Logic for Computer Science (234292)

Prof. J.A. Makowsky
Department of Computer Science
Technion—Israel Institute of Technology, Haifa, Israel

October 6, 1997

Abstract

Draft of a textbook on Logic in Computer Science. The book is based on the courses Logic 1, Logic 2 and Definability and Computability, as taught by the author during several years at the Computer Science Department at the Technion—Israel Institute of Technology. Comments welcome.

This version is used for the course 'Logic 1' (234292) given in the

winter semester 1997/8

at the Computer Science Department of the Technion–Israel Institute of Technology. It was also used in 1990-1996 by the same lecturer. Previous versions are still useful.

The lecturers are

Dr. Y. Kimchi (yechiel@CS.Technion.AC.IL)
Prof. J. Makowsky (janos@cs.technion.ac.il)

The coordinator for Tirgul is

L. Ravve (cselena@cs.technion.ac.il)

The lecture takes place on

Wednesday, 14:30-16:30

Details of Tirgulim, Reception hours etc. will be posted at the home-page of the course

http://www.cs.technion.ac.il/ janos/COURSES/LOGIC/logic1.html

There are seven home assignments (H), a mid–term exam (M) and a final exam (F). The exams are with open books. The six best home assignements count. The final grade is computed by the formula

$$max\{F, (F+M)/2, (F+M+H)/3, 2F+H/3\}.$$

Contents

1 Introduction 4
2 The World of Sets 7
2.1 Sets and operations on sets 7
2.2 Relations and functions 10
2.3 Inductive Definitions and Proofs 12
2.4 Some more sets 14
2.5 Implementing Datastructures as sets 17
2.5.1 Words 17
2.5.2 Trees 18
2.5.3 Natural numbers and its arithmetic operations . . . 19
2.6 Equipotence 21
3 Propositional Logic 27
3.1 The Syntax and Semantics of Propositional Logic 27
3.1.1 Syntax of Propositional Logic 27
3.1.2 The Truth Table Semantics of Propositional Logic . 30
3.2 Basic Semantic Concepts 33
3.2.1 Validity, Logical Equivalence and Logical Consequence 33
3.2.2 Substitution 37
3.2.3 Normal Forms 38
3.3 Deduction Methods and Compactness 40
3.3.1 Proof Sequences 40
3.3.2 Manipulations of Proof Seequences 42
3.3.3 Completeness and Compactness 43
3.3.4 Resolution 45
3.4 Compactness 49
3.4.1 A Semantic Proof of Compactness 49
3.4.2 Applications of Compactness: Definability 50
3.4.3 Truth Table Extensions of Propositional Logic . . . 52
4 First Order Logic 55
4.1 Vocabularies and Structures 56
4.1.1 Vocabularies 56
4.1.2 Interpretations of Vocabularies 57
4.1.3 Isomorphisms and Substructures * 58
4.2 A Museum of Structures 61
4.2.1 Structures for Arithmetic 61
4.2.2 Graphs and Orders 63
4.2.3 Words and Sets of Words as Structures 64
4.2.4 Data Structures of Computer Science 65
4.2.5 $\in$–Structures 66
4.3 Syntax and Semantics of First Order Logic 66
4.3.1 Syntax of First Order Logic 67

4.3.2 Semantics of First Order Logic 70
4.4 Basic Semantic Concepts 72
4.4.1 Validity, Logical Equivalence and Logical Consequence 72
4.4.2 Normal Forms 75
4.4.3 Models and Theories * 76
4.5 Visit to the Museum: The Meaning Function and Definability 76
4.5.1 The Logical Lieu 77
4.5.2 Ordered Fields 77
4.5.3 The Natural Numbers 79
4.5.4 Graphs and Orders 80
4.5.5 Words and Sets of Words as Structures 80
4.5.6 Data Structures of Computer Science 81
4.5.7 $\in$–Structures 82
4.5.8 Substructures 82
4.6 Hilbert–style Deduction for First Order Logic 83
4.6.1 Hilbert-style Axioms and Inference Rules 83
4.6.2 Manipulations of Proof Sequences 85
4.6.3 Completeness and Compactness 87
4.6.4 Proof of the Completeness Theorem 88
4.6.5 The Case with Equality 90
4.7 Visit to the Museum: Non–Definability 90
4.7.1 Finite Structures 90
4.7.2 The Real Numbers 91
4.7.3 The Natural Numbers 92
4.8 Unification and Resolution 92

1 Introduction

The purpose of this book is to analyze in a mathematical way the syntax and semantics of formal languages for the specification of data structures, for programming, and for reasoning about programs. For this purpose we model all the notions introduced in the world of sets and use the natural language of set theoretic mathematics, also known as Naive Set Theory. We introduce set theoretic concepts whenever they are needed and concentrate on the themes of syntax and semantics. The first formal language discussed in detail is propositional logic. The treatment of propositional logic is dictated by what we need later, and is not a goal by itself. Methodologically, it serves as a paradigm for treating such a formal language by mathematical, i.e., set theoretic, means. Substantially it serves us well to introduce the basic concepts and questions we want to ask about such formal languages. It also serves as a basis for further arguments. The main emphasis of the first part, the course Logic 1, is on first order logic, culminating in Goedel's Completeness Theorem and Ehrenfeucht's classification of first order definability. The second part, the course Logic 2, concentrates on the limitations of first order logic, such as Incompleteness Theorems and the theorems of Lindstrom, and on various extensions of propositional and first order logic such as Modal Logic, Dynamic Logic, and Temporal Logic.

As the emphasis of the book is on the *mathematical treatment* of topics in syntax and semantics, a word is needed about how to read this book. Beginning computer science students very often lack *performance and proficiency* in mathematical arguments. They often complain about the scarcity of examples in the mathematical treatment of a subject, and confuse the abundancy of examples with the soundness of an argument. Reading a mathematical text is an *exploratory* activity which involves also paper and pencil. When encountering a definition the reader should be able to explore the definition by *providing examples and counterexamples actively.* Very often simple conclusion and explorations of the definitions phrased as *propositions* are really encapsulated formulations of *infinitely many examples.* To stress this point, our text very often has examples and propositions, lemmas, corollaries, marked as exercises. The intended meaning of this, and a *conditio sine qua non* for the understanding of the presented material, is that the student really performs these required exercises proficiently. Printing the solution to these exercises is not only a waste of paper, but invites the reader to cheat himself, by glancing through the solution rather than resorting to paper and pencil. Proficiency in mathematical arguments is a means of emancipating oneself from beliefs, prejudice, and superstition. As R. Godement has put it in his *Cours d'Algebre* in 1966:

Even in teaching mathematics one can at least attempt to teach the students the flavour of freedom and critical thought, and to get them used to

the idea of being treated as humans empowered with the ability of understanding.

What I would like to add, as Godement took it for granted, but forgot to mention it, is that I also treat the reader as a human being who is *willing* to use this very ability, not only in the limited context of performing the exercises of this book, but in his whole attitude towards life. Science, as the logician W.Quine put it, is *selfconscious common sense.* And so are logic and all the activities which are related to programming machines to perform tasks previously reserved to humans.

The following is a discussion of related textbooks. The first five are classics of mathematical logic (written for the mathematically minded) without an outlook to computer science. The last three are texts for beginning computer scientists.

(i) E.Mendelson, Introduction to mathematical logic, Van Nostrand, 1964.
This is an excellent, though oldfashioned, book on mathematical logic, suitable as a first course for mathematics students.

(ii) R.C. Lyndon, Notes on logic, Van Nostrand, 1964.
Very elegant and concise classic text. Written for mathematicians with some algebraic background. Our notion of proof sequence is taken from here.

(iii) H. B. Enderton, A mathematical introduction to logic, Academic Press, 1972.
A very elegant treatment of the basic subjects of elementary logic. Very precise and still intuitive in its motivation of the basic concepts.

(iv) E.D. Ebbinghaus, J.Flum and W. Thomas, Mathematical Logic, Springer, 1984.
The state of the art introductory text on first order logic, written for mathematicians. Covers all relevant basic developments in first order logic as seen today.

(v) J. Malitz, Introduction to mathematical logic, Springer, 1979.
A very elegant and modern treatment of the basic subjects of logic: Naive Set Theory, Computability and First Order Logic. Written as an introduction for mathematicians. In spirit very similar to our approach.

(vi) P. Halmos, Naive set theory, Van Nostrand, 1960.
This is the classic *introductory course in Naive Set Theory. Recommended as background reading.*

(vii) Z. Manna, Mathematical theory of computation, McGraw-Hill, 1974.
This was one of the first books on logic in computer science. The

presentation is less rigorous than our treatment. Recommended as a background reading.

(viii) M. Wand, Induction, Recursion and Programming, North Holland, 1980.
Here the intended audience are students of computer science. The approach is similar to our treatment, but the material presented is chosen with an emphasis on programming languages.

(ix) H.R. Lewis and C.H. Papadimitriou, Elements of the theory of computation, Prentice Hall, 1981.
This book combines material of three courses: Discrete Mathematics, Automata Theory and Computability and Logic 1 for Computer Scientists. The last two chapters basically cover the material of our course.
More references will be given at later stages.

Acknowledgments

I would like to thank my previous assistants Y. Bargury, E. Dichterman, R. A. Hason, E. Ravve (Mouratova) and A. Sharell for their critical reading and valuable suggestions during the preparation of this text. I would like to thank A. Ben–Ephraim, B. Farison, L. Finkelshtein, G. Granot, Y. Pnueli and E. Roytman whose notes of my course Logic 2 I could use as a basis for some sections.

I am also indebted to my colleagues S. Ben–David, N. Francez and A. Litman for their interest and advice.

2 The World of Sets

In this chapter we introduce the (natural, mathematical) language of sets. We expect the reader to be familiar with sets and functions as taught in a course on Discrete Mathematics, Calculus, Number Systems or alike. We also assume the reader is vaguely familiar with data structures such as graphs and trees. One purpose of this chapter is to fix notation. But, more importantly, the main purpose of this chapter is to get the reader acquainted with the handling of mathematical objects which are well–defined as sets of some kind. We are *not* dealing here with *set theory*. We hope the reader will learn the *language of sets* naïvely on the way, as one may learn a language just by living among speakers of that language und using it. The correct use of the language of sets is sometimes called (misleadingly) *Naïve Set Theory*. There is no *theory* here in the sense that we will not reflect upon the foundation of the language of sets. We shall use it critically, as a style of reasoning. We shall express our mathematical reasoning in it and we shall model our objects under study in it, be it formal languages, computing devices or any other object we wish to study mathematically. We shall learn the language of sets by using it. The heading 'Proposition–Exercise' is given to a statement (or collection of statements) which we want to exhibit, but which the reader should prove for himself, before proceeding further.

2.1 Sets and operations on sets

We first introduce some sets by well known examples. Let $\mathbf{N}$ be the set of natural numbers including 0, $\mathbf{N}^+$ be the set of natural numbers without 0, $\mathbf{Q}$ the set of rational numbers, $\mathbf{R}$ the set of real numbers, $\mathbf{C}$ the set of complex numbers. All these sets are infinite. $\emptyset$ denotes the empty set. The empty set is finite.

We write $a \in A$ for the statement *'a is an element of A'*. This statement is only meaningful provided A is a set. We write $A \subseteq B$ if A, B are sets and for every $a \in A$ it is true that $a \in B$. A is called a subset of B. If A, B are sets, we write $A = B$ if $A \subseteq B$ and $B \subseteq A$ and we say that A equals B. If A is a set (or a well defined object), then $\{A\}$ denotes the set whose only element is A.

Let a, b, c be letters of the alphabet. We denote by $\{a, b, c\}$ the set having exactly a, b, c as its elements. We consider letters as atoms (urelements), i.e. entities which are *not sets,* but which can be elements of sets. We shall treat all ASCI-symbols, letters of the greek alphabet, and possibly other symbols as such atoms. In particular, if X is a symbol, by abuse of notation, we may denote by $\{X_i : i \in \mathbf{N}\}$ an infinite set of atoms X_i.

Let A be a finite set of atoms. We denote by $(A)^*$ the set of finite words (strings) over A. We identify A with the set of one letter words over

A and therefore have that $A \subseteq (A)^*$. We denote by ϵ the empty word. If $a, b \in (A)^*$ are two words, we denote by $a \circ b$ the word obtained from writing b after a. If the context is clear we also write ab instead of $a \circ b$.

We allow explicit description of sets. If a, b, c are atoms and A, B are sets, then $\{a, A, b, B\}$ denotes the set having exactly a, b, A, B as its elements. $\{a, B, \{c, B\}\}$ is a set with three elements, namely a, B and the set $\{c, B\}$. More generally, if A is a set and Φ is a statement about elements x of A, then $\{x \in A : \Phi(x)\}$ denotes the subset of A whose elements are exactly those elements of A for which the statement Φ is true.

If A, B are sets we denote by $A \cap B$ the set whose elements are exactly those a such that $a \in A$ and $a \in B$. $A \cap B$ is called the intersection of A and B. If A, B are sets we denote by $A \cup B$ the set whose elements are exactly those a such that $a \in A$ or $a \in B$. $A \cup B$ is called the union of A and B.

2.1.1: Proposition–Exercise
Verify the following statements:

(i) $\emptyset \subseteq A$ *for every set* A;

(ii) $A \subseteq A$ *(Reflexivity of inclusion);*

(iii) If $A \subseteq B$ *and* $B \subseteq C$ *then* $A \subseteq C$ *(Transitivity of inclusion);*

(iv) $A \subseteq B$ *implies that* $A \cup B = B$;

(v) $A \subseteq B$ *implies that* $A \cap B = A$;

(vi) $A \cap B = B \cap A$ *(Commutativity of intersection);*

(vii) $A \cup B = B \cup A$ *(Commutativity of union);*

(viii) $A \cup (B \cup C) = (A \cup B) \cup C$ *(Associativity of union);*

(ix) $A \cap (B \cap C) = (A \cap B) \cap C$ *(Associativity of intersection);*

(x) $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$ *(Distributivity of intersection);*

(xi) $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$ *(Distributivity of union).*

Let X be a non-empty set of sets.
We denote by $\bigcup X$ the set which consists of all the elements of elements of X, i.e. $y \in \bigcup X$ iff there is a $Y \in X$ such that $y \in Y$. $\bigcup X$ is called the union over X. If $X = \{A, B\}$ and A, B are sets, then $\bigcup X = A \cup B$.
We denote by $\bigcap X$ the set which consists of all the elements which are elements of every element of X, i.e. $y \in \bigcap X$ iff for every $Y \in X$ it is true that $y \in Y$. $\bigcap X$ is called the intersection over X. If $X = \{A, B\}$ and A, B are sets, then $\bigcap X = A \cap B$. For $X = \emptyset$ we put $\bigcup X = \emptyset$, but $\bigcap X$ is undefined.

2.1.2: Proposition–Exercise
Let A, B, C be sets and X, Y be non–empty sets of sets. Verify the following statements:

(i)

$$\begin{aligned}(\bigcup X) \cap (\bigcup Y) &= \bigcup\{(A \cap B) \subseteq \bigcup X : A \in X \text{ and } B \in Y\} \\ &= \bigcup\{(A \cap B) \subseteq \bigcup Y : A \in X \text{ and } B \in Y\}.\end{aligned}$$

(ii)

$$\begin{aligned}(\bigcap X) \cup (\bigcap Y) &= \bigcap\{(A \cup B) \subseteq \bigcup X : A \in X \text{ and } B \in Y\} \\ &= \bigcap\{(A \cup B) \subseteq \bigcup Y : A \in X \text{ and } B \in Y\}.\end{aligned}$$

Let A, B be sets. We denote by $A \setminus B$ the set consisting of all the elements of A which are not elements of B. $A \setminus B$ is called the difference of B from A or the complement of B in A.

2.1.3: Proposition–Exercise (de Morgan's laws)
. Let A, B, C be sets and X be a non–empty set of sets. Verify the following statements:

(i) $A \subseteq B$ *implies that* $B \setminus (B \setminus A) = A$.

(ii) $A \subseteq B$ *and* $B \subseteq C$ *implies that* $C \setminus B \subseteq C \setminus A$

(iii) $C \setminus (A \cup B) = (C \setminus A) \cap (C \setminus B)$.

(iv) $C \setminus (A \cap B) = (C \setminus A) \cup (C \setminus B)$.

(v) $C \setminus (\bigcup X) = \bigcap\{C \setminus A \subseteq C : A \in X\}$.

(vi) $C \setminus (\bigcap X) = \bigcup\{C \setminus A \subseteq C : A \in X\}$.

Let A be a set. We define $\wp(A)$ to be the set whose elements are exactly all the subsets of A. $\wp(A)$ is called the power set of A. Clearly $\emptyset \in \wp(A)$ and $A \in \wp(A)$.

2.1.4: Proposition–Exercise
Let A, B be sets and X be a non–empty set of sets. Verify the following statements:

(i) $A \subseteq B$ *implies that* $\wp(A) \subseteq \wp(B)$.

(ii) $\wp(A) \cup \wp(B) \subseteq \wp(A \cup B)$.

(iii) $\wp(A \cap B) \subseteq \wp(A) \cap \wp(B)$.

(iv) $\bigcup\{\wp(A) \in \wp(\wp(A)) : A \in X\} \subseteq \wp(\bigcup X)$.

(v) $\wp(\bigcap X) \subseteq \bigcap\{\wp(A) \in \wp(\wp(A)) : A \in X\}$.

2.2 Relations and functions

The aim of this section is to supply definitions of 'relation', 'function' and related notions in enough generality to be of service throughout the book. These notions ultimately rest on that of the *ordered pair* $\langle a, b\rangle$. Although 'ordered pair' can be defined in terms of the membership relation, as can all the notions of classical mathematics, we will not do this here. For the time being we shall take the ordered pair $\langle a, b\rangle$ to be a basic (undefined) notion with the property that $\langle a, b\rangle = \langle c, d\rangle$ if and only if $a = c$ and $b = d$. $\langle a, b\rangle$ itself is not a set and is treated like an atom, a, b may be sets, atoms or ordered pairs. By abuse of notation we shall write $\langle a_1, a_2, a_3, \ldots, a_n\rangle$ for $\langle \ldots\langle a_1, a_2\rangle, a_3\rangle, \ldots, a_n\rangle$. $\langle a_1, a_2, a_3, \ldots, a_n\rangle$ is called an *n-tuple.*

The *Cartesian product* of two sets A and B, written $A \times B$, is the set consisting of all ordered pairs $\langle a, b\rangle$ with $a \in A$ and $b \in B$. By abuse of notation we write $A_1 \times A_2 \times A_3 \times \ldots \times A_n$ for $(\ldots(A_1 \times A_2) \times A_3) \times \ldots \times A_n)$. We write A^n for $(\ldots(A \times A) \times A)\underbrace{\times \ldots\ldots \times}_{n-3} A)$. We put further $A^1 = A$ and $A^0 = \{\emptyset\}$. The informal definition of A^n can be replaced later by an *inductive definition.*

2.2.1: Proposition–Exercise

(i) $(A \cup B) \times C = (A \times C) \cup (B \times C)$;

(ii) $(A \cap B) \times (C \cap D) = (A \times C) \cap (B \times D)$;

(iii) $(A \setminus B) \times C = (A \times C) \setminus (B \times C)$;

(iv) $A \times B = \emptyset$ *if and only if either* $A = \emptyset$ *or* $B = \emptyset$;

(v) *If* $A \subseteq B$ *then* $A \times C \subseteq B \times C$.

2.2.2: Exercise
Prove or disprove the following statements:

(i) $(A \cup B) \times (C \cup D) = (A \times C) \cup (B \times D)$;

(ii) *If* $B \cup C \subseteq A$ *then* $(A \times A) \setminus (B \times C) = (A \setminus B) \times (A \setminus C)$.

A *binary relation* is a set of ordered pairs. A *binary relation on a set* A is a set of ordered pairs $\langle a_1, a_2 \rangle$ with $a_1, a_2 \in A$. In other words, a binary relation on A is a subset of $A \times A$. The *domain* of a binary relation $R \subseteq A \times B$, which is denoted by $Dom(R)$, is the set $\{a \in A :$ there is a $b \in B$ such that $\langle a, b \rangle \in R\}$. The *range* of a binary relation $R \subseteq A \times B$, which is denoted by $Ran(R)$, is the set $\{b \in B :$ there is a $a \in A$ such that $\langle a, b \rangle \in R\}$. The *field of* R is the set $Ran(R) \cup Dom(R)$. For a binary relation R we denote by R^{-1} the set of ordered pairs $\langle y, x \rangle$ such that $\langle x, y \rangle \in R$.

An *n-ary relation* is a set of n-tuples. An *n-ary relation on a set* A is a subset of A^n.

2.2.3: Example

(i) *The set of* $\langle x, y \rangle \in \mathbf{N^2}$ *such that* $x \leq y$ *is a binary relation. This relation is called the 'natural' order on* $\mathbf{N}$.

(ii) *The set of* $\langle x, y, z \rangle \in \mathbf{R^3}$ *such that* $x^2 + y^2 = z^2$ *is a ternary relation. This relation is called the* geometrical lieu *of all the points in* $\mathbf{R^3}$ *satisfying the equation* $x^2 + y^2 = z^2$.

