{AtFml}(x) \vee ((x)_0 = 5 \;\&\; x = 2^{(x)_0} \cdot 3^{(x)_1} \;\&\; \mathrm{Fml}((x)_1)) \\ & \vee \; (6 \leq (x)_0 \leq 9 \;\&\; x = 2^{(x)_0} \cdot 3^{(x)_1} \cdot 5^{(x)_2} \;\&\; \mathrm{Fml}((x)_1) \;\&\; \mathrm{Fml}((x)_2)) \\ & \vee \; (10 \leq (x)_0 \leq 11 \;\&\; x = 2^{(x)_0} \cdot 3^{(x)_1} \cdot 5^{(x)_2} \;\&\; \mathrm{Var}((x)_1) \;\&\; \mathrm{Fml}((x)_2)) \end{aligned}$$

and this completes the proof. □

Lemma 6.3.4 (substitution). There is a primitive recursive function $\mathrm{sub}(x, y, z)$ such that for any formula A and any variable v and any closed term t,

$$\mathrm{sub}(\#(A), \#(v), \#(t)) \quad = \quad \#(A[v/t]) \; .$$

Proof.

$$\mathrm{sub}(x, y, z) \quad = \quad \begin{cases} z & \text{if } x = y \\ 2^{(x)_0} \cdot 3^{(x)_1} \cdot \prod_{i=2}^{\mathrm{lh}(x) \dot{-} 1} p_i^{\mathrm{sub}((x)_i, y, z)} & \text{if } 3 \leq (x)_0 \leq 4 \\ 2^{(x)_0} \cdot 3^{\mathrm{sub}((x)_1, y, z)} & \text{if } (x)_0 = 5 \\ 2^{(x)_0} \cdot 3^{\mathrm{sub}((x)_1, y, z)} \cdot 5^{\mathrm{sub}((x)_2, y, z)} & \text{if } 6 \leq (x)_0 \leq 9 \\ 2^{(x)_0} \cdot 3^{(x)_1} \cdot 5^{\mathrm{sub}((x)_2, y, z)} & \text{if } 10 \leq (x)_0 \leq 11 \,\&\, (x)_1 \neq y \\ x & \text{otherwise.} \end{cases}$$

□

Lemma 6.3.5. If L is primitive recursive, then the predicate

$$\mathrm{Snt}(x) \;\equiv\; x = \#(A) \text{ for some sentence } A$$

is primitive recursive.

Proof. Recall that, by Exercise 2.1.10, a formula A is a sentence if and only if $A[v/a] = A$ for all variables v occurring in A. Note also that if $y = \#(v_i^\sigma)$ then $2y = \#(a_i^\sigma)$. Thus we have

$$\mathrm{Snt}(x) \;\equiv\; \mathrm{Fml}(x) \;\&\; (\forall y < x)\,(\mathrm{Var}(y) \Rightarrow x = \mathrm{sub}(x, y, 2y)) \;.$$

□

Lemma 6.3.6. There is a primitive recursive function $\mathrm{num}(x)$ such that

$$\mathrm{num}(n) \;=\; \#(\underline{n})$$

for any nonnegative integer n.

Proof. The recursion equations for $\mathrm{num}(x)$ are

$$\begin{aligned} \mathrm{num}(0) &= \#(\underline{0}) \;, \\ \mathrm{num}(x+1) &= 2^3 \cdot 3^{\#(\underline{S})} \cdot 5^{\mathrm{num}(x)} \;. \end{aligned}$$

□

6.4 Undefinability of Truth

In this section, let T be a theory which includes PRA. For example, we could take T to be PRA itself. Or, by Section 6.2, we could take T to be an appropriate definitional extension of Z_1 or Z_2 or ZFC.

Lemma 6.4.1 (Self-Reference Lemma). Let L be the language of T. Let $A(x)$ be an L-formula with a free number variable x. Then we can find an L-formula B such that

$$T \;\vdash\; B \Leftrightarrow A(\underline{\#(B)}) \;.$$

The free variables of B are those of $A(x)$ except for x. In particular, if x is the only free variable of A, then B is an L-sentence.