A *function* f is a binary relation, such that for every x there is at most one y for which $\langle x, y \rangle \in f$. A *function* f *from a set* A *to a set* B, for which we also write $f : A \to B$, is a function with domain A and $Ran(f) \subseteq B$. When f is a function we write $f(x) = y$ instead of $\langle x, y \rangle \in f$. A function $f : A \to B$ is *one to one* if f^{-1} is a function. A function $f : A \to B$ is *onto* if $Ran(f) = B$. The set of all functions $f : A \to B$ is denoted by B^A. If $f : A^n \to B$ is a function from A^n to B, we say also that f is an n-ary function from A to B.

2.2.4: Proposition–Exercise

(i) $Y^\emptyset$ *has exactly one element, namely* $\emptyset$, *regardless whether* Y *is empty or not;*

(ii) *For every set* X *we have* $X^\emptyset = \{\emptyset\}$.

If $f : A \to B$ and $g : B \to C$ are two functions we denote by $g \circ f$ the function $g \circ f : A \to C$ defined by $g \circ f(x) = g(f(x))$. $g \circ f$ is called the *composition of* g *and* f.

2.2.5: Proposition–Exercise
Let $f : A \to B$ *and* $g : B \to C$ *be functions.*

(i) *If* f *and* g *are one to one so is* $g \circ f$;

(ii) *If* f *and* g *are onto so is* $g \circ f$.

2.2.6: Definition (Restriction of a function)
Let $f : A \to B$ be a function and $C \subseteq A$. We denote by $f_{|C}$ the unique function $g : C \to B$ such that for every $a \in C$ we have $f(a) = g(a)$.

This section will be expanded further according to what we might need later.

2.3 Inductive Definitions and Proofs

One of the most frequently used tools to construct sets and prove statements about them in this book is the *principle of mathematical induction*. As we need it in a more general setting than the reader may be used to we first define a few auxiliary concepts.

2.3.1: Definition
Let A be a set, $B \subseteq A$ be a subset of A and $F = \bigcup\{F_n : n \in \mathbf{N}\}$ be a set of functions such that for each $f \in F_n$ $f : A^n \to A$ is an n-ary function on A. We call n the arity of $f \in F_n$. A set $X \subseteq A$ is F-closed over B *if*

(i) $B \subseteq X$ and

(ii) for every $n \in \mathbf{N}$, every $f \in F_n$ and every $\langle a_1, a_2, \ldots, a_n\rangle \in X^n$ also $f(a_1, a_2, \ldots, a_n) \in X$.

2.3.2: Proposition–Exercise
Let A, F be as above. Prove the following statements:

(i) A is F-closed over every $B \subseteq A$.

(ii) Let $X, Y \subseteq A$ be F-closed over some $B \subseteq A$. Then $X \cap Y$ is F-closed over B.

(iii) Let $\mathbf{X}$ be a set of sets $X \subseteq A$ which are all F-closed over some $B \subseteq A$. Then $\bigcap \mathbf{X}$ is F-closed over B.

2.3.3: Proposition–Exercise
Let A, B, F be as above. Let $X_{B,F}$ be the intersection of all subsets of A which are F-closed over B. Prove the following statements:

(i) $X_{B,F}$ is F-closed over B.

(ii) If $X \subseteq A$ is F-closed over B then $X_{B,F} \subseteq X$. In other words, $X_{B,F}$ is the smalllest subset of A which is F-closed over B.

2.3.4: Definition (Inductively defined set)
. We say that a subset Y of A is inductively defined with basis B and closure condition F *if $Y = X_{B,F}$. Y is called the closure of B under F.*

2.3.5: Remark
Propositions 2.3.2 and 2.3.3 establish that Definition 2.3.4 is 'well-defined'.

2.3.6: Example (The set of polynomials in one free variable:)

Let A be the set $(A_0)^*$, the set of finite words over $A_0 = \{0, 1, x, \cdot, +, (,)\}$. We define the set P (of polynomials in one free variable) inductively with basis B and closure condition F.
Basis: *$B = \{0, 1, x\}$, in other words, $0, 1, x$ are in P.*
Closure (condition): *Let $f_1 : A^2 \to A$ be the function consisting of all the pairs $\langle\langle w_1, w_2\rangle, (w_1 + w_2)\rangle$ and Let $f_2 : A^2 \to A$ be the function consisting of all the pairs $\langle\langle w_1, w_2\rangle, w_1 \cdot w_2\rangle$. Now let $F = \{f_1, f_2\}$.*
In other words, if $w_1, w_2 \in P$ so are $(w_1 + w_2)$ and $w_1 \cdot w_2$.

2.3.7: Convention (Proof by induction:)

Let $Y \subseteq A$ be inductively defined with basis B and closure condition F. We would like to prove that some statement Φ is true for all elements of Y. To do this, let X_Φ be the set of all elements of Y for which Φ is true. We call the proof that X_Φ is F-closed over B an inductive proof *for 'Φ is true for all elements of Y'.*

2.3.8: Remark
The use of the phrase 'inductive proof' is justified by the following observations: By Proposition 2.3.3 it suffices to show that X_Φ is F-closed over B, because then $Y \subseteq X_\Phi$ and $X_\Phi \subseteq Y$ by assumption, therefore $Y = X_\Phi$. Clearly then Φ is true for all elements of Y.
Now, a 'proof by induction' consists of the following scenario: First we show that Φ is true for every element of B. We usually call this part of the scenario **Basis**. *Then we have to show that for every $f \in F_n \subseteq F$ and every $a_1, a_2, \ldots, a_n$ which satisfy Φ (i.e which are in X_Φ) also $f(a_1, a_2, \ldots, a_n)$ satisfies Φ. This part of the scenario we usually call* **Closure**.

2.3.9: Example
Let Φ be the statement 'all words (polynomials) of P have an even number of parentheses'.
Basis: *As zero is even, $B \subseteq X_\Phi$.*
Closure: *f_1 adds no parentheses and f_2 always adds two parentheses to a word, therefore X_Φ is F-closed.*

2.3.10: Definition (Inductively defined function)
Let $g : A \to B$ be a function. We say that F is inductively defined, if the set $g \subseteq A \times B$ is inductively defined as a set and g is a function. In other words, g is the smallest set $X_{B,F}$ closed under F over B for some set of functions F and $B \subseteq A \times B$, and for every $a \in A$ there is a unique $b \in B$ with $\langle a, b\rangle \in X_{B,F}$.

2.3.11: Example
We give here an inductive definition for the factorial $fact : \mathbf{N} \to \mathbf{N}$.
Basis: $\langle 0,1\rangle \in fact$ *and* $\langle 1,1\rangle \in fact$;
Closure: *If* $\langle n,k\rangle \in fact$ *then* $\langle n+1, kn\rangle \in fact$.
As usual we write $fact(n) = k$ *for* $\langle n,k\rangle \in fact$.

2.3.12: Proposition–Exercise

(i) If X is finite (or possibly empty) with n elements, then $\{0,1\}^X$ has 2^n elements.

(ii) Let $fact : \mathbf{N} \to \mathbf{N}$ as defined above. Show that for every $n \in \mathbf{N}$ we have that $2^n - 1 \leq fact(n)$

2.4 Some more sets

When we build our mathematical objects, we have at our disposal at the beginning very few sets. We may have some atoms (or a set of atoms), and we have the empty set. We list here a few set building principles, which will be the basis of any later construction. The first three have been used implicitly before.

2.4.1: Principle (Two element sets)
For every two sets A, B there is a set, denoted by $\{A, B\}$, whose only elements are exactly A and B.

2.4.2: Principle (Union of two sets)
For every two sets A, B there is a set, denoted by $A \cup B$, whose only elements are exactly the elements of A and of B.

We have used the *ordered pair* as a basic construction. We now will give a definition of the ordered pair as a set. This definition was proposed by K. Kuratowski. Their are other definition possible, but this is the mostly used.

2.4.3: Definition (Ordered pair)
We denote by $\langle A, B\rangle$ the set $\{\{A\}, \{A, B\}\}$.

2.4.4: Proposition–Exercise
Let A, B, C, D be four sets (or atoms).

(i) If $A \neq B$ then $\langle A, B\rangle \neq \langle B, A\rangle$.

(ii) $\langle A, B\rangle = \langle C, D\rangle$ iff $A = C$ and $B = D$.

2.4.5: Exercise
Which of the following alternative definitions satisfy $F(A, B) = F(C, D)$ iff $A = B$ and $C = D$?

(i) $F(A,B) = \{A, \{A,B\}\}$;

(ii) $F(A,B) = \{A, \{B\}\}$;

(iii) $F(A,B) = \{\{\{A\}\}, \{A,B\}\}$;

(iv) $F(A,B) = \{\{\{A\}\}, \{\{A,B\}\}\}$.

2.4.6: Principle (Union of set of sets)
For every two sets A there is a set, denoted by $\bigcup A$, whose only elements are exactly the elements of every element of A.

2.4.7: Principle (Power set)
For every set A there is a set, denoted by $\wp(A)$, such that $X \in \wp(A)$ iff $X \subseteq A$.

2.4.8: Principle (Specification of subsets)
Let Φ be a property and A be a set. Then there is a set, which contains exactly those elements of A which satisfy Φ. We denote this set by $\{x \in A : x\ satisfies\ \Phi\}$.

The principles described so far are very intuitiv. We build our sets from previously built sets by describing the construction. The sets we can build up to now are all, what we describe intuitively as 'finite'. The next few principles allow us to construct some 'infinite' sets. The German mathematician J.Dedekind (1831-1916) and the Italian mathematician G. Peano (1858-1932) gave the first definition of the Natural Numbers as a set. It says that the Natural Numbers are the smallest set closed under a 'successor function'. The particular definition of such a 'successor function' we shall use here is due to J. von Neumann (1903-1957)

2.4.9: Principle (Closure under successor)
For every set A there is a set B such that $A \subseteq B$ and whenever $x \in B$ then also $(x \cup \{x\}) \in B$.

2.4.10: Proposition–Exercise
For every set A there is a (unique) smallest set $SU(A)$ such that $A \subseteq SU(A)$ and whenever $x \in SU(A)$ then also $(x \cup \{x\}) \in SU(A)$.

2.4.11: Definition (Set of Natural Numbers)
We denote the set $SU(\{\emptyset\})$ by $\mathbf{N}$. and the set $SU(\{\emptyset\}) \setminus \{\emptyset\}$ by $\mathbf{N}^+$.

2.4.12: Definition (Successor function of N)
We define a relation $succ : \mathbf{N} \to \mathbf{N}$ as follows by $succ = \{\langle a,b\rangle \in \mathbf{N}^2 : b = a \cup \{a\}\}$.

2.4.13: Proposition–Exercise
succ is a function which is one–one and onto.

2.4.14: Principle (Closure under ordered pairs)
For every set A there is a set B such that $A \subseteq B$ and whenever $x, y \in B$ then also $\langle x, y \rangle \in B$.

2.4.15: Proposition–Exercise
For every set A there is a (unique) smallest set $CART(A)$ such that $A \subseteq CART(A)$ and whenever $x, y \in CART(A)$ then also $\langle x, y \rangle \in CART(A)$.

2.4.16: Proposition–Exercise
For every A we have:

(i) $A \times A \subseteq CART(A)$;

(ii) $(A \times A) \times A \subseteq CART(A)$ *and* $A \times (A \times A) \subseteq CART(A)$;

(iii) if $X, Y \subseteq CART(A)$ *then* $X \times Y \subseteq CART(A)$.

We now can give a proper definition if the n–fold cartesian product of a set A:

2.4.17: Definition (n–fold Cartesian product)
Let A be a set. We define inductively a function $cart_A : \mathbf{N} \to CART(A)$ as follows:
Basis: $cart_A(0) = \{\emptyset\}$ *and* $cart_A(1) = A$.
Closure: $cart_A(n+1) = cart_A(n) \times A$.

The following is a generalization of the previous constructions.

2.4.18: Principle
For every set A there is a set B such that $A \subseteq B$ and whenever $x, y \in B$ then also $(x \cup \{y\}) \in B$.

2.4.19: Proposition–Exercise
For every set A there is a (unique) smallest set $HF(A)$ such that $A \subseteq HF(A)$ and whenever $x, y \in HF(A)$ then also $(x \cup \{y\}) \in HF(A)$.

2.4.20: Definition (Hereditary finite sets)
We denote the set $HF(\{\emptyset\})$ by **HF**.

HF looks like a 'small world of sets'. In some precise sense all the finite objects which can be built from the empty set alone are in **HF**. The following proposition makes this more precise.

2.4.21: Proposition–Exercise
The set **HF** *has the following properties:*

(i) (Transitivity of $\in$)
If $x \in$ **HF** *and $y \in x$ so $y \in$* **HF**;

(ii) (Closure under subsets)
If $x \in \mathbf{HF}$ *and* $y \subseteq x$ *so* $y \in \mathbf{HF}$*;*

(iii) (Closure under union and intersection)
If $x, y \in \mathbf{HF}$ *so are* $x \cup y \in \mathbf{HF}$ *and* $x \cap y \in \mathbf{HF}$*;*

(iv) (Closure under ordered and unordered pairs)
If $x, y \in \mathbf{HF}$ *so are* $\{x, y\} \in \mathbf{HF}$ *and* $\langle x, y \rangle \in \mathbf{HF}$*;*

(v) (Closure under power sets)
If $x \in \mathbf{HF}$ *so* $\wp(x) \in \mathbf{HF}$*.*

2.4.22: Proposition–Exercise
For every set A *we have that* $SU(A) \subseteq HF(A)$ *and* $CART(A) \subseteq HF(A)$*.*

2.5 Implementing Datastructures as sets

In this section we shall introduce various objects we shall study, such as words, trees, natural numbers and give their definitions as sets. The contents of this section are not essential for understanding the script.

2.5.1 Words

Given the natural number $\mathbf{N}$ denote by I_n the set $I_n = \{0, 1, \ldots, n-1\}$. $I_0 = \emptyset$. Denote by $A^{(n)}$ the set A^{I_n}, i.e. the set of functions $w : I_n \to A$. For $w \in A^{(n)}$

Now we put

2.5.1: Definition (Words)
Let A *be a set (an alphabet).*

(i) $A^* = \bigcup_{n \in \mathbf{N}} A^{(n)}$ *is the set of finite words over* A*;*

(ii) $A^+ = \bigcup_{n \in \mathbf{N}^+} A^{(n)}$ *is the set of finite non–empty words over* A*.*

(iii) We denote by $\ell(w)$ *the unique* $n \in \mathbf{N}$ *such that* $w \in A^{(n)}$*.* $\ell(w)$ *is called the* length of w*, and* ℓ *is a function from* A^* *to* $\mathbf{N}$*.*

(iv) $w(k) \in A$ *denotes the* k*–th letter of the word* w*, if* $k < n$*, otherwise it is not defined.*

(v) The empty word ϵ *is the only element of* $A^{\emptyset}$*.*

(vi) Let $k < n$*,* $v \in A^{(k)}$ *and* $w \in A^{(n)}$*.* v *is an* initial segment of w *if for every* $j \leq k$ *we have* $v(j) = w(j)$*.*

2.5.2: Definition (Concatenation of words)
Let A be a set. We define a binary function $\circ : A^{*2} \to A^*$ inductively as follows:

(i) For $v \in A^{(n)}$ and $a \in A$ we put $v \circ a \in A^{(n+1)}$ defined as $v \circ a(k) = v(k)$ for $k < n$ and $v \circ a(n) = a$.

(ii) We note that for every non–empty word $w \in A^*$ there is a unique $a \in A$ and a unique $v \in A^*$ such that $w = v \circ a$.

(iii) Assume that $\circ : A^* \times A^{(n)} \to A^*$ has been defined, and let $u \in A^*$ and $w \in A^{(n+1)}$. As $w = v \circ a$, we put $u \circ w = (u \circ v) \circ a$.

2.5.3: Proposition–Exercise
Prove by induction:

(i) $\circ : A^* \times A^* \to A^*$ is associative, i.e. for every $u, v, w \in A^*$ we have $(u \circ v) \circ w = u \circ (v \circ w)$.

(ii) ϵ is a neutral element for $\circ$, i.e. for every $w \in A^*$ we have $\epsilon \circ w = w = w \circ \epsilon$.

(iii) For every $v, w \in A^*$ we have $\ell(v \circ w) = \ell(v) + \ell(w)$ and if v is an initial segment of w then $\ell(v) \leq \ell(w)$.

(iv) If A has k elements, then $A^{(n)}$ has k^n elements.

2.5.2 Trees

Trees are often described as undirected circuit–free graphs. We shall describe here how to build directed trees as sets.

We first give an inductive definition of finite trees as a subset of $CART(A)$.

2.5.4: Definition (Tree over A)
Let A be a set. The set $TREE(A) \subseteq CART(A)$ and a function $d : TREE(A) \to \mathbf{N}$, the depth *of a tree, are inductively defined as follows:*
Basis: *Every $a \in A$ is in $TREE(A)$. $d(a) = 0$.*
Closure: *If $T_1, \ldots, T_n \in TREE(A)$ and $d(T_i) = d_i$ for $i \leq n$ then $T = \langle T_1, \ldots, T_n \rangle \in TREE(A)$ and $d(T) = max\{d_1, \ldots, d_n\} + 1$.*
$\langle T_1, \ldots, T_n \rangle$ is called the father *of $T_1, \ldots, T_n$. For $i \leq n$ T_i is the i-th* son *of $\langle T_1, \ldots, T_n \rangle$.*

2.5.5: Definition (Subtrees)
Let T_1 and T_2 be in $TREE(A)$. T_1 is a subtree *of T_2 is defined inductively:*
Basis: *T_2 is a subtree of itself.*

Closure: *If T_1 is a subtree of T_2 and T_0 is a son of T_1 then T_0 is a subtree of T_2.*
The subtrees T of T_2 with $d(T) = 0$ are called leaves *of T_2.*

We next define labeled trees in a similar way:

2.5.6: Definition (Tree over A with node labels from B)
Let A and B be sets. The set $TREE_B(A) \subseteq CART(A \cup B)$ and a function $d : TREE_B(A) \to \mathbf{N}$, the depth *of a tree, are inductively defined as follows:*
Basis: *Every $a \in A$ is in $TREE_B(A)$ with root a. $d(a) = 0$.*
Closure: *If $T_1, \ldots, T_n \in TREE_B(A)$, $b \in B$ and $d(T_i) = d_i$ for $i \leq n$ then $T = \langle b \langle T_1, \ldots, T_n \rangle \rangle \in TREE(A)$ and has root b. $d(T) = max\{d_1, \ldots, d_n\} + 1$.*
b is called the father *of $T_1, \ldots, T_n$. For $i \leq n$ T_i is the i-th* son *of b.*

2.5.7: Definition (Subtrees of labeled trees)
Let T_1 and T_2 be in $TREE_B(A)$. T_1 is a subtree *of T_2 is defined inductively:*
Basis: *T_2 is a subtree of itself.*
Closure: *If T_1 is a subtree of T_2 and T_0 is a son of T_1 then T_0 is a subtree of T_2.*
The subtrees T of T_2 with $d(T) = 0$ are called leaves *of T_2.*

Trees and labeled trees can be drawn as two–dimensional pictures. The drawing can be defined inductively: For a leave just write down the letter $a \in A$.
If $T = \langle T_1, \ldots, T_n \rangle$ and you know how to draw the T_i's, draw T by drawing the T_i's in their order, draw a point above them and link this point with the trees below.
If $T = \langle b, \langle T_1, \ldots, T_n \rangle, \rangle$ and you know how to draw the T_i's, draw T by drawing the T_i's in their order, draw the letter b above them and link this point with the trees below.

2.5.8: Exercise
Draw some sample trees.

2.5.3 Natural numbers and its arithmetic operations

In this subsection we define the arithmetic operations on the Natural Numbers $\mathbf{N}$. Recall that $succ(x) = x \cup \{x\}$.

Our first task is to define the linear order on $\mathbf{N}$.

2.5.9: Definition
We define a binary relation 'less or equal' on $\mathbf{N}$, which we write by infix notation: for $a, b \in \mathbf{N}$ we write $a \leq b$ iff $a \in b$ or $a = b$.

We have to verify that our definition satisfies our intuition, i.e. that $\leq$

behaves as we expect.

2.5.10: Proposition–Exercise (Linear order on N)
Show by induction that with the above definition, $\leq$ is a discrete linear order on $\mathbf{N}$ *with a first element, i.e.*

(i) (Reflexivity) For every $a \in \mathbf{N}$ *we have* $a \leq a$.

(ii) (Transitivity) For every $a, b, c \in \mathbf{N}$, *if* $a \leq b$ *and* $b \leq c$ *then* $a \leq c$.

(iii) (Linearity) For every $a, b \in \mathbf{N}$ *we have* $a \leq b$ *or* $b \leq a$.

(iv) (Antisymmetry) For every $a, b \in \mathbf{N}$ *we have* $a = b$ *iff* $a \leq b$ *and* $b \leq a$.

(v) (Discreteness) For every $a \in \mathbf{N}$ *we have*

(v.a) $a \leq succ(a)$,

(v.b) $a \neq succ(a)$ *and*

(v.c) For every $b \in \mathbf{N}$ *with* $a \leq b \leq succ(a)$ *we have that* $a = b$ *or* $b = succ(a)$.

(vi) (First element) For every $a \in \mathbf{N}$ *we have that* $\emptyset \leq a$.

(vii) (No last element) For every $a \in \mathbf{N}$ *there is a* $b \in \mathbf{N}$ *such that* $b \neq a$ *and* $a \leq b$.

2.5.11: Convention
We write 0 *for* $\emptyset$, *the first element of* $\mathbf{N}$, 1 *for* $succ(0)$, 2 *for* $succ(1)$, *etc.*

Our next task is to define the arithmetic operations addition and multiplication.

2.5.12: Definition (Addition)
Next we define inductively a binary function 'addition', which we again write in infix notation.
Basis: *For* $a \in \mathbf{N}$ *we put* $a+0 = a$ *and* $a+1 = succ(a)$.
Closure: *For* $a, b \in \mathbf{N}$ *we put* $a + succ(b) = succ(a + b)$.

2.5.13: Proposition–Exercise (Properties of addition)
Show by induction that for every $a, b, c \in \mathbf{N}$ *we have*

(i) (Commutativity) $a + b = b + a$.

(ii) (Associativity) $(a + b) + c = a + (b + c)$

(iii) (Neutral element) $a + 0 = 0 + a = a$.

(iv) (Monotonicity) If $a \leq b$ *then* $a + c \leq b + c$.

2.5.14: Definition (Multiplication)
Next we define inductively a binary function 'multiplication', which we again write in infix notation.
Basis: *For* $a \in \mathbf{N}$ *we put* $a \cdot 0 = 0$ *and* $a \cdot 1 = a$.
Closure: *For* $a, b \in \mathbf{N}$ *we put* $a \cdot succ(b) = (a \cdot b) + a$.

2.5.15: Proposition–Exercise (Properties of multiplication)
Show by induction that for every $a, b, c \in \mathbf{N}$ *we have*

(i) $a \cdot 0 = 0$.

(ii) (Commutativity) $a \cdot b = b \cdot a$.

(iii) (Associativity) $(a \cdot b) \cdot c = a \cdot (b \cdot c)$

(iv) (Distributivity) $(a + b) \cdot c = (a \cdot c) + (b \cdot c)$

(v) (Neutral element) $a \cdot 1 = 1 \cdot a = a$.

(vi) (Monotonicity) If $a \leq b$ *then* $a \cdot c \leq b \cdot c$.

2.6 Equipotence

In this section we introduce the notion of 'two sets having the an equal number of elements'.

2.6.1: Definition (Equipotenct sets)
Let A *and* B *be sets. We say that a set* A *is* equipotent *to set* B *if there is a* one–one *function on* A onto B. *If* A *and* B *are* equipotent *we write* $A \sim B$.

This definition corresponds to our intuition when dealing with 'small' sets. Exploring this definition will show that our intuition does not necessarily go beyond the every day expirience. For example, a set A may properly contain a set B and still be equipotent to B. Of course this can not happen if A is 'finite'.

2.6.2: Example
Let $B = \{n^2 : n \in \mathbf{N}\}$. *Then* $\mathbf{N} \supset B$. *Define* $f : B \to \mathbf{N}$, $f(x) = x^2$. *Clearly* f *is one–one and onto* B, *so* $\mathbf{N}$ *and* B *are equipotent.*

2.6.3: Example
The open unit interval $(0, 1) = \{x \in \mathbf{R} \mid 0 < x < 1\}$ *is equipotent to the set* $\mathbf{R}$ *of all real numbers. To see this, let* $f(x) = \tan \frac{\pi(2x-1)}{2}$. *Then* $f : (0, 1) \to \mathbf{R}$ *is one–one and onto, so* $(0, 1) \sim \mathbf{R}$.

We next show that equipotence between sets is an equivalence relation.

2.6.4: Theorem
For all sets $A, B,$ *and* C :

(i) $A \sim A$.

(ii) *If* $A \sim B$, *then* $B \sim A$.

(iii) *If* $A \sim B$, *and* $B \sim C$, *then* $A \sim C$.

Proof. (i) Define $f : A \to A$ by $f(a) = a$ for all $a \in A$. Then $f : A \to A$ is one–one and onto, so $A \sim A$.

(ii) Suppose $A \sim B$. Then there is a function $f : A \to B$, one–one and onto. Clearly $f^{-1} : B \to A$ is one–one and onto, so $B \sim A$.

(iii) Suppose $f : A \to B$, and $g : B \to C$ are one–one and onto functions. Then from Proposition–Exercise 1.2.5, the composition $g \circ f$ is one–one and onto.

∎

So far, equipotence allows us to compare sets. Comparisons to particular sets, of which we have some intuition, is useful. Using $\mathbf{N}$ for such comparisons allows us to define, what we called till now 'finite' and 'infinite'.

2.6.5: Definition (Finite, infinite, countable and uncountable)
Let A *be a set.*

(i) A *is* countable *if* $A \sim B$ *for some* $B \subseteq \mathbf{N}$.

(ii) *If for some* $n \in \mathbf{N}$, $A \sim \{0, 1, \ldots, n-1\}$ *then* A *is* finite.

(iii) A *is* infinite, *if there is a function* $f : \mathbf{N} \to A$ *which is one–one (but not necessarily onto).*

(iv) A *is* uncountable *if it is infinite, but not countable.*

2.6.6: Examples

(i) $\mathbf{R}$ *and* $\mathbf{HF}$ *are infinite.*

(ii) *If* $A \neq \emptyset$ *then* $CART(A)$ *is infinite.*

(iii) *Let* $E = \{n \in \mathbf{N} : n \text{ is even}\}$. E *is countable and infinite.*

2.6.7: Proposition–Exercise

(i) *No set is both finite and infinite.*

(ii) *Let* A *and* B *be finite sets, then* $A \cup B$ *is finite.*

(iii) The union of a finite number of finite sets is finite.

(iv) Let A and B be finite sets, then $A \times B$ is finite.

(v) The cartesian product of a finite number of finite sets is finite.

2.6.8: Proposition

(i) Let A and B be countable sets, then $A \cup B$ is countable.

(ii) Let A and B be countable sets, then $A \times B$ is countable.

(iii) The cartesian product of a finite number of countable sets is countable.

(iv) If A is a countable set, so is $CART(A)$.

(v) The union of a countable number of countable sets is countable.

Sketch of proof:. (i) As A and B are countable, map A into the even numbers and B into the odd numbers.
(ii) and (iii) follow from (iv).
(iv) Define $f : CART(A) \to \mathbf{N}$ inductively. Map A into the even numbers by a function f_0. Extend f as follows: For $a, b \in CART(A)$ and f already defined, put $f(\langle a, b\rangle) = 2^{f(a)} \cdot 3^{f(b)}$. It is easy to see, from the properties of the ordered pair, that f so defined is one–one.
(v) Let $A = \{A_i : n \in \mathbf{N}$ be a family of countable sets and let $f_i : A_i \to \mathbf{N}$ be one–one. We shall map $\bigcup_{i \in \mathbf{N}} A_i$ into $\mathbf{N} \times \mathbf{N}$ by the function g defined as follows: For $a \in A_i$ we put $g(a) = (i, f_i(a))$. It is easy to see that g is one–one. ∎

2.6.9: Proposition
HF *is countable.*

Proof:. We shall write **HF** as a countable union of finite sets. Then the theorem follows from proposition 2.6.8.
We define inductively $\mathbf{HF} = \bigcup\{H_n : n \in \mathbf{N}\}$ with
$H_0 = \{\emptyset\}$;
$H_{n+1} = H_n \cup \{a \cup \{b\} \in \mathbf{HF} : a, b \in H_n\}$. Now prove by induction, that each H_n is finite. ∎

We have not given an example of an uncountable set. To exhibit such sets we first prove atheorem:

2.6.10: Theorem
Let A be an arbitrary set, and f any function $f : A \to \wp(A)$. Then f is not onto $\wp(A)$.

Proof. We need to show that there is a $B \in \wp(A)$ such that $B \notin Ran(f)$.

Let $B = \{x : x \in A \text{ and } x \notin f(x)\}$. Then $B \subseteq A$ and so, $B \in \wp(A)$. Suppose $B \in Ran(f)$, then $B = f(a)$ for some $a \in A$. We ask whether or not $a \in B$?

If $a \in B$ then by definition of B $a \notin f(a)$. But $B = f(a)$ so $a \notin B$, a contradiction.

If $a \notin B$ then $a \in f(a)$ so, again by the definition of B we conclude that $a \in B$, contradiction.

Both assumptions $a \in B$ and $a \notin B$ lead to contradictions, our assumption that $B \in Ran(f)$ is erroneous, so $B \notin Ran(f)$. ∎

2.6.11: Proposition–Exercise
Let A be an arbitrary set. Then $A \not\sim \wp(A)$.

To compare sets which are not equipotent, we introduce the following definition:

2.6.12: Definition
We say that B is at least as numerous as *A if A is equipotent with a subset of B, and we write $A \preceq B$.*
If B is at least as numerous *as A but not equipotent to A, we say that B is* more numerous *than A and we write $A \prec B$.*

2.6.13: Proposition–Exercise
For any three sets A, B, C we have:

(i) $A \prec A$;

(ii) if $A \prec B$ *and* $B \prec C$ *then* $A \prec C$.

Now we are ready to show that uncountable sets in fact do exist.

2.6.14: Theorem (Cantor)
Let A be an arbitrary set, then $A \prec \wp(A)$.

Proof. Let function $f : A \to \wp(A)$ defined by $f(a) = \{a\}$ for all $a \in A$. f is one to one function from A into $\wp(A)$. Thus $A \preceq \wp(A)$. By the previous proposition, A is not equipotent with $\wp(A)$. Hence A is less numerous then $\wp(A)$. ∎

Proposition 2.6.13 states that $\preceq$ is reflexiv and transitiv. If we find that for two sets A, B we have $A \preceq B$ and $B \preceq A$ we would suspect that $A \sim B$. Indeed, we have

2.6.15: Theorem (Cantor-Bernstein)
If $A \preceq B$ and $B \preceq A$, then $A \sim B$.

Proof. Let $f : A \to B$ and $g : B \to A$ be one–one functions. Define C_n, D_n inductively as follows:

$$C_0 = A - Ran(g), \quad D_0 = f(C_0) \quad \text{and,}$$

$$C_{n+1} = g(D_0), \qquad D_n = f(C_n)$$

The function showing that $A \sim B$ is the function $h : A \to B$ defined by:

$$h(x) = \begin{cases} f(x) & \text{if } x \in C_n \text{ for some } n, \\ g^{-1}(x) & \text{otherwise} \end{cases}$$

Clearly that if $x \in A$ but $x \notin C_n$ for any n, follows that $x \notin C_0$ and hence $x \in Ran(g)$, so $g^{-1}(x)$ can be applyied in this case.

We have to show that h is one-to-one and onto B. To show that h is one-to-one consider distinct x' and x'' in A. Since both f and g^{-1} is one-to-one, the case need to be cheked when $x' \in C_k$ for some k, and $x'' \notin \bigcup_n C_n$. In this case there is k such that $h(x') = f(x') \in D_k$, whereas $h(x'') = g^{-1}(x'') \notin D_k$, otherwise we would get that $x'' \in C_{k+1}$, contradiction. So,$h(x') \neq h(x'')$. To show that h is onto B, consider a point y' in $B - \bigcup_n D_n$. Where is $g(y')$? Clearly $g(y') \notin C_0$, also $g(y') \notin C_n$, because $C_{n+1} = g[\![D_0]\!]$, $y' \notin D_n$, and g is one-to-one. So there is x' such that $x' = g(y)$ and $x' \notin C_n$ for any n. This shows that for every $y \in B$, $y \in Ran(h)$. ∎

2.6.16: Corollary
N ∼ HF.

proof:. $\mathbf{N} \subseteq \mathbf{HF}$ by their definitions. So we have $\mathbf{N} \preceq \mathbf{HF}$. By proposition 2.6.9 **HF** is countable. ∎

The Cantor-Bernstein theorem allows us to establish equipotency of various sets. E.g. the set$\{0,1\}^{\mathbf{N}} = 2^{\mathbf{N}}$ off all infinite sequences of 0′s and 1′s is also uncountable.

2.6.17: Theorem
$\wp(\mathbf{N}) \sim 2^{\mathbf{N}}$.

Proof. For each $A \subseteq \mathbf{N}$ define the characteristic function of A, $\chi : \mathbf{N} \to \{0,1\}$, as follows:

$$\chi_A(n) = \begin{cases} 0 & \text{if } n \in A; \\ 1 & \text{if } n \notin A \end{cases}$$

It is easy to check that the correpondence between sets and their characteristic functions is one-to-one mapping of $\wp(\mathbf{N})$ onto $\{0,1\}^{\mathbf{N}}$. ∎

2.6.18: Proposition–Exercise
Let A be an arbitrary set. Then $\wp(A) \prec 2^A$.

2.6.19: Proposition–Exercise
If $A \subseteq B \subseteq C$ and $A \sim C$, then all three sets are equipotent.

2.6.20: Proposition–Exercise
The set $\mathbf{R}$ of real numbers is equipotent to the closed unit interval $[0, 1]$.

Hint: use the preceding proposition with example 2.6.3.

2.6.21: Theorem
The set $\mathbf{R}$ of real numbers is equipotent to the power set of $\mathbf{N}$, i.e. $\mathbf{R} \sim \wp(\mathbf{N})$.

Proof sketch. To prove he theorem we show that $\mathbf{R} \sim 2^{\mathbf{N}}$, and hence $\mathbf{R} \sim \wp(\mathbf{N})$, from theorem 2.6.17.

To prove this it suffices, by the Cantor-Bernstein theorem, to show $\mathbf{R} \preceq 2^{\mathbf{N}}$ and $2^{\mathbf{N}} \preceq \mathbf{R}$.

To show that $\mathbf{R} \preceq \mathbf{2^N}$, we construct a one-to-one function from the open unit interval $(0, 1)$ into $2^{\mathbf{N}}$. The existence of such a function,together with the fact that $\mathbf{R} \sim (\mathbf{0}, \mathbf{1})$, gives us $\mathbf{R} \sim (\mathbf{0}, \mathbf{1}) \preceq \mathbf{2^N}$. The function is defined by use of binary expansions of real numbers: map the real whose binary expansion is $0.1100010\ldots$ to the function in $2^{\mathbf{N}}$ whose succesive values are $1, 1, 0, 0, 0, 1, 0, \ldots$. For definiteness, always select the nonterminating binary expansion.

To show that $2^{\mathbf{N}} \preceq R$ we use decimal expansions. The functionin in $2^{\mathbf{N}}$ whose successive values are $1, 1, 0, 0, 0, 1, 0, \ldots$ is maped to the real number with decimal expansion $0.1100010\ldots$. This maps $2^{\mathbf{N}}$ one-to-one into the closed interval $[0, \frac{1}{9}]$. ∎

3 Propositional Logic

The purpose of this chapter is to introduce the basic questions which one might ask about formal languages and to formulate them precisely in the framework of the language of sets. We also answer these questions in the case of Propositional Logic.

3.1 The Syntax and Semantics of Propositional Logic

3.1.1 Syntax of Propositional Logic

Let $Symb$ be the set consisting of the atoms $\{\wedge, \vee, \rightarrow, \neg, \mathbf{T}, \mathbf{F}, (,)\}$ and Var be the set $\{p_0, p_1, p_2, ..., p_i, ...\}$ where $i \in \mathbf{N}$. We treat the elements of Var also like atoms. $Symb$ is called the set of *logical symbols and parentheses* and Var is called the set of *propositional variables.*

3.1.1: Definition (The set of well formed formulas WFF as strings:)
The set of well formed formulas **WFF** *is a subset of* $(Symb \cup Var)^*$*, the finite words over* $Symb \cup Var$*, defined inductively as follows:*
Basis: *(Atomic formulas of* **WFF***).*
(i) For each $i \in \mathbf{N}$ p_i *is in* **WFF***.*
(ii) $\mathbf{F} \in$ **WFF***.*
(iii) $\mathbf{T} \in$ **WFF***.*
Closure:
(i) If ϕ_1, ϕ_2 *are in* **WFF***, so is* $(\phi_1 \wedge \phi_2)$*;*
(ii) If ϕ_1, ϕ_2 *are in* **WFF***, so is* $(\phi_1 \vee \phi_2)$*;*
(iii) If ϕ_1, ϕ_2 *are in* **WFF***, so is* $(\phi_1 \rightarrow \phi_2)$*;*
(iv) If ϕ *is in* **WFF***, so is* $\neg\phi$*.*

The set $\mathbf{WFF}_{\{\rightarrow,\mathbf{F}\}} \subseteq \mathbf{WFF}$ is defined similarly but using only (i) and (ii) of the basis and (iii) of the closure condition.

We shall always use lower case Greek letters to denote elements of **WFF**. We shall sometimes write $\phi(p_1, p_2, ..., p_n)$ meaning that all the variables occurring in the word ϕ are among $p_1, p_2, ..., p_n$.

3.1.2: Examples

(i) The following are well formed formulas of **WFF***:*
$((p_1 \rightarrow \mathbf{T}) \wedge \neg(\neg p_3 \vee p_1))$, $(((p_1 \rightarrow p_2) \rightarrow p_1) \rightarrow p_3)$, $\neg(\neg p_1 \wedge p_2)$, $(\neg\neg p_1 \vee p_2)$.

(ii) The following are not well formed formulas of **WFF***:*
$(((\neg))$, $(\neg p_1))$, $(\neg p_1)$, $p_1 \wedge p_2$, $(p_1 \wedge p_2$, $(p_1 \wedge p_2))$, $p_1 p_2 \neg))$.

(iii) The following are well formed formulas of $\mathbf{WFF}_{\{\rightarrow,\mathbf{F}\}}$*:*
$(((p_1 \rightarrow \mathbf{F}) \rightarrow p_1) \rightarrow \mathbf{F})$, $((p_1 \rightarrow \mathbf{F}) \rightarrow (p_1 \rightarrow \mathbf{F}))$.

Next we define the set of *well formed tree formulas* **WFTF**.

3.1.3: Definition (The set of well formed tree formulas WFTF:)
Basis: $Var \subseteq$ **WFTF** *and* $\mathbf{T}, \mathbf{F} \in$ **WFTF**. *They are trees consisting of a single node.*
Closure:
(i) If $T_1, T_2 \in$ **WFTF** *then the tree with root* $\wedge$ *and* T_1 *as its left son and* T_2 *as its right son is also in* **WFTF**.
(ii) If $T_1, T_2 \in$ **WFTF** *then the tree with root* $\vee$ *and* T_1 *as its left son and* T_2 *as its right son is also in* **WFTF**.
(iii) If $T_1, T_2 \in$ **WFTF** *then the tree with root* $\rightarrow$ *and* T_1 *as its left son and* T_2 *as its right son is also in* **WFTF**.
(iv) If $T_1 \in$ **WFTF** *then the tree with root* $\neg$ *and the only subtree* T_1 *as its son is also in* **WFTF**.

Let $T \in$ **WFTF**. We can think of *writing* a well formed formula ϕ as the process of obtaining a string $write(T)$ from a tree T. We start at the leaves and just write them down as strings. If we are at a node labeled by a symbol $\bullet$ from $\{\wedge, \vee, \rightarrow\}$ and two subtrees T_1, T_2 we add the parentheses and write the string $(write(T_1) \bullet write(T_2))$. If a node is labeled $\neg$ followed by a subtree T we write the string $\neg write(T)$. We easily observe that $write(T) \in$ **WFF**.

We can think of *reading a formula* ϕ as the process of obtaining a tree T from the string ϕ. One can easily convince oneself that writing is unique. i.e. there is exactly one string ϕ which can be obtained from T in the above way. However, reading ϕ seems more complicated. To establish its uniqueness we have to prove it. The proof can easily be converted into an algorithm.

3.1.4: Theorem (Unique readability of WFF:)

For every well formed formula $\phi \in$ **WFF** *there is exactly one labelled tree* T *such that* $write(T) = \phi$.

Proof. The proof uses the lemmas below. The details of the proof of the theorem and the lemmas are left as an exercise. The first four lemmas can be proved independently. To prove lemma 3.1.9 one needs all the previous lemmas. For lemma 3.1.10 one needs only lemma 3.1.9. Finally, to prove the theorem, one needs only lemma 3.1.10.

3.1.5: Lemma
If $\phi \in$ **WFF** *then* ϕ *is not the empty word and contains at least one variable* $p_i \in Var$ *or one of the constants* **T**,**F**.

3.1.6: Lemma
Let $\phi \in$ **WFF**. *Then the number of left parenthesis in* ϕ *equals the number of right parentheses in* ϕ.