Proof. Put $d(z) = \mathrm{sub}(z, \#(x), \mathrm{num}(z))$. Thus d is a 1-ary primitive recursive function such that, if $A(x)$ is any L-formula containing the number variable x as a free variable, then $d(\#(A)) = \#(A(\underline{\#(A)}))$. Now given $A(x)$ as in the hypothesis of the lemma, let $D(x)$ be the formula $A(\underline{d}(x))$, and let B be the formula $A(\underline{d}(\#(D)))$, i.e., $D(\underline{\#(D)})$. Note that $d(\#(D)) = \#(B)$. It follows that $\mathsf{PRA} \vdash \underline{d}(\underline{\#(D)}) = \underline{\#(B)}$. Since T includes PRA, it follows that $T \vdash \underline{d}(\underline{\#(D)}) = \underline{\#(B)}$. Hence $T \vdash A(\underline{d}(\underline{\#(D)})) \Leftrightarrow A(\underline{\#(B)})$. In other words, $T \vdash B \Leftrightarrow A(\underline{\#(B)})$. This completes the proof. □

Definition 6.4.2. If M is any model of T, let True_M be the set of Gödel numbers of sentences that are true in M, i.e.,

$$\mathrm{True}_M \;=\; \{\#(B) : B \text{ is a sentence and } v_M(B) = \mathrm{T}\}\ .$$

Definition 6.4.3. An *ω-model of T* is a model M of T such that the number domain of M is $\omega = \{0, 1, 2, \ldots\}$ and $0_M = 0$ and $S_M(n) = n + 1$ for all $n \in \omega$. More generally, if M is any model of T, we may assume that ω is identified with a subset of the number domain of M in such a way that $0_M = 0$ and $S_M(n) = n + 1$ for each $n \in \omega$. Thus each $n \in \omega$ is identified with with the element of M that is denoted by $\underline{n}$, i.e., $n = v_M(\underline{n})$.

Theorem 6.4.4 (undefinability of truth). If M is an ω-model of T, then True_M is not explicitly definable over M. More generally, if M is any model of T, then the characteristic function of True_M is is not included in the characteristic function of any subset of M that is explicitly definable over M.

Proof. Let X be a subset of the number domain of M which is explicitly definable over M. Let $A(x)$ be an L-formula with a free number variable x and no other free variables, such that $A(x)$ explicitly defines X over M. Applying Lemma 6.4.1 to the negation of $A(x)$, we obtain an L-sentence B such that $T \vdash B \Leftrightarrow \neg\, A(\underline{\#(B)})$. Since M is a model of T, it follows that $\#(B) \in \mathrm{True}_M$ if and only if $\overline{\#(B)} \notin X$. Hence the characteristic function of True_M is not included in the characteristic function of X, q.e.d. □

Corollary 6.4.5. Let $M = (\omega, +, \cdot, 0, 1, =)$ be the standard model of first order arithmetic, Z_1. Then True_M is not explicitly definable over M. This[1] may be paraphrased by saying that arithmetical truth is not arithmetically definable.

Remark 6.4.6. With $M = (\omega, +, \cdot, 0, 1, =)$ as above, it can be shown that True_M is implicitly definable over M. (See also Exercise 6.4.7.) Thus the Beth Definability Theorem does not hold for definability over this particular model.

Exercise 6.4.7. Let M be an ω-model of Z_1 or Z_2 or ZFC. Let Sat_M be the satisfaction relation on M. Show that Sat_M is implicitly definable over M.

6.5 The Provability Predicate

In this section, let L be a primitive recursive language, and let T be an L-theory which is primitive recursively axiomatizable. For example, T could be PRA itself, or T could be any of the mathematical or foundational theories discussed in Sections 5.2 and 5.3.