3.1.7: Lemma
*Let $\phi \in$ **WFF** and α be a proper initial segment of the word ϕ, i.e. there exists a word β such that $\beta \neq \epsilon$ and $\phi = \alpha \circ \beta$. Then either α contains no parentheses or the number of left parentheses of α is bigger than the number of its right parentheses.*

3.1.8: Lemma
*Let $\phi \in$ **WFF** and α be an initial segment of the word ϕ. If $\alpha \neq \phi$ and has no parentheses then $\alpha \in \{\neg\}^*$.*

3.1.9: Lemma
*Let $\phi \in$ **WFF** and α be a proper initial segment of the word ϕ. Then $\alpha \notin$ **WFF**.*

3.1.10: Lemma
*Let $\phi, \alpha, \alpha_1, \beta, \beta_1 \in$ **WFF**.*

(i) Let $\bullet\bullet_1 \in \{\wedge, \vee, \rightarrow\}$. If $\phi = (\alpha \bullet \beta$ and $\phi = (\alpha_1 \bullet_1 \beta_1$ then $\alpha = \alpha_1, \beta = \beta_1$ and $\bullet = \bullet_1$.

(ii) If $\phi = \neg\alpha$ and $\phi = \neg\beta$ then $\alpha = \beta$. Furthermore, there are no α_1, β_1 and no $\bullet \in \{\wedge, \vee, \rightarrow\}$ such that If $\phi = (\alpha_1 \bullet \beta_1$.

Combining the preceeding lemmata gives a proof of the above theorem.
∎

3.1.11: Exercise
Change the definition of **WFF** *such that no parentheses are used and show that the Unique Readability Theorem fails for the resulting definition.*

3.1.12: Remark
The inductive definition of well formed formulas suggests that formulas are defined in stages. Inductive definitions are bottom up constructions. The Unique Readibility Theorem allows us to view formulas also top down without ambiguity. We want to make this more precise.

3.1.13: Definition (Rank of a formula:)
*Let rank be a function rank : **WFF** $\rightarrow$ **N** whose value indicates in how many stages a formula has been built. We define inductively sets $\mathbf{WFF}_n$ for $n \in$ **N**.*
Basis: *Let $\mathbf{WFF}_0$ be the set of formulas in $\{\mathbf{T}, \mathbf{F}\} \cup Var$. We call these formulas also* atomic formulas. *If $\phi \in \mathbf{WFF}_0$ then $rank(\phi) = 0$.*
Closure: *Let $\phi_1, \phi_2 \in \mathbf{WFF}_n$. Then ϕ_1, $(\phi_1 \wedge \phi_2)$, $(\phi_1 \vee \phi_2)$, $(\phi_1 \rightarrow \phi_2)$ and $\neg\phi_1$ are in $\mathbf{WFF}_{n+1}$.*
*$rank(\phi_1)$ is defined to be the smallest $n \in$ **N** such that $\phi_1 \in \mathbf{WFF}_n$.*

3.1.14: Proposition–Exercise
Prove the following:

(i) $\mathbf{WFF}_n \subseteq \mathbf{WFF}_{n+1}$;

(ii) $rank(\phi_1 \wedge \phi_2) = 1 + max\{rank(\phi_1), rank(\phi_2)\}$;

(iii) $rank(\phi_1 \vee \phi_2) = 1 + max\{rank(\phi_1), rank(\phi_2)\}$;

(iv) $rank(\phi_1 \rightarrow \phi_2) = 1 + max\{rank(\phi_1), rank(\phi_2)\}$;

(v) $rank(\neg\phi_1) = 1 + rank(\phi)$.

We now generalize our notion of well formed formulas. This will be used in the sequel, but it is useful for checkin whether one understands the material presented so far.

3.1.15: Definition (The set of well formed formulas $\mathbf{WFF}_S$:)
Let $S = \{s_1, s_2, ..., s_m\}$ *be a set of symbols and* $n(s_i) \in \mathbf{N}$ *be a natural number called the* arity *of* s_i. *Let* $S_1 = S \cup \{(, ;,)\}$. *Let* Var *be the set of propositional variables.* $\mathbf{WFF}_S$ *is inductively defined as a subset of* $(S_1 \cup Var)^*$.
Basis:
(i) $Var \subseteq \mathbf{WFF}_S$.
Closure: *If* $s_i \in S$ *and* $n(s_i) = k$ *and* $\phi_1, \phi_2, ..., \phi_k \in \mathbf{WFF}_S$ *then* $s_i(\phi_1; \phi_2; ...; \phi_k) \in \mathbf{WFF}_S$.

3.1.16: Example
(Exercise): Let $S = \{\mathbf{F}, \neg, \wedge, kuku\}$ *with* $n(\mathbf{F}) = 0, n(\neg) = 1, n(\wedge) = 2$, *and* $n(kuku) = 3$. *Write some formulas of* $\mathbf{WFF}_S$.

3.1.17: Theorem
(Exercise): Formulate and prove the Unique Readibility Theorem for $\mathbf{WFF}_S$.

The following will be useful later:

3.1.18: Theorem
Let S *be countable. Then* $\mathbf{WFF}_S$ *is countable.*

3.1.2 The Truth Table Semantics of Propositional Logic

The "meaning" of a well formed formula of **WFF** is a an element of the set $\{0, 1\}$, where we think of 0 as "false" and 1 as "true". Note that the set $\{0, 1\}$ is a subset of $\mathbf{N}$ and the intended interpretation of its elements as "true" and "false" is outside the scope of our mathematical framework. The definition of a *meaning function* given in in the language of sets is the goal of this subsection. We shall do this in three stages, defining *truth tables, assignments* and only then the *meaning function.*

3.1.19: Definition (Truth tables:)

*Let $n \in \mathbf{N}$. An n-*ary truth table *TT is a function $TT : \{0,1\}^n \to \{0,1\}$. With each symbol $\wedge, \vee, \to$ we shall associate binary truth tables $TT_\wedge, TT_\vee, TT_\to$ respectively in the following way:*

(i) $TT_\wedge$ is given by
$TT_\wedge(0,0) = 0$, $TT_\wedge(0,1) = 0$, $TT_\wedge(1,0) = 0$, $TT_\wedge(1,1) = 1$,

(ii) $TT_\vee$ is given by
$TT_\vee(0,0) = 0$, $TT_\vee(0,1) = 1$, $TT_\vee(1,0) = 1$, $TT_\vee(1,1) = 1$,

(iii) $TT_\to$ is given by
$TT_\to(0,0) = 1$, $TT_\to(0,1) = 1$, $TT_\to(1,0) = 0$, $TT_\to(1,1) = 1$,

(iv) With the symbol $\neg$ we shall associate a unary truth table $TT_\neg$ defined by $TT_\neg(0) = 1$ and $TT_\neg(1) = 0$.

(v) With the symbols $\mathbf{T}, \mathbf{F}$ we shall associate the zero-ary truth tables (i.e. constant functions) $TT_\mathbf{T} = 1$ and $TT_\mathbf{F} = 0$ respectively.

3.1.20: Remark
We can think of the truth tables as behavioural descriptions of boolean circuits. The truth table of a $\vee$, ($\wedge, \neg$) describes the behaviour of an or–gate (and–gate, not–gate respectively).

3.1.21: Definition (Truth assignments:)
A (propositional) truth assignment *is a function $z : Var \to \{0,1\}$. We denote by Ass the set $\{0,1\}^{Var}$ of all truth assignments.*

3.1.22: Remark
We can think of the variables in Var as registers and z as a function reading the content of the registers in a current state.

3.1.23: Definition (Meaning function:)
A meaning function *M is a function $M : \mathbf{WFF} \times Ass \to \{0,1\}$.*
We shall denote by M_{PL} the meaning function for Propositional Logic defined inductively as follows:
Basis:
(i) $M_{PL}(p_i, z) = z(p_i)$;
(ii) $M_{PL}(\mathbf{T}, z) = TT_\mathbf{T} = 1$;
(iii) $M_{PL}(\mathbf{F}, z) = TT_\mathbf{F} = 0$.
Closure:
(i) $M_{PL}((\phi_1 \wedge \phi_2), z) = TT_\wedge(M_{PL}(\phi_1), M_{PL}(\phi_2))$;
(ii) $M_{PL}((\phi_1 \vee \phi_2), z) = TT_\vee(M_{PL}(\phi_1), M_{PL}(\phi_2))$;
(iii) $M_{PL}((\phi_1 \to \phi_2), z) = TT_\to(M_{PL}(\phi_1), M_{PL}(\phi_2))$;
(iv) $M_{PL}(\neg\phi, z) = TT_\neg(M_{PL}(\phi))$.

3.1.24: Remark
(i) Note that this defines also a meaning function $M_{\{\rightarrow,\mathbf{F}\}}$ for the well formed formulas of $\mathbf{WFF}_{\{\rightarrow,\mathbf{F}\}}$. As $\mathbf{WFF}_{\{\rightarrow,\mathbf{F}\}} \subseteq \mathbf{WFF}$ we set $M_{\{\rightarrow,\mathbf{F}\}}$ to be the restriction of M_{PL} to $\mathbf{WFF}_{\{\rightarrow,\mathbf{F}\}} \times Ass$.
(ii) We note that the definition above rests on the Unique Readibility Theorem for the formulas of $\mathbf{WFF}$, as it relies not on the string ϕ but on the unique T_ϕ such that $write(T_\phi) = \phi$.

3.1.25: Examples
(i) Show that for every formula $\phi \in \mathbf{WFF}$ such that $\phi = (\phi_1 \rightarrow \phi_1)$ and for every propositional assignment z $M_{PL}(\phi, z) = 1$.
(ii) Show that for every formula $\phi \in \mathbf{WFF}$ such that $\phi = (\phi_1 \wedge \neg\phi_1)$ and for every propositional assignment z $M_{PL}(\phi, z) = 0$.
(iii) Let $\phi_1 = (\neg\psi_1 \vee \psi_2)$ and $\phi_2 = (\psi_1 \rightarrow \psi_2)$. Show that for every propositional assignment z $M_{PL}(\phi_1, z) = M_{PL}(\phi_2, z)$.
(iv) Choose your favorite set of well formed formulas and propositional assignments and compute their respective value under M_{PL}.

It is obvious from the definitions, that the function M_{PL} only depends on finitely many values of z. The next proposition makes this precise.

3.1.26: Proposition (Finite dependency of the meaning function:)

Let $\phi \in \mathbf{WFF}$ be a formula with all its propositional variables in the set $\{p_1, p_2, ..., p_n\}$. Let z_1 and z_2 be two propositional assignments such that for every $i \leq n$ $z_1(p_i) = z_2(p_i)$. Then $M_{PL}(\phi, z_1) = M_{PL}(\phi, z_2)$.

Proof. (by Induction):
Basis: If $\phi \in \mathbf{WFF}_0$ then $\phi = p_i$ for some $i \leq n$ or $\phi = \mathbf{T}$ or $\phi = \mathbf{F}$. In all these cases $M_{PL}(\phi, z)$ depends only on the value of $z(p_i)$ or is constant.
Closure: If $\phi \in \mathbf{WFF}_{n+1}$ and the proposition is true for all $\phi_1, \phi_2 \in \mathbf{WFF}_i$ $i \leq n$, we have four cases.
Let $\phi = (\phi_1 \wedge \phi_2)$. As $M_{PL}((\phi_1 \wedge \phi_2), z) = TT_\wedge(M_{PL}(\phi_1, z), M_{PL}(\phi_2, z))$ and $TT_\wedge$ does not depend on z, the proposition is also true for ϕ.
The other cases are left to the reader. ∎

The above proposition allows us to associate with each well formed formula $\phi \in \mathbf{WFF}$ a truth table TT_ϕ in the following way:

3.1.27: Definition (Truth table associated with ϕ:)
Let $\phi \in \mathbf{WFF}$ and let $p_{i_1}, p_{i_2}, ..., p_{i_n}$ be all the variables occuring in ϕ. Let $TT_\phi : \{0,1\}^n \rightarrow \{0,1\}$ the truth table defined by

$$TT_\phi(x_1, x_2, ..., x_n) = M_{PL}(\phi, z)$$

with $z(p_{i_j}) = x_j$ for $j = 1, 2, \ldots, n$.

Let TT be an n-ary truth table. The question arises whether there exists a formula $\phi \in \mathbf{WFF}$ such that $TT = TT_\phi$? A positive answer to this question, as it is the case, gives us a justification for the choice of the basic truth tables $(TT_\wedge, TT_\vee, TT_\neg)$ underlying the semantics of **WFF**.

3.1.28: Theorem (Functional completeness of the semantics for WFF:)
Let TT be an n-ary truth table. Then there exists a formula $\phi \in \mathbf{WFF}$ such that $TT = TT_\phi$.

Proof. For $n = 0$ there are two constant truth tables.
For $n > 0$, let $x = \langle x_1, x_2, ..., x_n \rangle \in \{0,1\}^n$ such that $TT(x) = 1$. Let l_i be p_i if $x_i = 1$ and $\neg p_i$ if $x_i = 0$. Let C_x be the conjunction of all the l_i, $C_x = ((...(l_1 \wedge l_2) \wedge \ldots \wedge l_n)$. Now let ϕ be the conjunction of all the C_x such that $TT(x) = 1$. It is now easy to verify that $TT_\phi = TT$. ∎

3.1.29: Definition (Semantics for $\mathbf{WFF}_S$:)
Let $S = \{s_1, s_2, ..., s_n\}$ be a set of symbols with arities $n(s_i) \in \mathbf{N}$. For each $i \leq n$ let TT_i be an $n(i)$-ary truth table. Let z be a propositional assignement. We define a meaning function M_S inductively as follows:
Basis: $M_S(\mathbf{T}, z) = 1$, $M_S(\mathbf{F}, z) = 0$, $M_S(p_i, z) = z(p_i)$.
Closure: *For every $i \leq n$ with $n(s_i) = k$ and $\phi_1, \phi_2, ..., \phi_k \in \mathbf{WFF}_S$ $M_S(s_i(\phi_1; \phi_2; ...; \phi_k), z) = TT_i(M_S(\phi_1), M_S(\phi_2), ..., M_S(\phi_k))$.*

3.2 Basic Semantic Concepts

3.2.1 Validity, Logical Equivalence and Logical Consequence

Validity, Logical Equivalence and Logical Consequence are fundamental concepts of the semantics of formal languages in the most general sense. We now introduce them for Propositional Logic, but the reader should have in mind that they are really concepts about meaning functions.

3.2.1: Definition (Validity and Satisfiability:)

(i) We say that a formula $\phi \in \mathbf{WFF}$ is (logically) valid *or a* tautology *if for every propositional assignment z $M_{PL}(\phi, z) = 1$.*

(ii) We say that a formula $\phi \in \mathbf{WFF}$ is a contradiction *if for every propositional assignment z $M_{PL}(\phi, z) = 0$.*

(iii) We say that a formula $\phi \in \mathbf{WFF}$ is satisfiable *if there is a propositional assignment z such that $M_{PL}(\phi, z) = 1$.*

(iv) We say that a set of formulas $\Sigma \subseteq \mathbf{WFF}$ is satisfiable *if there is a propositional assignment z such that $M_{PL}(\phi, z) = 1$ for every $\phi \in \Sigma$. We abreviate this as $M_{PL}(\Sigma, z) = 1$. (Strictly speaking, we extend the function M_{PL} to the power set of* **WFF**.*)*

3.2.2: Examples

(i) Show that ϕ is valid if and only if $\neg\phi$ is not satisfiable.

(ii) Show that ϕ is a contradiction if and only if $\neg\phi$ is valid.

(iii) Find an infinite set of valid formulas.

(iv) Find an infinite set of formulas which are contradictions.

(v) Find an infinite set of satisfiable formulas which are not valid.

3.2.3: Proposition–Exercise
*Show that the following formulas $\phi \in$ **WFF** are valid:*

$$\phi = (\psi_1 \to (\psi_2 \to \psi_1)) \qquad (i)$$

$$\phi = ((\psi_1 \to (\psi_2 \to \psi_3)) \to ((\psi_1 \to \psi_2) \to (\psi_1 \to \psi_3))) \qquad (ii)$$

$$\phi = (((\psi_1 \to \mathbf{F}) \to \mathbf{F}) \to \psi_1) \qquad (iii)$$

3.2.4: Definition (Logical equivalence:)
We say that two formulas ϕ_1, ϕ_2 are logically equivalent (semantically equivalent) *if and only if for every propositional assignement z $M_{PL}(\phi_1, z) = M_{PL}(\phi_2, z)$.*

3.2.5: Examples

*(i) Show that $\phi \in$ **WFF** is valid if and only if ϕ is logically equivalent to the formula $\mathbf{T}$.*

*(ii) Show that $\phi \in$ **WFF** is a contradiction if and only if ϕ is logically equivalent to the formula $\mathbf{F}$.*

*(iii) Show that $\phi \in$ **WFF** is valid if and only if $\neg\phi$ is logically equivalent to the formula $\mathbf{F}$.*

3.2.6: Proposition–Exercise

(i) ϕ is a tautology if and only if TT_ϕ is the constant function with value 1.

(ii) ϕ is satisfiable if and only if there are $x_1, x_2, ..., x_n \in \{0, 1\}$ such that $TT_\phi(x_1, x_2, ..., x_n) = 1$.

*(iii) Two well formed formulas $\phi, \psi \in$ **WFF** are logically equivalent if and only if they have the same truth tables associated with them, i.e. $TT_\phi = TT_\psi$.*

3.2.7: Proposition–Exercise
Show that the following pairs of formulas ϕ_1, ϕ_2 are logically equivalent:
Commutativity:

$$\phi_1 = (\psi_1 \wedge \psi_2), \quad \phi_2 = (\psi_2 \wedge \psi_1); \tag{i}$$

$$\phi_1 = (\psi_1 \vee \psi_2), \quad \phi_2 = (\psi_2 \vee \psi_1); \tag{ii}$$

Associativity:

$$\phi_1 = ((\psi_1 \wedge \psi_2) \wedge \psi_3), \quad \phi_2 = ((\psi_1 \wedge (\psi_2 \wedge \psi_3)); \tag{iii}$$

$$\phi_1 = ((\psi_1 \vee \psi_2) \vee \psi_3), \quad \phi_2 = ((\psi_1 \vee (\psi_2 \vee \psi_3)); \tag{iv}$$

Distributivity:

$$\phi_1 = ((\psi_1 \wedge \psi_2) \vee \psi_3), \quad \phi_2 = ((\psi_1 \vee \psi_3) \wedge (\psi_2 \vee \psi_3)); \tag{v}$$

$$\phi_1 = ((\psi_1 \vee \psi_2) \wedge \psi_3), \quad \phi_2 = ((\psi_1 \wedge \psi_3) \vee (\psi_2 \wedge \psi_3)); \tag{vi}$$

De Morgan's laws:

$$\phi_1 = \neg(\psi_1 \wedge \psi_2), \quad \phi_2 = (\neg\psi_1 \vee \neg\psi_2); \tag{vii}$$

$$\phi_1 = \neg(\psi_1 \vee \psi_2), \quad \phi_2 = (\neg\psi_1 \wedge \neg\psi_2); \tag{viii}$$

Double negation:

$$\phi_1 = \neg\neg\psi, \quad \phi_2 = \psi. \tag{ix}$$

3.2.8: Definition (Logical consequence:)
Let Σ be a (possibly infinite) set of well formed formulas in **WFF**, *and let $\phi \in$* **WFF**. *We say that ϕ* is a logical (semantical) consequence of Σ *or alternatively* Σ logically (semantically) entails ϕ *if and only if for every propositional assignement z such that $M_{PL}(\Sigma, z) = 1$ we have also that $M_{PL}(\phi, z) = 1$. We write $\Sigma \models \phi$ for Σ entails ϕ.*

3.2.9: Examples

(i) Show that ϕ is valid if and only if the empty set $\emptyset$ entails ϕ, i.e. $\emptyset \models \phi$;

(ii) Show that $\{\phi\} \models \phi$; and more generally, that if $\phi \in \Sigma$, then $\Sigma \models \phi$.

3.2.10: Proposition–Exercise
The following are some simple but useful properties of the logical relation consequence:

(i) (False implies everything) For every $\phi \in \mathbf{WFF}$ *we have that* $\{\mathbf{F}\} \models \phi$*;*

(ii) For every $\phi, \psi \in \mathbf{WFF}$ $\{\phi\} \models \psi$ *iff* $(\phi \to \psi)$ *is a tautology;*

(iii) (Modus Ponens) For every Σ, ϕ, ψ *we have that* $\Sigma \cup \{\phi, (\phi \to \psi)\} \models \psi$*.*

(iv) (Monotonicity) If $\Sigma \subseteq \Sigma_1 \subseteq \mathbf{WFF}$*,* $\phi \in \mathbf{WFF}$ *and* $\Sigma \models \phi$ *then also* $\Sigma_1 \models \phi$*.*

(v) (Consequence) $\Sigma \models (\phi \to \psi)$ *iff* $\Sigma \cup \{\phi\} \models \psi$*.*

In the following we sketch a semantic decision procedure for the logical consequence. It is called *semantic,* because it resorts to the truth tables associated with the formulas involved. A *syntactic* decision procedure is a decision procedure whose only data used are the formulas themselves. Syntactic decision procedures will be discussed in a later section.

3.2.11: Theorem (Semantic decision procedure for logical consequences:)
Let Σ *be a finite set of well formed formulas in* **WFF** *and* $\phi \in \mathbf{WFF}$*. There is a decision procedure which decides whether* $\Sigma \models \phi$*.*

Proof. First we observe that $\Sigma \models \phi$ iff for every $\Sigma \cup \{\neg\phi\}$ is not satisfiable. Let $\Sigma\{\psi_1, \ldots, \psi_n\}$ Let TT be the truth table for $(\bigwedge_{i=1,\ldots,n} \psi_i \to \phi)$. By proposition 3.1.26 this truth table is well defined and finite. $\Sigma \models \phi$ iff TT is constant with unique value 0. ∎

3.2.12: Exercise
Generalize the notions tautology, logical equivalence, logical consequence, truth table associated with a formula, functional completeness *to* $\mathbf{WFF}_S$ *with semantics given by arbitrary truth tables and find examples and counter examples for these generalized notions.*

We end this subsection with some additional exercises.

3.2.13: Exercises
Let $k \in \mathbf{N}$*. We define* $\mathbf{WFF}(k)$ *to be the set of formulas* $\phi \in \mathbf{WFF}$ *containing only the variables* $p_1, p_2, \ldots, p_k$*.*

(i) Count the number of distinct formulas in $\mathbf{WFF}(k)$*.*

(ii) Count the number of formulas in $\mathbf{WFF}(k)$ *which are pairwise logically not equivalent.*

(iii) How long is the longest sequence of formulas $\phi_1, \phi_2, \ldots, \phi_n \in \mathbf{WFF}(k)$ *such that for every* $i < n$ *we have that* $\phi_i \models \phi_{i+1}$ *but* $\phi_{i+1} \not\models \phi_i$*.*

3.2.2 Substitution

The purpose of this subsection is to give a precise definition of what we mean by replacing or substituting variables by formulas and replacing or substituting subformulas by other formulas.

3.2.14: Definition (Substitution of variables:)
*Let $\phi \in$ **WFF** be a well formed formula. Let $s : Var \to$ **WFF** be a function assigning to each propositional variable $p_i, i \in \mathbf{N}$ a well formed formula. s is called a* substitution function. *We define inductively a function subst : $\mathbf{WFF} \times \mathbf{WFF}^{Var} \to \mathbf{WFF}$. $subst(\phi, s)$ is the formula obtained from ϕ and s by* replacing *all the variables p_i in ϕ simultaneously by $s(p_i)$.*
Basis: *$subst(p_i, s) = s(p_i)$, $subst(\mathbf{F}, s) = \mathbf{F}$, $subst(\mathbf{T}, s) = \mathbf{T}$.*
Closure: *If $\phi, \phi_1, \phi_2 \in$ **WFF** then*
(i) $subst((\phi_1 \wedge \phi_2), s) = (subst(\phi_1, s) \wedge subst(\phi_2, s))$;
(ii) $subst((\phi_1 \vee \phi_2), s) = (subst(\phi_1, s) \vee subst(\phi_2, s))$;
(iii) $subst((\phi_1 \to \phi_2), s) = (subst(\phi_1, s) \to subst(\phi_2, s))$;
(iv) $subst(\neg\phi, s) = \neg subst(\phi, s)$.

3.2.15: Examples
Make your own examples for s, ϕ and compute $subst(\phi, s)$.

3.2.16: Proposition–Exercise (Finite dependency of substitution:)

*Let $\phi \in$ **WFF** be a formula with all its propositional variables in the set $\{p_1, p_2, ..., p_n\}$. Let s_1 and s_2 be two substitution functions such that for every $i \leq n$ $s_1(p_i) = s_2(p_i)$. Then $subst(\phi, s_1) = subst(\phi, s_2)$.*

3.2.17: Proposition–Exercise

*(i) Let $\phi \in$ **WFF** be not satisfiable and s be a substitution function. Then $subst(\phi, s)$ is not satisfiable.*

*(ii) Let $\phi \in$ **WFF** be a tautology and s be a substitution function. Then $subst(\phi, s)$ is a tautology.*

*(iii) Let $\phi \in$ **WFF** , $z \in Ass$ an assignment and s a substitution function. Define the assignment $z' \in Ass$ by $z'(p_i) = M_{PL}(s(p_i), z)$. Then $M_{PL}(\phi, z' = M_{PL}(subst(\phi, s), z)$.*

3.2.18: Exercise
Visualize for yourself the effect of substitution on ϕ when ϕ is considered as a tree.

We shall also define substitution for subformulas rather than for variables.

3.2.3 Normal Forms

This subsection introduces several normal forms of well formed formulas. In general, a normal form of a formula $\phi \in \mathbf{WFF}$ is a formula $\psi \in \mathbf{WFF}$ which is equivalent to ϕ and whose syntax is constraint by certain limitations such as

(i) negation symbols are only permitted if they occur immediately before a variable (negational normal form);

(ii) when building a formula, conjunctions are applied last (conjunctive normal form), or

(iii) when building a formula, disjunctions are applied last (disjunctive normal form).

In the following we make this precise. The purpose of this subsection is twofold: It gives us many examples of equivalent formulas and it provides us with preprocessing techniques which are the basis for our further development.

3.2.19: Definition (Negational Normal Form:)
We define a subset $\mathbf{NNF} \subseteq \mathbf{WFF}$ *inductively as follows:*
Basis: *The variables* p_i *and their negations* $\neg p_i$ *are in* **NNF**. $\mathbf{F}, \mathbf{T} \in \mathbf{NNF}$.
Closure: *If* $\phi, \psi \in \mathbf{NNF}$ *so are* $(\phi \wedge \psi)$ *and* $(\phi \vee \psi)$.

3.2.20: Remark
Note that formulas in **NNF** *do not contain the symbol* $\rightarrow$. *As* $\phi \rightarrow \psi$ *is logically equivalent to* $\neg\phi \vee \psi$ *formulas containing* $\rightarrow$ *'somehow' contain a 'hidden' negation.*

3.2.21: Examples
$((p_1 \wedge \neg p_2) \vee (\neg p_2 \vee \mathbf{F}))$ *is in* **NNF** *but*
$((p_1 \wedge \neg p_2) \vee (\neg p_2 \rightarrow \mathbf{F}))$ *and* $((p_1 \wedge \neg p_2) \vee \neg(\neg p_2 \vee \mathbf{F}))$ *are not.*

3.2.22: Theorem (Negational Normal Form:)
For every formula $\phi \in \mathbf{WFF}$ *there is a formula* $\psi \in \mathbf{NNF}$ *such that:*

(i) ϕ *is equivalent to* ψ *and*

(ii) ϕ *and* ψ *have the same variables.*

Proof. We define a procedure (function) *mvin* : $\mathbf{WFTF} \rightarrow \mathbf{WFTF}$ inductively. The procedure consists of transforming the tree presentation of a formula by moving the negations to the leaves while preserving logical equivalence. By abuse of notation we shall write nevertheless ϕ instead of T_ϕ, the tree obtained from ϕ by the Unique Readability Theorem.
Basis:

(i) $mvin(p_i) = p_i$, $mvin(\mathbf{F}) = \mathbf{F}$, $mvin(\mathbf{T}) = \mathbf{T}$.

(ii) $mvin(\neg p_i) = \neg p_i$, $mvin(\neg\mathbf{F}) = \mathbf{T}$, $mvin(\neg\mathbf{T}) = \mathbf{F}$.

Closure:

(i) $mvin(\neg\neg\phi) = mvin(\phi)$;

(ii) $mvin((\phi_1 \wedge \phi_2)) = (mvin(\phi_1) \wedge mvin(\phi_2))$;

(iii) $mvin((\phi_1 \vee \phi_2)) = (mvin(\phi_1) \vee mvin(\phi_2))$;

(iv) $mvin((\phi_1 \rightarrow \phi_2)) = (mvin(\neg\phi_1) \vee mvin(\phi_2))$;

(v) $mvin(\neg(\phi_1 \wedge \phi_2)) = (mvin(\neg\phi_1) \vee mvin(\neg\phi_2))$;

(vi) $mvin(\neg(\phi_1 \vee \phi_2)) = (mvin(\neg\phi_1) \wedge mvin(\neg\phi_2))$;

As *mvin* is defined for **WFTF** rather then **WFF** it is well defined and it is easy to verify (cf. Proposition 3.2.7) that logical equivalence is preserved. ∎

3.2.23: Definition (Conjunctive and Disjunctive Normal Form:)
We define subsets $\mathbf{CNF}, \mathbf{DNF} \subseteq \mathbf{WFF}$ *inductively in two stages as follows. We first define* $\mathbf{DISJ}, \mathbf{CONJ} \subseteq \mathbf{WFF}$*:*
Basis*: The variables* p_i *and their negations* $\neg p_i$ *are both in* **DISJ** *and* **CONJ**. $\mathbf{F}, \mathbf{T} \in \mathbf{DISJ}$. $\mathbf{F}, \mathbf{T} \in \mathbf{CONJ}$.
Closure*: If* $\phi, \psi \in \mathbf{DISJ}$ *so is* $(\phi \vee \psi)$.
If $\phi, \psi \in \mathbf{CONJ}$ *so is* $(\phi \wedge \psi)$.
Now we define $\mathbf{CNF}, \mathbf{DNF} \subseteq \mathbf{WFF}$*:*
Basis*:* $\mathbf{DISJ} \subseteq \mathbf{CNF}$. $\mathbf{CONJ} \subseteq \mathbf{DNF}$.
Closure*: If* $\phi, \psi \in \mathbf{CNF}$ *so is* $(\phi \wedge \psi)$.
If $\phi, \psi \in \mathbf{DNF}$ *so is* $(\phi \vee \psi)$.

3.2.24: Examples
$((p_1 \vee \neg p_2) \wedge (\neg p_2 \vee \mathbf{F}))$ *is in* **CNF** *but* $((p_1 \vee \neg p_2) \wedge (\neg p_2 \rightarrow \mathbf{F}))$ *and* $((p_1 \wedge \neg p_2) \vee \neg(\neg p_2 \vee \mathbf{F}))$ *are not.*

3.2.25: Theorem (Conjunctive Normal Form:)
For every formula $\phi \in \mathbf{WFF}$ *there is a formula* $\psi \in \mathbf{CNF}$ *such that:*
(i) ϕ *is equivalent to* ψ *and*
(ii) ϕ *and* ψ *have the same variables.*

Proof. The proof is similar to the proof of the Negational Normal Form Theorem. We define inductively a function *mvout* whose domain is the set of tree presentations of formulas in **NNF** which gives the tree presentation of the desired formula in **CNF**.
Basis:

If $\phi \in \mathbf{CNF}$ then $mvout(\phi) = \phi$.
Closure: Assume ϕ, ϕ_1, ϕ_2 are in **CNF**.
(i) $mvout((\phi \vee (\phi_1 \wedge \phi_2))) = (mvout((\phi \vee \phi_1)) \wedge mvout((\phi \vee \phi_2)))$;
(ii) $mvout(((\phi_1 \wedge \phi_2) \vee \phi)) = (mvout((\phi_1 \vee \phi)) \wedge mvout((\phi_2 \vee \phi)))$;
The remaining details are left as an exercise. ∎

3.2.26: Remark
*Note that the construction in Theorem 3.1.28 gives for any truth table TT a $\psi \in$ **CNF**. This can be exploited for an alternative proof of Theorem 3.2.25.*

3.2.27: Proposition–Exercise (Disjunctive Normal Form:)
*For every formula $\phi \in$ **WFF** there is a formula $\psi \in$ **DNF** such that:*
(i) ϕ is equivalent to ψ and
(ii) ϕ and ψ have the same variables.

3.3 Deduction Methods and Compactness

In this section we present to methods of deduction: proof sequences (or Hilbert style deduction) and resolution (popular in Artificial Intelligence and Automated Theorem Proving). The first is supposed to model human reasoningr. This is an exageration: In the best case it models the way mathematicians and other scholastically educated people write down their arguments, when pressed to do so. In short, it models a stylized form of human reasoning. In this sense it is user friendly. In contrast to this, resolution is more fit for machine implementations, and in this sense it is more machine friendly.

3.3.1 Proof Sequences

In this subsection we want to characterize the notion of logical consequence syntactically. To keep things simple we restrict ourselves to the case of propositional formulas in $\mathbf{WFF}_{\{\rightarrow,\mathbf{F}\}} \subseteq \mathbf{WFF}$.

3.3.1: Definition (The axioms:)
For every $\phi, \psi, \theta \in \mathbf{WFF}_{\{\rightarrow,\mathbf{F}\}}$

(i) $(\phi \rightarrow (\psi \rightarrow \phi))$

(ii) $((\phi \rightarrow (\psi \rightarrow \theta)) \rightarrow ((\phi \rightarrow \psi) \rightarrow (\phi \rightarrow \theta)))$

(iii) $(((\phi \rightarrow \mathbf{F}) \rightarrow \mathbf{F}) \rightarrow \phi)$

are axioms.

3.3.2: Definition (Deducible formulas:)
Let $\Sigma \subseteq \mathbf{WFF}_{\{\rightarrow,\mathbf{F}\}}$ be a set of formulas. We define inductively the set $Ded(\Sigma)$ as follows:
Basis:

(i) $\Sigma \subseteq Ded(\Sigma)$;

(ii) If $\phi \in \mathbf{WFF}_{\{\rightarrow,\mathbf{F}\}}$ *is an axiom then* $\phi \in Ded(\Sigma)$.

Closure: *(Modus ponens)*
If $\phi \in Ded(\Sigma)$ *and* $(\phi \rightarrow \psi) \in Ded(\Sigma)$ *then* $\psi \in Ded(\Sigma)$.

$Ded(\Sigma)$ will turn out to be an inductive definition of the logical consequences of Σ. First we state:

3.3.3: Proposition–Exercise (Soundness of $Ded(\Sigma)$:)
Let $\Sigma \subseteq \mathbf{WFF}_{\{\rightarrow,\mathbf{F}\}}$ *be a set of formulas and* $\phi \in \mathbf{WFF}_{\{\rightarrow,\mathbf{F}\}}$. *If* $\phi \in Ded(\Sigma)$ *then* $\Sigma \models \phi$.

3.3.4: Examples
Prove the following statements:

(i) $Ded(\emptyset)$ *is a subset of the tautologies of* $\mathbf{WFF}_{\{\rightarrow,\mathbf{F}\}}$.

(ii) $Ded(\{\mathbf{F}\}) = \mathbf{WFF}_{\{\rightarrow,\mathbf{F}\}}$

(iii) Let Σ *be infinite and* $\phi \in Ded(\Sigma)$. *Then there is a finite subset* $\Sigma_0 \subseteq \Sigma$ *such that* $\phi \in Ded(\Sigma_0)$.

To show that a formula $\phi \in Ded(\Sigma)$ one has to unwind the inductive definition of $Ded(\Sigma)$. Such an unwinding will be called a proof sequence. More precisely:

3.3.5: Definition (Proof sequences:)
Let $\Sigma \subseteq \mathbf{WFF}_{\{\rightarrow,\mathbf{F}\}}$ *be a set of formulas and* $\phi_1, \ldots, \phi_n$ *be formulas in* $\mathbf{WFF}_{\{\rightarrow,\mathbf{F}\}}$ *We say that* $\phi_1, \ldots, \phi_n$ *is a* proof sequence over Σ *if for each* $i \leq n$ *either*

(i) ϕ_i *is an axiom or* $\phi_i \in \Sigma$ *or*

(ii) (Modus ponens) there are $k, l < i$ *such that* $\phi_l = (\phi_k \rightarrow \phi_i)$

We write $\Sigma \vdash \phi$ *if there is a proof sequence* $\phi_1, \ldots, \phi_n$ *over* Σ *such that* $\phi_n = \phi$.

3.3.6: Proposition–Exercise
Let $\Sigma \subseteq \mathbf{WFF}_{\{\rightarrow,\mathbf{F}\}}$ *be a set of formulas and* $\phi \in \mathbf{WFF}_{\{\rightarrow,\mathbf{F}\}}$. *Then* $\phi \in Ded(\Sigma)$ *iff* $\Sigma \vdash \phi$.

3.3.7: Corollary (Soundness of proof sequences)
Let $\Sigma \subseteq \mathbf{WFF}_{\{\rightarrow,\mathbf{F}\}}$ *be a set of formulas and* $\phi \in \mathbf{WFF}_{\{\rightarrow,\mathbf{F}\}}$. *If* $\Sigma \vdash \phi$ *then* $\Sigma \models \phi$.

3.3.8: Exercise
Show the following statements:

(i) *If* $\emptyset \vdash \phi$ *then* ϕ *is a tautology.*

(ii) *For every* $\phi \in \mathbf{WFF}_{\{\rightarrow,\mathbf{F}\}}$ *we have that* $\mathbf{F} \vdash \phi$.

(iii) *Let* Σ *be infinite and* $\Sigma \vdash \phi$ *there is a finite subset* $\Sigma_0 \subseteq \Sigma$ *such that* $\Sigma_0 \vdash \phi$.

3.3.9: Definition
We say that $\Sigma \subseteq \mathbf{WFF}_{\{\rightarrow,\mathbf{F}\}}$ *is* inconsistent *if* $\Sigma \vdash \mathbf{F}$. *If* Σ *is not inconsistent, we say that* Σ *is* consistent.

3.3.10: Remark
Note, by the soundness of proof sequences, that if Σ *is inconsistent, then* Σ *is not satisfiable.*

3.3.2 Manipulations of Proof Seequences

The following are useful properties for the manipulation of proof sequences.

3.3.11: Proposition–Exercise
Let $\Sigma_0 \subseteq \Sigma \subseteq \mathbf{WFF}_{\{\rightarrow,\mathbf{F}\}}$ *be a set of formulas and* $\phi \in \mathbf{WFF}_{\{\rightarrow,\mathbf{F}\}}$.

(i) *If* $\Sigma_0 \vdash \phi$ *then* $\Sigma \vdash \phi$;

(ii) *If* $\phi_1, \phi_2, \ldots, \phi_n$ *is a proof sequence over* Σ *then for each* $i \leq n$ *we have that* $\Sigma \vdash \phi_i$.

(iii) *If* $\Sigma \vdash \phi$ *and* $\Sigma \vdash (\phi \rightarrow \psi)$ *then* $\Sigma \vdash \psi$.

(iv) *If* $\Sigma \vdash (\phi \rightarrow (\theta \rightarrow \psi))$ *and* $\Sigma \vdash (\phi \rightarrow \theta)$ *then* $\Sigma \vdash (\phi \rightarrow \psi)$.

3.3.12: Proposition (Deduction Theorem)
Let $\Sigma \subseteq \mathbf{WFF}_{\{\rightarrow,\mathbf{F}\}}$ *be a set of formulas and* $\phi, \psi \in \mathbf{WFF}_{\{\rightarrow,\mathbf{F}\}}$.
$\Sigma \vdash (\phi \rightarrow \psi)$ *iff* $\Sigma \cup \{\phi\} \vdash \psi$.

Proof. (i) Assume $\Sigma \vdash (\phi \rightarrow \psi)$. We have to prove that $\Sigma \cup \{\phi\} \vdash \psi$. By proposition 3.3.11 (i) above we have $\Sigma \cup \{\phi\} \vdash (\phi \rightarrow \psi)$ and, using modus ponens, $\Sigma \cup \{\phi\} \vdash \psi$.
(ii) Assume $\Sigma \cup \{\phi\} \vdash \psi$. We have to show that $\Sigma \vdash (\phi \rightarrow \psi)$. Equivalently, we can show that $(\phi \rightarrow \psi) \in Ded(\Sigma)$. Let $K \subseteq Ded(\Sigma \cup \{\phi\})$ be the set of formulas ψ such that $\Sigma \vdash (\phi \rightarrow \psi)$. We show that $K = Ded(\Sigma \cup \{\phi\})$.
Basis: $\Sigma \subseteq K$ and all the axioms are in K (Exercise). In the case that $\psi = \phi$ we use that $\emptyset \vdash (\phi \rightarrow \phi)$.
Closure: Assume θ and $(\theta \rightarrow \psi)$ are in K. We have to show that $\psi \in K$. By assumption $\Sigma \vdash (\phi \rightarrow \theta)$ and $\Sigma \vdash (\phi \rightarrow (\theta \rightarrow \psi))$, therefore, by proposition 3.3.11 (iv), $\Sigma \vdash (\phi \rightarrow \psi)$. ∎

3.3.13: Proposition–Exercise (Dychotomy Theorem)
Let $\Sigma \subseteq \mathbf{WFF}_{\{\rightarrow,\mathbf{F}\}}$ *be a set of formulas and* $\phi, \psi \in \mathbf{WFF}_{\{\rightarrow,\mathbf{F}\}}$.
If both $\Sigma \cup \{\phi\} \vdash \psi$ *and* $\Sigma \cup \{(\phi \rightarrow \mathbf{F})\} \vdash \psi$ *then* $\Sigma \vdash \psi$.