Definition 6.5.1. Choose a primitive recursive predicate Ax_T for the set of Gödel numbers of axioms of T. In terms of Ax_T show that various predicates

[1] This result is due to Tarski [3].

associated with T are primitive recursive. Introduce the provability predicate Pvbl_T by definition:

$$\mathrm{Pvbl}_T(x) \;\Leftrightarrow\; \exists y\, \mathrm{Prf}_T(x, y) \ .$$

Note that, for all L-sentences B, $\mathrm{Pvbl}_T(\underline{\#(B)})$ is true if and only if $T \vdash B$.

Lemma 6.5.2 (derivability condition 1). For any L-sentence A, if $T \vdash A$ then

$$\mathsf{PRA} \;\vdash\; \mathrm{Pvbl}_T(\underline{\#(A)}) \ .$$

Proof. Suppose $T \vdash A$. Let p be a proof of A in T. Then we have $\mathrm{Prf}_T(\#(A), \#(p))$. Since $\mathrm{Prf}_T(x, y)$ is a primitive recursive predicate, it follows by Theorem 6.1.2 that $\mathsf{PRA} \vdash \mathrm{Prf}_T(\underline{\#(A)}, \underline{\#(p)})$. Hence $\mathsf{PRA} \vdash \mathrm{Pvbl}_T(\underline{\#(A)})$, q.e.d. □

Lemma 6.5.3 (derivability condition 2). For any L-sentence A, we have

$$\mathsf{PRA} \;\vdash\; \mathrm{Pvbl}_T(\underline{\#(A)}) \Rightarrow \mathrm{Pvbl}_{\mathsf{PRA}}(\underline{\#(\mathrm{Pvbl}_T(\underline{\#(A)}))}) \ .$$

Proof. This is just Lemma 6.5.2 formalized in PRA. The details of the formalization are in Section 6.7. □

Lemma 6.5.4 (derivability condition 3). For any L-sentences A and B, we have

$$\mathsf{PRA} \;\vdash\; \mathrm{Pvbl}_T(\underline{\#(A \Rightarrow B)}) \Rightarrow (\mathrm{Pvbl}_T(\underline{\#(A)}) \Rightarrow \mathrm{Pvbl}_T(\underline{\#(B)})) \ .$$

Proof. This is a straightforward consequence of the fact that our rules of inference include modus ponens. □

6.6 The Incompleteness Theorems

In this section, let T be a theory which is primitive recursively axiomatizable and includes PRA. For example, T could be PRA itself, or it could be an appropriate definitional extension of Z_1 or Z_2 or ZFC. As in Section 6.5, let Pvbl_T be a provability predicate for T.

Lemma 6.6.1. Let $A(x)$ be a PRA-formula with one free variable x. Then we can find a PRA-sentence B such that $\mathsf{PRA} \vdash B \Leftrightarrow A(\underline{\#(B)})$.

Proof. This is the Self-Reference Lemma 6.4.1 specialized to PRA. □

Lemma 6.6.2. We can find a PRA-sentence S such that

$$(*) \qquad \mathsf{PRA} \;\vdash\; S \Leftrightarrow \neg\, \mathrm{Pvbl}_T(\underline{\#(S)}) \ .$$

Note that S is self-referential and says "I am not provable in T."

Proof. This is an instance of Lemma 6.6.1 with $A(x) \equiv \neg\, \mathrm{Pvbl}_T(x)$. □

Lemma 6.6.3. Let S be as in Lemma 6.6.2. If T is consistent, then $T \nvdash S$.

Proof. Suppose for a contradiction that $T \vdash S$. By Lemma 6.5.2 we have $\mathsf{PRA} \vdash \mathrm{Pvbl}_T(\underline{\#(S)})$. Hence by $(*)$ it follows that $\mathsf{PRA} \vdash \neg S$. Since T includes PRA, we have $T \vdash \neg S$. Thus T is inconsistent. □

Theorem 6.6.4 (the First Incompleteness Theorem). If T is consistent, then we can find a sentence S' in the language of first order arithmetic such that S' is true yet S' is not a theorem of T.