Hint for proof. Use the Deduction Theorem to prove

$$\{(\phi \rightarrow \psi), (\psi \rightarrow \theta)\} \vdash (\phi \rightarrow \theta)$$

Then use this to prove the following three tautologies:

$$((\phi \rightarrow \psi) \rightarrow ((\psi \rightarrow \mathbf{F}) \rightarrow (\phi \rightarrow \mathbf{F})))$$

$$(((\phi \rightarrow \mathbf{F}) \rightarrow \psi) \rightarrow ((\psi \rightarrow \mathbf{F}) \rightarrow ((\phi \rightarrow \mathbf{F}) \rightarrow \mathbf{F})))$$

$$(((\psi \rightarrow \mathbf{F}) \rightarrow ((\phi \rightarrow \mathbf{F}) \rightarrow \mathbf{F})) \rightarrow (((\psi \rightarrow \mathbf{F}) \rightarrow (\phi \rightarrow \mathbf{F})) \rightarrow \psi))$$

Then use these three tautologies and Modus Ponens to prove the Dychotomy Theorem. ∎

Proof sequences capture the essence of proofs and can be used for similar formulas in the following sense:

3.3.14: Proposition–Exercise
Let $\Sigma \vdash \phi$ *and* $s : Var \rightarrow \mathbf{WFF}$ *be a substitution. Then*

$$\{subst(\psi, s) : \psi \in \Sigma\} \vdash subst(\phi, s).$$

3.3.3 Completeness and Compactness

The following shows that the method of proof sequences is sufficiently powerful to obtain all tautologies, or, more generally, all logical consequences of a given set of formulas.

3.3.15: Theorem (Completeness Theorem for Deductions)
Let $\Sigma \subseteq \mathbf{WFF}_{\{\rightarrow,\mathbf{F}\}}$ *be a set of formulas and* $\phi \in \mathbf{WFF}_{\{\rightarrow,\mathbf{F}\}}$.
If $\Sigma \models \phi$ *then* $\Sigma \vdash \phi$.

To prove this theorem we need a definition and two lemmas.

3.3.16: Definition (Maximally consistent set)
A set Σ *of* **WFF** *is* maximally consistent *if it is consistent, and for every* $\phi \in \mathbf{WFF}$ *either* $\phi \in \Sigma$ *or* $\neg\phi \in \Sigma$.

3.3.17: Lemma (Maximally consistent extensions)
Let $\Sigma \subseteq \mathbf{WFF}$ *be a consistent. Then there is a maximally consistent* Σ^* *such that* $\Sigma \subseteq \Sigma^*$.

T. he proof is in stages:
(i) Assume now, that Σ is consistent. Let $\{\phi_i : i \in \mathbf{N}\}$ be an enumeration of the formulas in $\mathbf{WFF}_{\{\rightarrow,\mathbf{F}\}}$. We define in stages a set $\Sigma_\omega \subseteq \mathbf{WFF}_{\{\rightarrow,\mathbf{F}\}}$ in the following way:
$\Sigma_0 = \Sigma$;
$\Sigma_{n+1} = \Sigma_n \cup \{\phi_n\}$ if $\Sigma_n \cup \{\phi_n\}$ is consistent, and $\Sigma_{n+1} = \Sigma_n \cup \{(\phi_n \rightarrow \mathbf{F})\}$ otherwise.
$\Sigma_\omega = \bigcup\{\Sigma_i : i \in \mathbf{N}\}$.
(ii) For every $i \in \mathbf{N}$ we use the Dychotomy Theorem to show that Σ_i is consistent.
(iii) Σ_ω is consistent. For otherwise, there are formulas $\{\psi_1, \ldots, \psi_k\}$ such that $\{\psi_1, \ldots, \psi_k\} \vdash \mathbf{F}$. Then there is $m \in \mathbf{N}$ such that $\psi_1, \ldots, \psi_k, \mathbf{F} \in \Sigma_m$, and therefore, Σ_m is inconsistent, contrary to (iv).
(iv) Σ_ω is a maximally consistent extensions of Σ, i.e. for every $\psi \in \mathbf{WFF}_{\{\rightarrow,\mathbf{F}\}}$ either $\psi \in \Sigma_\omega$ or $\neg\psi \in \Sigma_\omega$. As $\{\phi_i : i \in \mathbf{N}\}$ is an enumeration of the formulas in $\mathbf{WFF}_{\{\rightarrow,\mathbf{F}\}}$, $\psi = \phi_n$ for some $n \in \mathbf{N}$. Therefore $\psi \in \Sigma_n$ or $\neg\psi \in \Sigma_n$.
Now we put $\Sigma^* = \Sigma_\omega$, which completes the proof. ∎

3.3.18: Remark
Note that different enumerations of $\mathbf{WFF}_{\{\rightarrow,\mathbf{F}\}}$ *give different sets* Σ_ω *for the same* Σ. *In general for countable* Σ *there are* 2^ω *maximally consistent extensions.*

3.3.19: Lemma (Assignment for maximally consistent set)
Let $\Sigma \subseteq \mathbf{WFF}$ *be maximally consistent. Then there is an assignment* $z : Var \rightarrow \{0,1\}$ *such that* $M_{PL}(\Sigma, z) = 1$. *In other words,* Σ *is satisfiable.*

T. he proof is in stages:
(i) We first define a propositional assignment $z : Var \rightarrow \{0,1\}$ for Σ in the following way: $z(p_i) = 1$ iff $p_i \in \Sigma$. As Σ is maximally consistent, this is well defined.
(ii) All we need to show now, is that $M_{PL}(\psi, z) = 1$ iff $\psi \in \Sigma$. This we do by induction for every $\psi \in \mathbf{WFF}_{\{\rightarrow,\mathbf{F}\}}$. For ψ atomic this follows from the definition of z. Assume we have shown it for ψ_2 and $\psi = (\psi_1 \rightarrow \psi_2)$. Then $M_{PL}(\psi, z) = 1$ iff $M_{PL}((\psi_1 \rightarrow \psi_2), z) = M_\rightarrow(M_{PL}(\psi_1, z), M_{PL}(\psi_2, z)) = 1$. By the induction hypothesis $M_{PL}(\psi_1, z) = 1$ iff $\psi_1 \in \Sigma$. So assume, for contradiction, that $M_{PL}(\psi_1, z) = 1$, $M_{PL}(\psi_2, z) = 0$ and $\psi \in \Sigma$. Then $\psi_1, (\psi_2 \rightarrow \mathbf{F}), \psi$ are all in Σ, which contradicts the consistency of Σ.
Conclusion: $M_{PL}(\Sigma, z) = 1$, and as $\Sigma \subseteq \Sigma$, $M_{PL}(\Sigma, z) = 1$ which shows that Σ is satisfiable. ∎

Proof of theorem 3.3.15. The proof is in several stages.
(i) (Exercise) First we observe that it suffices to prove that if Σ consistent

then Σ is satisfiable. For this we use the Deduction Theorem.
(ii) By the soundness of proof sequences, if Σ is satisfiable, then Σ is consistent.
(iii) Assume now, that Σ is consistent. So there is Σ^* maximally consistent with $\Sigma \subseteq \Sigma^*$ by lemma 3.3.17. But Σ^* is satisfiable by lemma 3.3.19. ∎

3.3.20: Remark
The above proof of the completeness theorem is mathematically very elegant, but hides the construction of proof sequence, whose existence we show. This proof method was independently suggested by L.Henkin (1949), Hasenjaeger and J.Hintikka (both before 1950 too). The actual construction of proof sequences is the topic of courses in Automated Theorem Proving or in Textbooks on Logic prior to 1960.

3.3.21: Corollary (Compactness Theorem)
Let $\Sigma \subseteq \mathbf{WFF}_{\{\rightarrow,\mathbf{F}\}}$ be an infinite set of formulas. Σ is satisfiable iff every finite subset $\Sigma_0 \subseteq \Sigma$ is satisfiable.

Proof. By the completeness theorem above Σ is satisfiable iff Σ is consistent. Clearly, if Σ is consistent, so is every finite subset $\Sigma_0 \subseteq \Sigma$. Conversely, assume Σ is inconsistent. Therefore there is a proof sequence $\psi_1, \ldots, \psi_k, \mathbf{F} \in \Sigma$, which shows the inconsistency, and therefore, $\{\psi_1, \ldots, \psi_k, \mathbf{F}\} \subseteq \Sigma$ is a finite inconsistent subset. ∎

3.3.4 Resolution

In this subsection we introduce a syntactic method of checking whether a set of formulas Σ is satisfiable called *resolution*. Resolution is a *machine friendly* method which involves some preprocessing transforming the set $\Sigma \subseteq \mathbf{WFF}$ into a set $clause(\Sigma)$ of *clauses*.

3.3.22: Definition (Clauses:)

(i) A literal *is a well formed formula which is either a propositional variable, p_i, or the negation of a propositional variable, $\neg p_i$. The constant $\mathbf{F}$ is also a literal. We denote literals by l_j.*

(ii) A clause *is a finite set of literals $\{l_1, l_2, ..., l_k\}$. We denote clauses by C_j. We denote the empty clause (the empty set of literals) by $\Box$.*

(iii) Let z be a propositional assignment. The meaning function M_{clause} for clauses is defined inductively as follows:
Basis: $M_{clause}(\{l_j\}, z) = M_{PL}(l_j, z)$. $M_{clause}(\Box, z) = 0$.
Closure: $M_{clause}(\{l_1, l_2, ..., l_k\}, z) = max\{M_{clause}(\{l_j\}, z) : j \leq k\}$.
If S is a set of clauses $M_{clause}(S, z) = min\{M_{clause}(C, z) : C \in S\}$.

(iv) Let $\phi \in$ **CNF** *be formula in conjunctive normal form. We define a set $clause(\phi)$ of clauses inductively as follows:*
Basis: *$clause(p_i) = \{p_i\}$. $clause(\neg p_i) = \{\neg p_i\}$. $clause(\mathbf{F}) = \{\Box\}$.*
Closure:
(1) If $clause(\phi_1) = \{C_1\}$ and $clause(\phi_2) = \{C_2\}$ then $clause(\phi_1 \vee \phi_2) = \{C_1 \cup C_2\}$.
(2) $clause(\phi_1 \wedge \phi_2) = clause(\phi_1) \cup clause(\phi_2)$.
(3) If $\Sigma \subseteq$ **CNF** *is a set of formulas in conjunctive normal form then $clause(\Sigma) = \bigcup\{clause(\phi) : \phi \in \Sigma\}$.*

(v) If $\Sigma \subseteq$ **WFF** *is a set of well formed formulas (not necessarily in conjunctive normal form) then $clause(\Sigma) = \bigcup\{clause(cnf(\phi)) : \phi \in \Sigma\}$. Here $cnf(\phi)$ denotes a formula in* **CNF** *logically equivalent to ϕ.*

3.3.23: Proposition–Exercise
(i) Let $\Sigma \in$ **WFF** *be a set of well formed formulas and z be a propositional assignment. Then*

$$M_{PL}(\Sigma, z) = M_{clause}(clause(\Sigma), z).$$

(ii) In particular Σ is satisfiable if and only if there is a propositional assignment z such that $M_{clause}(clause(\Sigma), z) = 1$.

3.3.24: Definition (Resolution trees:)

(i) Let $C_1 \cup \{p_j\}, C_2 \cup \{\neg p_j\}$ be two clauses. We say that the clause $C_1 \cup C_2$ is obtained from $C_1 \cup \{p_j\}, C_2 \cup \{\neg p_j\}$ by one resolution step, *and we write*

$$\frac{C_1 \cup \{p_j\}, C_2 \cup \{\neg p_j\}}{C_1 \cup C_2}$$

(ii) A resolution tree *T is a binary (directed) labeled tree $T = \langle V, E\rangle$ such that the labels of V are clauses and if C_1, C_2 are labels of the two sons of a vertex labeld with C then $C_1 = D_1 \cup \{p_j\}$, $C_2 = D_2 \cup \{\neg p_j\}$ and $C = D_1 \cup D_2$. In other words, the label of the father is the result of performing a resolution step on the labels of its two sons. Note that several vertices may carry the same label.*

(iii) Let S be a set of clauses and C be a clause. We say that S proves *C* by resolution, *if there is a resolution tree with the root labeled C and all the leaves labeled with clauses from S. We write $S \vdash_{res} C$ for S proves C by resolution.*

3.3.25: Examples

(i) Draw a resolution tree for $S \vdash_{res} \Box$ *for*

$$S = \{\{p_1\}, \{p_2\}, \{\neg p_1, \neg p_2\}\}.$$

(ii) Draw a resolution tree for $S \vdash_{res} \Box$ *for*

$$S = \{\{p_1, p_2, p_3\}, \{\neg p_1, p_2\}, \{\neg p_3, p_2\}, \{\neg p_2\}\}.$$

The next proposition establishes that resolution steps preserve the meaning of the clauses on which they are based. We call this the *soundness* of resolution steps. More precisely:

3.3.26: Proposition (Soundness of Resolution:)

(i) Let S *be a set of clauses and* $C_1 \cup \{p_j\}, C_2 \cup \{\neg p_j\} \in S$. *Let* z *be a propositional assignment such that* $M_{clause}(S, z) = 1$. *Then* $M_{clause}(S \cup \{C_1 \cup C_2\}, z) = 1$.

(ii) Let T *be a resolution tree and* S_0 *be the set of clauses which are the labels of its leaves and* S *the set of all its labels. Let* z *be a propositional assignment such that* $M_{clause}(S_0, z) = 1$. *Then* $M_{clause}(S, z) = 1$.

(iii) If S *is a set of clauses such that* $S \vdash_{res} \Box$ *then* S *is not satisfiable.*

Proof. We prove only (i). To prove (ii), we can proceed by induction on the depth of the tree applying (i) as the induction step. (iii) is a direct consequence of (i).
So let $C_1 \cup \{p_j\}, C_2 \cup \{\neg p_j\} \in S$ be two clauses and z be a propositional assignment such that $M_{clause}(S \cup \{C_1 \cup \{p_j\}, C_2 \cup \{\neg p_j\}\}, z) = 1$. We have to show that $M_{clause}(S \cup \{C_1 \cup C_2\}, z) = 1$. It suffices to prove the case where $z(p_j) = 1$, as the case $z(p_j) = 0$ is similar. Now $z(p_j) = 1$ implies that $M_{clause}(C_2, z) = 1$ and therefore $M_{clause}(C_1 \cup C_2, z) = 1$. As $M_{clause}(S, z) = 1$ we also have $M_{clause}(S \cup \{C_1 \cup C_2\}, z) = 1$.
this completes the proof of (i). ∎

We would like to state a completeness theorem for resolution. The obvious formulation would be that for a set of clauses S and a clause C we have that $S \vdash_{res} C$ iff C is a logical consequence of S. The following example shows that this is not true:

3.3.27: Example
Let S *be* $\{\{p_0\}\}$ *and* C *be* $\{p_0, p_1\}$. *Clearly* C *is a logical consequence of* S *but there is no resolution step applicable given* S *only.*

The best we can hope for is the following:

3.3.28: Theorem (Completeness of Resolution for Satisfiability)
Let S *be a set of clauses.* S *is not satisfiable iff* $S \vdash_{res} \Box$.

Proof. This will follow from the Compactness Theorem and the Completeness of the Davis–Putnam Procedure, below. ∎

3.3.29: Definition (Davis–Putnam Procedure)
Let S be a finite set of clauses. Without loss of generality let $p_0, \ldots, p_n$ be all the variables occuring in S.
We define inductively sets of clauses S_j, $S_j(pos)$, $S_j(neg)$, $S_j(-)$ for $j = 0, \ldots, n$ as follows:
$S_0 = S$,
$S_j(pos) = \{C \in S_j : p_j \in C\}$,
$S_j(neg) = \{C \in S_j : \neg p_j \in C\}$,
$S_j(-) = \{C \in S_j : p_j \notin C \text{ and } \neg p_j \notin C\}$,
$S_{j+1} = S_j(-) \cup \{C \cup D : C \cup \{p_j\} \in S_j(pos), D \cup \{\neg p_j\} \in S_j(neg)\}$.

3.3.30: Lemma
For every $j = 0, \ldots, n$ S_j is satisfiable iff S_{j+1} is satisfiable.

Proof. If S_j is satisfiable then S_{j+1} is satisfiable, by the soundness of resolution. So let us assume that S_{j+1} is satisfiable by a truth assignment z. Let z_1 be the truth assignment obtained from z by putting $z_1(p_j) = 1 - z(p_j)$. An easy computation shows that if neither z nor z_1 satisfy S_j then there are clauses C, D with $C \cup D \in \{C \cup D : C \cup \{p_j\} \in S_j(pos), D \cup \{\neg p_j\} \in S_j(neg)\}$ and $M_{clause}(C \cup D, z) = 0$, contradiction. ∎

3.3.31: Lemma
S_{n+1} is either empty or contains only the empty clause. Furthermore, S_{n+1} is empty iff S is satisfiable.

Proof. Exercise. ∎

3.3.32: Theorem (Completeness of the Davis–Putnam Procedure)

A finite set of clauses S is satisfiable iff the Davis–Putnam Procedure returns the empty set.

Proof. Use the lemmas. ∎

3.3.33: Remark (Complexity of Resolution)
The Davis–Putnam Procedure seems rather crude and may need an exponential number of resolution steps. Its performance is also sensitive to the numbering of the variables. However, it was shown in a sequence of papers (Tseitin, Galil, Haken, Urquhart, Szemeredi) that there are many sets of n clauses S which are unsatisfiable and which need an exponential

number of resolution steps to discover the unsatisfiability. In the case of average complexity the situation is more complex as it depends on the input distribution for clauses. There are quite natural distributions for which resolution is polynomial on the average, and others, equally natural, for which resolution is exponential on the average.

3.4 Compactness

3.4.1 A Semantic Proof of Compactness

As a corollary to the Completeness Theorem we stated the Compactness Theorem. We now give a more general version with a semantic proof, which is structurally very similar to the proof of the Completeness Theorem.

3.4.1: Definition
Let $\Sigma \subseteq$ **WFF** *be an infinite set of formulas (even uncountable). We say that* Σ *is* finitely satisfiable *if every finite subset* $\Sigma_0 \subseteq \Sigma$ *is satisfiable.*

3.4.2: Exercise
Let $\Sigma \subseteq$ **WFF** *be an infinite set of formulas (even uncountable) and* $\phi \in$ **WFF**. *Assume that neither* $\Sigma \cup \{\phi\}$ *nor* $\Sigma \cup \{\neg\phi\}$ *are finitely satisfiable. Then* Σ *is not finitely satisfiable.*

3.4.3: Theorem (Compactness Theorem)
Let $\Sigma \subseteq$ **WFF** *be an infinite set of formulas (even uncountable).* Σ *is satisfiable iff* Σ *is finitely satisfiable.*

Proof. The proof is in stages. Here we prove only the countable case. The uncountable case will be dealt with in an exercise in the section on well–orderings.
(i) Clearly, if Σ is satisfiable, then Σ is finitely satsifiable.
(ii) Assume now, that Σ is finitely satisfiable. Let $\{\phi_i : i \in \mathbf{N}\}$ be an enumeration of the formulas in **WFF**. (For uncountable Σ we would use some well ordering of the formulas). We define in stages a set $\Sigma_\omega \subseteq$ **WFF** in the following way:
$\Sigma_0 = \Sigma$;
$\Sigma_{n+1} = \Sigma_n \cup \{\phi_n\}$ if $\Sigma_n \cup \{\phi_n\}$ is finitely satisfiable, and $\Sigma_{n+1} = \Sigma_n \cup \{\neg\phi_n\}$ otherwise.
$\Sigma_\omega = \bigcup\{\Sigma_i : i \in \mathbf{N}\}$.
(iii) For every $i \in \mathbf{N}$ we use the exercise above to show that Σ_i is finitely satisfiable.
(iv) Σ_ω is finitely satisfiable. For otherwise, there is a finite $X \subseteq \Sigma$ which is not satisfiable. Then there is $m \in \mathbf{N}$ such that $X \subseteq \Sigma_m$, and therefore, Σ_m is not finitely satisfiable, contrary to (iii)
(v) Σ_ω is a maximally finitely satisfiable extensions of Σ, i.e. for every $\psi \in$ **WFF** either $\psi \in \Sigma_\omega$ or $\neg\psi \in \Sigma_\omega$. As $\{\phi_i : i \in \mathbf{N}\}$ is an enumeration

of the formulas in **WFF** we have that $\psi = \phi_n$ for some $n \in \mathbf{N}$. Therefore $\psi \in \Sigma_n$ or $\neg\psi \in \Sigma_n$.
(vi) We now define a propositional assignment $z : Var \to \{0,1\}$ for Σ_ω in the following way: $z(p_i) = 1$ iff $p_i \in \Sigma_\omega$. As Σ_ω is maximally finitely satisfiable, this is well defined.
(vii) All we need to show now, is that $M_{PL}(\psi, z) = 1$ iff $\psi \in \Sigma_\omega$ for every $\psi \in$ **WFF**. This we do by induction: For ψ atomic this follows from the definition of z. Assume we have shown it for ψ_1 and ψ_2. If $\psi = (\psi_1 \wedge \psi_2)$, then $M_{PL}(\psi, z) = M_{PL}((\psi_1 \wedge \psi_2), z) = M_\wedge(M_{PL}(\psi_1, z), M_{PL}(\psi_2, z)) = 1$ iff both ψ_1 and ψ_2 are in Σ_ω. By the fact that Σ_ω is maximally finitely satisfiable this is the case iff $\psi \in \Sigma_\omega$. The cases for $\psi = (\psi_1 \vee \psi_2)$, $\psi = (\psi_1 \to \psi_2)$ and $\psi = (\neg\psi_1)$ are similar and left as an exercise.
Conclusion: $M_{PL}(\Sigma_\omega, z) = 1$, and as $\Sigma \subseteq \Sigma_\omega$, $M_{PL}(\Sigma, z) = 1$ which shows that Σ is satisfiable. ∎

3.4.2 Applications of Compactness: Definability

In this subsection we show some applications of the Compactness Theorem. For this we need some more definitions:

3.4.4: Definition (Definability)

(i) Let Σ be a set of formulas in **WFF**. *Let $Ass(\Sigma)$ be the set of truth assignments $z \in Ass$ such that $M_{PL}(\Sigma, z) = 1$. Note that $Ass(\mathbf{T}) = Ass$, the set of all truth assignments.*

(ii) Let K be a subset of Ass. We say that K is definable in **WFF** *if there is a set Σ of formulas in* **WFF** *such that $K = Ass(\Sigma)$.*

(iii) A set K of truth assignments is finitely definable *if there is a finite Σ such that $K = Ass(\Sigma)$.*

3.4.5: Examples

(i) Let K_0 be the set of truth assignments z such that $z(p_0) = 1$. Then $K_0 = Ass(p_0)$, and therefore is definable.