Proof. Let S be a PRA-sentence as in Lemma 6.6.2. By Lemma 6.6.3, $T \nvdash S$. It follows by $(*)$ that S is true. As in Section 6.2, let S' be the translation of S into the language of first order arithmetic. Thus S' is also true. By the results of Section 6.2, $\mathsf{PRA} \vdash S \Leftrightarrow S'$. Hence $T \vdash S \Leftrightarrow S'$. Hence $T \nvdash S'$. □

Assume now that the primitive recursive predicate Ax_T has been chosen in such a way that $\mathsf{PRA} \vdash \forall x\,(\mathrm{Ax}_{\mathsf{PRA}}(x) \Rightarrow \mathrm{Ax}_T(x))$. It follows that $\mathsf{PRA} \vdash \forall x\,(\mathrm{Pvbl}_{\mathsf{PRA}}(x) \Rightarrow \mathrm{Pvbl}_T(x))$. In particular we have:

Lemma 6.6.5. For all PRA-sentences A, we have

$$\mathsf{PRA} \;\vdash\; \mathrm{Pvbl}_{\mathsf{PRA}}(\underline{\#(A)}) \Rightarrow \mathrm{Pvbl}_T(\underline{\#(A)})\ .$$

Definition 6.6.6. Con_T is defined to be the sentence $\neg\,\mathrm{Pvbl}_T(\underline{\#(\mathrm{F})})$. Here F is the identically false sentence. Note that Con_T is a PRA-sentence which asserts the consistency of T.

Theorem 6.6.7 (the Second Incompleteness Theorem). If T is consistent, then $T \nvdash \mathrm{Con}_T$.

Proof. Let S be as in Lemma 6.6.2. By Theorem 6.6.4 we have $T \nvdash S$. Therefore, to show $T \nvdash \mathrm{Con}_T$, it suffices to show $T \vdash \mathrm{Con}_T \Rightarrow S$. Since T includes PRA, it suffices to show $\mathsf{PRA} \vdash \mathrm{Con}_T \Rightarrow S$. By $(*)$ it suffices to show

$$(**) \qquad \mathsf{PRA} \;\vdash\; \mathrm{Con}_T \Rightarrow \neg\,\mathrm{Pvbl}_T(\underline{\#(S)})\ .$$

But this is just Lemma 6.6.3 formalized in PRA.

Details: We need to prove $(**)$. Reasoning in PRA, suppose $\mathrm{Pvbl}_T(\underline{\#(S)})$. By Lemma 6.5.3 we have $\mathrm{Pvbl}_{\mathsf{PRA}}(\underline{\#(\mathrm{Pvbl}_T(\underline{\#(S)}))})$. Moreover, from $(*)$ and Lemma 6.5.2 we have $\mathrm{Pvbl}_{\mathsf{PRA}}(\underline{\#(S \Leftrightarrow \neg\,\mathrm{Pvbl}_T(\underline{\#(S)}))})$. Hence by Lemmas 6.5.2 and 6.5.4 we have $\mathrm{Pvbl}_{\mathsf{PRA}}(\underline{\#(\neg\, S)})$. By Lemma 6.6.5 it follows that $\mathrm{Pvbl}_T(\underline{\#(\neg\, S)})$. Hence by Lemmas 6.5.2 and 6.5.4 it follows that $\mathrm{Pvbl}_T(\underline{\#(\mathrm{F})})$, i.e., $\neg\,\mathrm{Con}_T$. This completes the proof. □

Exercise 6.6.8. Show that $\mathsf{PRA} \vdash S \Leftrightarrow \mathrm{Con}_T$.

6.7 Proof of Lemma 6.5.3

FIXME Write this section!

Bibliography

[1] Neil D. Jones and Alan L. Selman. Turing machines and the spectra of first-order formulas. *Journal of Symbolic Logic*, 39:139–150, 1974.

[2] Raymond M. Smullyan. *First-Order Logic*. Dover Publications, Inc., 1995. XII + 158 pages.

[3] Alfred Tarski. *Introduction to Logic and to the Methodology of Deductive Sciences*. Oxford University Press, New York, 4th edition, 1994. XXII + 229 pages.

Index