(ii) Let K_1 be the set of truth assignments z such that $z(p_i) = z(p_{2i})$ for every $i \in \mathbf{N}$. Then $K_1 = Ass(\Sigma_1)$, for $\Sigma_1 = \{((p_i \to p_{2i}) \wedge (p_{2i} \to p_i)) : i \in \mathbf{N}\}$, and therefore is definable.

(iii) If $K = Ass(\Sigma)$ and $K' = Ass(\Sigma')$, then $K \cap K'$ is definable by $\Sigma \cup \Sigma'$.

(iv) If $K = Ass(\Sigma)$ and $K' = Ass(\Sigma')$, is $K \cup K'$ definable by $\{\phi \vee \psi : \phi \in \Sigma, \psi \in \Sigma'\}$?

(v) Let $K = Ass(\Sigma)$ and $K' = Ass(\Sigma')$, what can you say about $Ass(\Sigma \cap \Sigma')$? Study several special cases.

(vi) *If* $\Sigma \subseteq \Sigma'$ *then* $Ass(\Sigma') \subseteq Ass(\Sigma)$.

(vii) *Let* $K_{\mathbf{T}}$ *be the set of truth assignments* z *such that* $z(p_i) = 1$ *for every* $i \in \mathbf{N}$. *Note that* $K_{\mathbf{T}}$ *has exactly one element and is definable by* $\{p_i : i \in \mathbf{N}\}$. *(Exercise: Show that* $K_{\mathbf{T}}$ *is not definable by any finite set* Σ *of formulas in* **WFF**, *as in such a* Σ *there are only finitely many variables.)*

(viii) *If* $K = Ass(\Sigma)$ *for some finite* Σ, *then the complement* $K' = Ass \setminus K$ *is also definable. (Exercise: Describe* Σ' *which defines* K'.*)*

3.4.6: Proposition

(i) *Let* K *be a definable set of truth assignments such that* $Ass \setminus K$ *is also definable, i.e. let* $K = Ass(\Sigma)$ *and let* $Ass \setminus K = Ass(\Sigma')$. *Then there is a finite subset* $\Sigma_0 \subseteq \Sigma$ *such that* $K = Ass(\Sigma_0)$.

(ii) *Let* K *be a set of truth assignments.* K *is finitely definable iff both* K *and its complement are definable.*

Proof. (i) By assumption and (iii) above $K \cap K' = \emptyset = \mathrm{Ass}(\Sigma \cup \Sigma')$. Therefore $\Sigma \cup \Sigma'$ is not satisfiable. By the Compactness Theorem there is a finite subset $\Sigma_1 \subseteq \Sigma \cup \Sigma'$ which is not satisfiable. Let $\Sigma_0 = \Sigma \cap \Sigma_1$. By (vi) above, we have $K \subseteq \mathrm{Ass}(\Sigma_0)$. To show that $\mathrm{Ass}(\Sigma_0) \subseteq K$ it suffices to show, that $\mathrm{Ass}(\Sigma_0) \cap (\mathrm{Ass} \setminus K) = \emptyset$. But, as Σ_1 is not satisfiable, neither is $\Sigma_0 \cup \Sigma'$, because $\Sigma_1 \subseteq \Sigma_0 \cup \Sigma'$.
(ii) follows from (i) and (viii) above. ∎

3.4.7: Exercises

(i) *Show that* $Ass \setminus K_{\mathbf{T}}$ *is not definable.*

(ii) *Let* K_{even} *be the set of truth assignments* z *such that* $z(p_{2i}) = 1$ *for every* $i \in \mathbf{N}$. *Show that* K_{even} *is not finitely definable.*

The following was not presented in the course:

3.4.8: Definitions

Let K *be a set of truth assignments.*

(i) *We say that* K does depend on a variable p_i *if there are truth assignments* z *and* z' *which only differ for* p_i, *i.e. with* $z(p_j) = z'(p_j)$ *for every* $j \neq i$, *such that* $z \in K$ *and* $z' \notin K$.

(ii) *Let Support*(K) *be the set of variables on which* K *depends.*

(iii) We say that K has finite support *if Support(K) is finite.*

(iv) Let Σ be a set of formulas in **WFF**. *We say that Σ* is satisfiable over *K if there is a truth assignment $z \in K$ such that $M_{PL}(\Sigma, z) = 1$.*

3.4.9: Examples

(i) If K is finitely definable, then K has finite support.

(ii) K_{even} does not have finite support.

3.4.10: Proposition (Finite Support)
Let K be a set of truth assignments. K is finitely definable iff K has finite support.

Proof. By (i) in the example above, if K is finitely definable, then K has finite support. So assume, K has finite support. Then it suffices to describe the truth table describing K on the variables of Support(K). ∎

3.4.3 Truth Table Extensions of Propositional Logic

In this subsection we consider extensions of propositional logic by infinitary connectives. Let $\bigodot_K$ be a new symbol.

3.4.11: Definition (Syntax of WFF$(\bigodot)$)
The set of infinitary formulas **WFF**$(\bigodot)$ *is defined inductively as follows:*
Basis: **WFF** $\subseteq$ **WFF**$(\bigodot)$.
Closure:
(i) If ϕ and ψ are in **WFF**$(\bigodot)$, *so are $(\phi \wedge \psi)$ $(\phi \vee \psi)$, and $\neg\phi$.*
(ii) If ϕ_i are in **WFF**$(\bigodot)$ *for $i \in \mathbf{N}$, then $\bigodot_{i\in\mathbf{N}} \phi_i$ is in* **WFF**$(\bigodot)$.

3.4.12: Definition (K-semantics for WFF$(\bigodot)$)
Let K be a set of truth assignments and $z : Var \to \{0,1\}$ be a truth assignment. We define the meaning function $M_K(\phi, z)$ for formulas in **WFF**$(\bigodot)$ *inductively as follows:*
Basis: *If $\phi \in$* **WFF** *then $M_K(\phi, z) = M_{PL}(\phi, z)$.*
Closure: *(i) $M_K((\phi_1 \wedge \phi_2), z) = TT_\wedge(M_K(\phi_1), M_K(\phi_2))$;*
(ii) $M_K((\phi_1 \vee \phi_2), z) = TT_\vee(M_K(\phi_1), M_K(\phi_2))$;
(iii) $M_K((\phi_1 \to \phi_2), z) = TT_\to(M_K(\phi_1), M_K(\phi_2))$;
(iv) $M_K(\neg\phi, z) = TT_\neg(M_K(\phi))$;
(v) $M_K(\bigodot_{i\in\mathbf{N}} \phi_i, z) = 1$ iff $\hat{z} \in K$ for $\hat{z}(p_i) = M_K(\phi_i, z)$.

3.4.13: Examples

(i) For $K = K_{\mathbf{T}}$ this defines an infinite conjunction. *In this case we denote $\bigodot$ by $\bigwedge$.*

(ii) For K is the set of truth assignments with at least one value 1 this defines an infinite disjunction*. In this case we denote $\bigodot$ by $\bigvee$.*

3.4.14: Exercise
Define the notions satisfiable, valid, logical consequence, etc. *for $WFF(\bigodot)$ with the K-semantics.*

3.4.15: Definition (Compact Propositional Logic)
Let K be a set of truth assignments. We say that K is compact *if $WFF(\bigodot)$ with the K-semantics is compact, i.e. for every set Σ of $WFF(\bigodot)$, Σ is satisfiable under the K-semantcis iff every finite subset $\Sigma_0 \subseteq \Sigma$ is satisfiable under the K-semantics.*

3.4.16: Examples

(i) If K is finitely definable in WFF then K is compact.

(ii) $K_{\mathbf{T}}$ is not compact.

(iii) K_{even} is not compact.

3.4.17: Theorem (Friedman's Theorem)
Let K be a set of truth assignments. The following are equivalent:

(i) K is finitely definable in **WFF***;*

(ii) K is of finite support;

(iii) K is compact.

Proof. $(i) \Rightarrow (iii)$ is (i) in the example above. $(ii) \Rightarrow (i)$ is the Finite Support Theorem of the previous subsection. So we are left to prove $(iii) \Rightarrow (ii)$.
Let ϕ be the formulas $\bigodot_{i \in \mathbf{N}} p_{2i}$ and ψ be the formulas $\neg \bigodot_{i \in \mathbf{N}} p_{2i+1}$. Let Σ be the set of formulas $\{((p_{2i} \to p_{2i+1}) \wedge (p_{2i+1} \to p_{2i})) : i \in \mathbf{N}\}$.
By assumption $\mathbf{WFF}(\bigodot)$ is compact in the K-semantics.
Claim 1: $\{\phi, \psi\} \cup \Sigma$ is not satisfiable in the K-semantics.
Assume, for contradiction, that z is a truth assignment satisfying $\{\phi, \psi\} \cup \Sigma$. Let z_1 be defined by $z_1(p_i) = z(p_{2i})$ and let z_2 be defined by $z_2(p_i) = z(p_{2i+1})$. As z satisfies Σ we have that $z_1 = z_2$. Furthermore, $M_K(\phi, z) = M_K(\psi, z) = 1$. But we note that $subst(\phi, s_1)$ is equivalent to $\neg subst(\psi, s_2)$ for substitutions s_1 with $s_1(p_{2i}) = p_i$ and s_2 with $s_2(p_{2i+1}) = p_i$. So we have $M_K(\phi, z) = 1 = M_K(subst(\phi, s_1), z_1) = 1 = M_K(\psi, z) = M_K(\neg subst(\phi, s_2), z_2) = 0$, which is a contradiction.
Claim 2: If K does not have finite support, then every finite subset Σ_0 of

$\{\phi, \psi\} \cup \Sigma$ is satisfiable in the K-semantics.
s follows from the definition of the support of K.
Assuming now, that K has no finite support, we get a contradiction to Claim 1. ∎

3.4.18: Corollary
If K is not compact, then either K or $Ass \setminus K$ is not definable in **WFF**.

4 First Order Logic

In the previous chapter we have studied the syntax and semantics of Propositional Logic by modelling both within the language of sets. We have set the examples of the type of reasoning which we can pursue when studying syntax and semantics of logic. The central notion in such a study is the notion of *logical consequence*. In this chapter we shall follow the same line of development in studying *First Order Logic*.

First Order Logic is a formal language for describing structures of algebraic nature. Historically, the first examples of structures indeed were motivated by the developments in Abstract Algebra, i.e. the study of *groups, rings, fields, semi–groups* and many others. In Computer Science such structures are studied under the name of *abstract data types* such as *words, strings, stacks, lists, doubly linked lists, undirected and directed graphs* and many others. The structures mentioned are of different kinds, depending on which notions we declare to be basic. The syntactic list of basic notions will be called a *vocabulary*. The semantic counterpart of a vocabulary τ is an *interpretation of the vocabulary* τ, in short, a τ–structure. A 'museum' of structures is introduced to illustrate the rather abstract concept of a τ–structure.

The chapter proceeds as follows:
We first introduce vocabularies and their interpretations, followed by the museum of examples.
Next we define for each vocabulary τ the set of well formed τ–formulas which forms the syntax of first order logic. Then we define its semantics, a meaning function which has three arguments, a formula, a structure and an assignment, and whose value again is a value in $\{0, 1\}$. Using this meaning function we shall, in stricy analogy, define tautologies of First Order Logic, satisfiability and logical consequence. After the introduction of these basic semantic concepts we return for a visit to our museum of examples.
We shall also introduce a notion of *proof sequences* for First Order Logic and show its completeness. As a consequence we shall also obtain a *compactness theorem* for First Order Logic. A new visit to our museum will yield surprising applications of the compactness theorem.

4.1 Vocabularies and Structures

Vocabularies are sets of relation symbols, function symbols and constant symbols. Like in natural language, vocabularies for first order logic are the building blocks of first order languages which are subject to various interpretations. These interpretations are called first order structures.

4.1.1 Vocabularies

4.1.1: Definition (The countable universal vocabulary)
The countable universal vocabulary τ_ω *consists of the following:*

(i) For every natural number n *and* α *we have a* relation symbol $R_{n,\alpha}$ *of* arity n *and identification number* α*;*

(ii) For every natural number n *and* α *we have a* function symbol $F_{n,\alpha}$ *of* arity n *and identification number* α*;*

(iii) For every natural number α *we have a* constant symbol c_α*.*

We can also consider uncountable vocabularies τ_I *over finite arities for an arbitrary set* I*. In this case we change the definition above and request that* n *be a natural number and* $\alpha \in I$*.*

4.1.2: Definitions (Vocabularies)

(i) A vocabulary *is a subset of* $\tau \subseteq \tau_\omega$*. We usually denote vocabularies with the Greek letter* τ *or with* τ_x *where* x *can be any symbol serving as an index.*

(ii) A vocabulary τ *is called* finite (empty) *if it is a finite (empty) subset of* τ_ω*.*

(iii) A vocabulary τ *is* relational *if it does not contain any function symbol.*

(iv) A function symbol (relation symbol) of arity one is called unary*. Unary relation symbols are also called* predicate symbols*. A function symbol (relation symbol) of arity two is called* binary*. A function symbol (relation symbol) of arity three is called* ternary*. More generally, a function symbol (relation symbol) of arity* n *is called* n*–ary.*

4.1.3: Examples

(i) $\tau_1 = \{R_{2,0}\}$ *is a vocabulary which consists of one binary relation symbol with identification number* 0*.* τ_1 *is relational and finite.*

(ii) $\tau_{arith} = \{c_0, c_1, F_{2,0}, F_{2,1}, R_{2,0}\}$ *consists of two constant symbols, two binary function symbols and one binary relation symbol.* τ_{arith} *is finite but not relational. Usually* $F_{2,0}$ *stands for addition,* $F_{2,1}$ *for multiplication and* $R_{2,0}$ *for an order relation, so we shall often write, for simplicity, but contrary to our convention,* $\tau_{arith} = \{c_0, c_1, F_+, F_*, R_<\}$.

(iii) As an example of an uncountable vocabulary, let τ_{real} *be* $\tau_{arith} \cup \{c_r : r \in \mathbf{R}\}$, *i.e. we add to* τ_{arith} *a constant symbol* c_r *for every real number* $r \in \mathbf{R}$. *Note that here the vocabulary is just a set, and the problem of giving its elements constructive names is disregarded.*

4.1.2 Interpretations of Vocabularies

Names in natural language are given meaning by associating with them objects or concepts. The symbols in our vocabularies are given meaning by associating with them sets.

4.1.4: Definition (τ–structures)
Let **Var** *be a dummy symbol, later to be used as the name of the set of variables. Let* A *be any set and let* $\mathcal{A}$ *be a function from* $\{\mathbf{Var}\} \cup \tau$ *into* $A \cup \bigcup_{n \in \mathbf{N}} \wp(A^n)$ *such that*

(i) $\mathcal{A}(\mathbf{Var}) = A$;

(ii) For every constant symbol $c_\alpha \in \tau$ $\mathcal{A}(c_\alpha) \in A$;

(iii) For every relation symbol $R_{n,\alpha} \in \tau$ $\mathcal{A}(R_{n,\alpha}) \subseteq A^n$;

(iv) For every function symbol $F_{n,\alpha} \in \tau$ $\mathcal{A}(F_{n,\alpha})$ *is a function from* A^n *into* A.

A *is called the* universe *of* $\mathcal{A}$. *We say also that* $\mathcal{A}$ *is a* τ*-structure over the universe* A.

4.1.5: Examples

(i) Let $\tau = \emptyset$. *Then a* τ*-structure* $\mathcal{A}$ *over a universe* A *is just the set* A.

(ii) Let $\tau = \tau_{arith}$. *Let* $\mathcal{N}$ *be the* τ*-structure with* $\mathcal{N}(\mathbf{Var}) = \mathbf{N}$ *and with*

(ii.a) $\mathcal{N}(c_0) = 0$, $\mathcal{N}(c_1) = 1$,
(ii.b) $\mathcal{N}(F_{2,0})(n, m) = n + m$,
(ii.c) $\mathcal{N}(F_{2,1})(n, m) = n \cdot m$,
(ii.d) $(n, m) \in \mathcal{N}(R_{2,0})$ *iff* $n \leq m$.

$\mathcal{N}$ *is called the* (standard) Arithmetic Structure of the Natural Numbers.

(iii) Again, let $\tau = \tau_{arith}$. *Let* $\mathcal{Z}$ *be the* τ*-structure with* $\mathcal{Z}(\mathbf{Var}) = \mathbf{Z}$, *the integers, and with*

(iii.a) $\mathcal{Z}(c_0) = 0$, $\mathcal{Z}(c_1) = 1$,

(iii.b) $\mathcal{Z}(F_{2,0})(n,m) = n + m$,

(iii.c) $\mathcal{Z}(F_{2,1})(n,m) = n \cdot m$,

(iii.d) $(n,m) \in \mathcal{Z}(R_{2,0})$ *iff* $n \leq m$.

$\mathcal{Z}$ *is called the* Arithmetic Structure of the Integers. *More examples are discussed in detail in subsection 4.2.*

4.1.3 Isomorphisms and Substructures *

We now introduce a notion of indistinguishability of structures called isomorphism of τ-structures. The underlying idea behind this notion is that the nature of the elements of the universe of a τ-structure is irrelevant to the structure. What matters are the relations between the elements, and the behaviour of the functions. When dealing with data structures in computing, the same approach is followed. If a programming language knows of natural numbers, integers, reals, stacks, lists, then those structures are determined by some behavioural description which does not depend on the particular implementation of the data structure in the assembly language of the implementation.

4.1.6: Definition
Let $\mathcal{A}$ *be a* τ*-structure and* X *be a subset of the universe* $A = \mathcal{A}(\mathbf{Var})$ *of* $\mathcal{A}$. X *is* τ-closed *if*

(i) For every $c_\alpha \in \tau$ *we have that* $\mathcal{A}(c_\alpha) \in X$ *and*

(ii) for every $F_{n,\alpha} \in \tau$ *and* $\vec{a} = (a_1, \ldots, a_n) \in X^n$ *we have that* $F_{n,\alpha}(\vec{a}) \in X$.

4.1.7: Exercise
Make some examples of τ*-closed and not* τ*-closed subsets of some* τ*-structures* $\mathcal{A}$.

4.1.8: Definition
Let $\mathcal{A}$ *and* $\mathcal{B}$ *be two* τ*-structures. Let* X *be a subset of the universe* $A = \mathcal{A}(\mathbf{Var})$ *of* $\mathcal{A}$ *and* Y *be a subset of the universe* $B = \mathcal{B}(\mathbf{Var})$ *of* $\mathcal{B}$. *Furthermore let* $f : X \mapsto Y$ *be a function, which is 1–1 and onto.*

(i) f *is a* partial isomorphism with domain X and range Y *if*

(i.a) X *is* τ*-closed;*

(i.b) For every $c_\alpha \in \tau$ *we have that* $f(\mathcal{A}(c_\alpha)) = \mathcal{B}(c_\alpha)$*;*

(i.c) for every $F_{n,\alpha} \in \tau$ *and* $\vec{a} = (a_1, \ldots, a_n) \in X^n$ *we have that* $f(\mathcal{A}(F_{n,\alpha})(\vec{a})) = \mathcal{B}(F_{n,\alpha})(f(a_1), \ldots, f(a_n))$*;*

(i.d) for every $R_{n,\alpha} \in \tau$ *and every* $\vec{a} = (a_1, \ldots, a_n) \in X^n$ *we have that* $\vec{a} \in \mathcal{A}(R_{n,\alpha})$ *iff* $(f(a_1), \ldots, f(a_n)) \in \mathcal{B}(R_{n,\alpha})$*.*

(ii) f *is an* isomorphism *between the* τ*-structures* $\mathcal{A}$ *and* $\mathcal{B}$ *if* f *is a partial isomorphism with domain* $\mathcal{A}(\mathbf{Var})$ *and range* $\mathcal{B}(\mathbf{Var})$*.*

(iii) Two τ*-structures* $\mathcal{A}$ *and* $\mathcal{B}$ *are* isomorphic *if there exists an isomorphism* f *between* $\mathcal{A}$ *and* $\mathcal{B}$*. In this case we write* $\mathcal{A} \simeq \mathcal{B}$*.*

(iv) f *is an* automorphism *of the* τ*-structure* $\mathcal{A}$ *if* f *is a partial isomorphism with domain* $\mathcal{A}(\mathbf{Var})$ *and range* $\mathcal{A}(\mathbf{Var})$*.*

(v) f *is an* embedding *of the* τ*-structure* $\mathcal{A}$ *into the* τ*-structure* $\mathcal{B}$ *if* f *is a partial isomorphism with domain* $\mathcal{A}(\mathbf{Var})$ *and range* $Y \subseteq \mathcal{B}(\mathbf{Var})$*. In this case we write* $\mathcal{A} \sqsubseteq_f \mathcal{B}$*.*

(vi) $\mathcal{A}$ *is* embeddable *into* $\mathcal{B}$ *if there exists an* f *such that* $\mathcal{A} \sqsubseteq_f \mathcal{B}$*. In this case we write* $\mathcal{A} \sqsubseteq \mathcal{B}$*.*

4.1.9: Proposition (Exercise)
The isomorphism relation between τ*-structures is an equivalence relation. More precisely, let* $\mathcal{A}, \mathcal{B}, \mathcal{C}$ *be three* τ*-structures. Then*

(i) $\mathcal{A} \simeq \mathcal{A}$*;*

(ii) If $\mathcal{A} \simeq \mathcal{B}$ *then* $\mathcal{B} \simeq \mathcal{A}$*;*

(iii) If $\mathcal{A} \simeq \mathcal{B}$ *and* $\mathcal{B} \simeq \mathcal{C}$ *then* $\mathcal{A} \simeq \mathcal{C}$*.*

4.1.10: Definition (Substructures)
Let $\mathcal{A}$ *be a* τ*–structure.*

(i) We say that a τ*–structure* $\mathcal{B}$ *is a* substructure *of* $\mathcal{A}$*, or that* $\mathcal{A}$ *is an* extension *of* $\mathcal{B}$*, and we write* $\mathcal{B} \subseteq \mathcal{A}$*, if*

(i.a) $\mathcal{B}(\mathit{Var}) \subseteq \mathcal{A}(\mathit{Var})$*;*

(i.b) $\mathcal{B}$ *is* τ*–closed;*

(i.c) for every n*–ary relation symbol* $R \in \tau$ $\mathcal{B}(R) = \mathcal{A}(R) \cap \mathcal{B}(\mathit{Var})^n$*.*

(ii) *Let $B \subseteq \mathcal{A}(Var)$ be τ–closed. Let $\mathcal{B}$ be the unique substructure of $\mathcal{A}$ with $\mathcal{B}(Var) = B$. We say that $\mathcal{B}$ is* induced *by B on $\mathcal{A}$.*

(iii) *Let $\mathcal{B}_i, i \in I$ be a family of substructures of $\mathcal{A}$. We denote by $\bigcap_{i\in I} \mathcal{B}_i$ the unique substructure on $\mathcal{A}$ induced by $X = \bigcap_{i\in I} \mathcal{B}_i(Var)$, provided X is not empty.*

4.1.11: Exercise

(i) *$\mathcal{B}$ is a* substructure *of $\mathcal{A}$ iff $\mathcal{B}(\mathbf{Var}) \subseteq \mathcal{A}(\mathbf{Var})$ and the identity function $1_B : \mathcal{B}(\mathbf{Var}) \mapsto \mathcal{A}(\mathbf{Var})$ is an* embedding *of $\mathcal{B}$ into the τ-structure $\mathcal{A}$.*

(ii) *Prove the uniqueness claims in the above definition, i.e. let $\mathcal{B}_1, \mathcal{B}_2$ be two substructures of $\mathcal{A}$ with $\mathcal{B}_1(Var) = \mathcal{B}_2(Var)$ then $\mathcal{B}_1 = \mathcal{B}_2$.*

(iii) *Show that if $\mathcal{B} \subseteq \mathcal{A}$ then $\mathcal{B} \sqsubseteq \mathcal{A}$.*

(iv) *Find two τ–structures $\mathcal{A}$ and $\mathcal{B}$ such that $\mathcal{B} \sqsubseteq \mathcal{A}$ but not $\mathcal{B} \subseteq \mathcal{A}$.*

4.1.12: Proposition (Exercise)
The substructure relation between τ–structures is a partial order on the class of τ–structures. The embedding relation is only transitive. In other words, let $\mathcal{A}, \mathcal{B}$ and $\mathcal{C}$ be τ-structures.

(i) *If $\mathcal{A} \subseteq \mathcal{B}$ and $\mathcal{B} \subseteq \mathcal{A}$ then $\mathcal{A} = \mathcal{B}$.*

(ii) *If $\mathcal{A} \subseteq \mathcal{B}$ and $\mathcal{B} \subseteq \mathcal{C}$ then $\mathcal{A} \subseteq \mathcal{C}$.*

(iii) *If $\mathcal{A} \sqsubseteq \mathcal{B}$ and $\mathcal{B} \sqsubseteq \mathcal{C}$ then $\mathcal{A} \sqsubseteq \mathcal{C}$.*

(iv) *If, additionally, $\mathcal{A}$ or $\mathcal{B}$ is finite, then If $\mathcal{A} \sqsubseteq \mathcal{B}$ and $\mathcal{B} \sqsubseteq \mathcal{A}$ then $\mathcal{A} \simeq \mathcal{B}$.*

4.1.13: Definition

(i) *Let $\mathcal{A}$ be a τ–structure. $\mathcal{A}$ is* minimal *if $\mathcal{A}$ has no proper substructure, i.e. for every substructure $\mathcal{B}$ of $\mathcal{A}$ we have that $\mathcal{B} = \mathcal{A}$.*

(ii) *Let $\mathcal{A}$ be a τ–structure and $X \subseteq \mathcal{A}(\mathbf{Var})$. $\mathcal{A}$ is* minimal over X *if $\mathcal{A}$ has no proper substructure containing X, i.e. for every substructure $\mathcal{B}$ with $X \subseteq \mathcal{B}(\mathbf{Var})$ of $\mathcal{A}$ we have that $\mathcal{B} = \mathcal{A}$.*

4.1.14: Exercise
Let $\mathcal{B}$ be a τ–structure and $\mathcal{A}$ be a substructure of $\mathcal{B}$. Show that the following are equivalent:

(i) $\mathcal{A}$ *is minimal;*

(ii) $\mathcal{A}$ *is minimal over* $\emptyset$*;*

(iii) $\mathcal{A} = \bigcap\{\mathcal{C} : \mathcal{C} \subseteq \mathcal{B}\}$*;*

(iv) For every $X \subseteq \mathcal{A}(Var)$ *which is* τ*–closed,* $X = \mathcal{A}(Var)$*.*

4.1.15: Proposition–Exercise
Let $\mathcal{A}$ *be a* τ*–structure and* $X \subseteq \mathcal{A}(\mathbf{Var})$ *and* $X \neq \emptyset$*. Show that there is a unique substructure* $\mathcal{B}_X$ *of* $\mathcal{A}$ *which is minimal over* X*.*

4.2 A Museum of Structures

In this section we list some examples, which are useful to train our imagination.

4.2.1: Definition (Expansions and reducts)
Let τ *be a vocabulary and* $\tau_1 \subseteq \tau$*. Let* $\mathcal{A}$ *be a* τ*–structure and* $\mathcal{A}_1$ *the* τ_1*–structure defined by* $\mathcal{A}_1(\mathbf{Var}) = \mathcal{A}(\mathbf{Var})$ *and for every relation, function and constant symbol* $X \in \tau_1$ $\mathcal{A}_1(X) = \mathcal{A}(X)$*. In this situation we say that* $\mathcal{A}$ *is a* τ–expansion of $\mathcal{A}_1$ *or that* $\mathcal{A}_1$ *is a* τ_1–reduct of $\mathcal{A}$*.*

4.2.1 Structures for Arithmetic

Let τ_{arith} be the vocabulary $\{F_+, F_*, R_<, c_0, c_1\}$ consisting of two binary function symbols, one binary relation symbol and two constant symbols. We have already defined two τ_{arith}-strcutures $\mathcal{N}$ and $\mathcal{Z}$, the arithmetic structure of the natural numbers and of the integers, cf. 4.1.5.

4.2.2: Exercise

(i) Show that $\mathcal{N}$ *is a minimal* τ_{arith}*–structure. Determine the vocabularies* $\tau \subseteq \tau_{arith}$ *such that the* τ*–reduct of* $\mathcal{N}$ *is still minimal and those such that not. In the latter case exhibit some proper substructures.*

(ii) Let $\mathcal{N}_0$ *be the reduct of* $\mathcal{N}$ *to the vocabulary* $\tau_{arith} \setminus \{c_1\}$*. Is* $\mathcal{N}_0$ *a minimal structure ? If not, exhibit some substructures.*

(iii) Show that $\mathcal{N}$ *is a substructure of* $\mathcal{Z}$*.*

(iv) Are there any structures $\mathcal{A}$ *such that* $\mathcal{N} \subseteq \mathcal{A} \subseteq \mathcal{Z}$ *?*

4.2.3: Definition (The Arithmetic Structure of the Real Numbers)

(i) The structure $\mathcal{R}$ is the τ_{arith}–structure given by
$\mathcal{R}(\mathbf{Var}) = \mathbf{R}$, *the real numbers,*
$\mathcal{R}(F_+)$ *the usual addition on the real numbers,*
$\mathcal{R}(F_*)$ *the usual multiplication on the real numbers,*
$\mathcal{R}(R_<)$ *the usual linear order on the real numbers,*
$\mathcal{R}(c_0) = 0$, $\mathcal{R}(c_1) = 1$.

(ii) Let τ_{Arith} be the vocabulary $\{F_+, F_*, R_<, c_0, c_1, F_-, F_{div}\}$ with F_-, F_{div} both unary function symbols. We expand $\mathcal{R}$ to a τ_{Arith}-structure $\mathcal{R}^{inv}$ by defining
$\mathcal{R}^{inv}(F_-)(a) = -a$ *and*
$\mathcal{R}^{inv}(F_{div})(a) = a^{-1}$ *for* $a \neq 0$ *and* $\mathcal{R}^{inv}(F_{div})(0) = 0$.

4.2.4: Exercise

(i) Show that $\mathcal{N} \sqsubseteq \mathcal{R}$. Why is it not true that $\mathcal{N} \sqsubseteq \mathcal{R}^{inv}$.

(ii) Determine the minimal substructures of $\mathcal{R}$ and $\mathcal{R}^{inv}$.

(iii) Let $I \subseteq \mathbf{R}$ be the set of irrational numbers. Determine the substructures of $\mathcal{R}$ and $\mathcal{R}^{inv}$ which are minimal over I.

4.2.5: Definition (The Arithmetic Structure of the Rational Numbers)

(i) The structure $\mathcal{Q}$ is the τ_{arith}–structure given by
$\mathcal{Q}(\mathbf{Var}) = \mathbf{Q}$, *the rational numbers,*
$\mathcal{Q}(F_+)$ *the usual addition on the rational numbers,*
$\mathcal{Q}(F_*)$ *the usual multiplication on the rational numbers,*
$\mathcal{Q}(R_<)$ *the usual linear order on the rational numbers,*
$\mathcal{Q}(c_0) = 0$, $\mathcal{Q}(c_1) = 1$.

(ii) For τ_{Arith} we expand $\mathcal{Q}$ to a τ_{Arith}-structure $\mathcal{Q}^{inv}$ by defining
$\mathcal{Q}^{inv}(F_-)(a) = -a$ *and*
$\mathcal{Q}^{inv}(F_{div})(a) = a^{-1}$ *for* $a \neq 0$ *and* $\mathcal{Q}^{inv}(F_{div})(0) = 0$.

4.2.6: Exercise

(i) Show that $\mathcal{N} \sqsubseteq \mathcal{Q}$. Why is it not true that $\mathcal{N} \sqsubseteq \mathcal{Q}^{inv}$.

(ii) Show that both $\mathcal{Q} \sqsubseteq \mathcal{R}$ and $\mathcal{Q}^{inv} \sqsubseteq \mathcal{R}^{inv}$.

(iii) Determine whether $\mathcal{N} \simeq \mathcal{Q}$ and $\mathcal{Q} \simeq \mathcal{R}$.

(iv) Determine the minimal substructures of $\mathcal{Q}$ and $\mathcal{Q}^{inv}$.

(v) Determine the vocabularies $\tau \subseteq \tau_{Arith}$ such that the τ–reduct of $\mathcal{Q}$ is still minimal and those such that not. In the latter case exhibit the minimal substructures.

4.2.7: Definition (The Powerset Structures)
Let A be a set. The structure $\mathcal{P}(A)$ is the τ_{arith}–structure given by
$\mathcal{P}(A)(\mathbf{Var}) = \wp(A)$, the subsets of A;
$\mathcal{P}(A)(F_+)$ the usual union of two sets,
$\mathcal{P}(A)(F_*)$ the usual intersection of two sets,
$\mathcal{P}(A)(R_<)$ the usual inclusion relation between sets,
$\mathcal{P}(A)(c_0) = \emptyset$, $\mathcal{P}(A)(c_1) = A$.

4.2.8: Exercise

(i) Determine the minimal substructures of $\mathcal{P}(A)$ for various choices of A.

(ii) Let $X = \{A_1, \ldots, A_k\}$ be a finite partition of a a set A. Determine the minimal substrcuture of $\mathcal{P}(A)$ over X.

Let $\tau_{field} \subseteq \tau_{arith}$ be the vocabulary $\{F_+, F_*, c_0, c_1\}$ and $\tau_{Field} \subseteq \tau_{Arith}$ be the vocabulary $\{F_+, F_*, c_0, c_1, F_-, F_{div}\}$.

4.2.9: Definition

(i) The structure $\mathcal{C}$ is the τ_{field}–structure given by
$\mathcal{C}(\mathbf{Var}) = \mathbf{C}$, the complex numbers,
$\mathcal{C}(F_+)$ the usual addition on the complex numbers,
$\mathcal{C}(F_*)$ the usual multiplication on the complex numbers,
$\mathcal{C}(c_0) = 0$, $\mathcal{C}(c_1) = 1$.

(ii) For τ_{Field} we expand $\mathcal{C}$ to a τ_{Field}-structure $\mathcal{C}^{inv}$ by defining
$\mathcal{C}^{inv}(F_-)(a) = -a$ and
$\mathcal{C}^{inv}(F_{div})(a) = a^{-1}$ for $a \neq 0$ and $\mathcal{C}^{inv}(F_{div})(0) = 0$.

(iii) Let $\mathcal{R}_{field}$ be the τ_{field}–reduct of $\mathcal{R}$ and Let $\mathcal{R}_{Field}$ be the τ_{Field}–reduct of $\mathcal{R}^{inv}$.

4.2.2 Graphs and Orders

Graphs are usually represented as $< V, E >$ where V is any set called the set of *vertices* and $E \subseteq V^2$ is called the set of *directed edges*. To describe graphs as structures in our sense, let τ_{graph} be the vocabulary consisting of one binary relation symbol R_E.

4.2.10: Definition (Graphs and Orders)

(i) *A* directed graph *is a* τ_{graph}*-structure* $\mathcal{G}$ *with* $\mathcal{G}(\mathbf{Var}) = V$ *the set of* vertices, $\mathcal{G}(R_E) = E$ *with* $E \subseteq V^2$ *the set of* edges.

(ii) *A* undirected graph *is a* τ_{graph}*-structure* $\mathcal{G}$ *with* $\mathcal{G}(\mathbf{Var}) = V$ *the set of* vertices, $\mathcal{G}(R_E) = E$ *with* $E \subseteq V^2$ *the set of* edges *with the property that for every* $a, b \in V$ $\langle a, b\rangle \in E$ *iff* $\langle b, a\rangle \in E$.

(iii) *A graph (directed or undirected)* $\mathcal{G}$ *is finite if* V *is finite.*

(iv) *A* directed graph *is a* partial order *if the relation* E *is transitive, i.e.* $\langle a, b\rangle \in E$ *and* $\langle b, c\rangle \in E$ *imply that* $\langle a, c\rangle \in E$, *and if* $\langle a, b\rangle \in E$ *and* $\langle b, a\rangle \in E$ *imply that* $a = b$.

(v) *A partial order is said to be* total (or linear) *if additionally we have that for every* $a, b \in V$ *either* $\langle a, b\rangle \in E$ *or* $\langle b, a\rangle \in E$.

4.2.11: Exercise
Draw some directed graphs by representing the set V *as points on a piece of paper and connecting two points* $a, b \in V$ *with an arrow from* a *to* b *iff* $\langle a, b\rangle \in E$. *Draw some undirected graphs similarly but connecting two points* $a, b \in V$ *with a line between* a *and* b *iff* $\langle a, b\rangle \in E$.

Similarly we can model *labelled graphs* and *relational data bases.*

4.2.3 Words and Sets of Words as Structures

Let τ_{word} be the vocabulary consisting of one binary relation symbol $R_<$ and one unary relation symbol R_U.

4.2.12: Definition (One word as a structure)

(i) *A* word *of length* n *is a* τ_{word}*-structure* $\mathcal{W}$ *with*
$\mathcal{W}(\mathbf{Var}) = \{1, 2, \ldots, n\}$,
$\mathcal{W}(R_<)$ *the natural order and*
$\mathcal{W}(R_U) \subseteq \mathcal{W}(\mathbf{Var})$.
We can visualize words over the letters a, b *as follows: the* k*-th letter of the word is* a *iff* $k \in \mathcal{W}(R_U)$.

(ii) *Let* $w \in \{a, b\}^*$ *be a word of length* n. *Let* $\mathcal{W}_w$ *be the* τ_{word}*-strcuture defined by*
$\mathcal{W}_w(\mathbf{Var}) = \{1, 2, \ldots, n\}$,
$\mathcal{W}_w(R_<)$ *the natural order and*
$\mathcal{W}_w(R_U) \subseteq \mathcal{W}(\mathbf{Var})$ *given by* $k \in \mathcal{W}_w(R_U)$ *iff the* k*th letter in* w *is* a.

4.2.13: Definition (The set of words A^* as a structure)
Let τ_{sg} *be the vocabulary consisting of a binary function symbol* $F_\circ$ *and a constant symbol* c_0. *Let* A *be a set and* A^* *be the set of finite words over*

A. We denote by $\mathcal{W}(A)$ *the* τ_{sg}*–structure defined by*
$\mathcal{W}(A)(\mathbf{Var}) = A^*a$,
$\mathcal{W}(A)(F_\circ)(x,y) = x \circ y$ *and*
$\mathcal{W}(A)(c_0) = \epsilon$, *the empty word.*

Let $\tau_{WFF} = \{F_\vee, F_\wedge, F_\rightarrow, F_\neg, R_\models, R_{taut}, c_{true}, c_{false}\}$ with $F_\vee, F_\wedge, F_\rightarrow$ binary function symbols, $F_\neg$ a unary function symbol, $R_\models$ a binary and R_{taut} a unary relation symbol and c_{true}, c_{false} constant symbols.

4.2.14: Definition (WFF as a structure)
Let $\mathcal{WFF}$ *be the* τ_{WFF}*–structure defined by*
$\mathcal{WFF}(\mathbf{Var}) = \mathbf{WFF}$,
$\mathcal{WFF}(F_\vee)(\phi,\psi) = (\phi \vee \psi)$,
$\mathcal{WFF}(F_\wedge)(\phi,\psi) = (\phi \wedge \psi)$,
$\mathcal{WFF}(F_\rightarrow)(\phi,\psi) = (\phi \rightarrow \psi)$,
$\mathcal{WFF}(F_\neg)(\phi) = \neg\phi$,
$\mathcal{WFF}(c_{false}) = \mathbf{F}$, $\mathcal{WFF}(c_{true}) = \mathbf{T}$,
$\mathcal{WFF}(R_{taut})$ *is the set of tautologies and*
$\langle\phi,\psi\rangle \in \mathcal{WFF}(R_\models)$ *iff* $\phi \models \psi$.

4.2.4 Data Structures of Computer Science

In this subsection we show one way of modelling the familiar data structures of Stacks, Queues and Lists as structures.

4.2.15: Definition (Stacks)
Let τ_{stacks} *be the vocabulary consisting of two unary function symbols* F_{pop}, F_{top}, *one ternary relation symbol* R_{push} *and a constant symbol* c_{new}. *Let* A *be a set.* $\mathcal{ST}(A)$ *is the* τ_{stacks}*–structure given by*
$\mathcal{ST}(A)(\mathbf{Var}) = A^*$;
$\mathcal{ST}(A)(F_{pop})(\epsilon) = \epsilon$ *and for every* $a \in A$ *and* $w \in A^*$;
$\mathcal{ST}(A)(F_{pop})(w \circ a) = w$;
$\mathcal{ST}(A)(F_{top})(\epsilon) = \epsilon$ *and for every* $a \in A$ *and* $w \in A^*$;
$\mathcal{ST}(A)(F_{top})(w \circ a) = a$;
$\mathcal{ST}(A)(R_{push}) \subseteq A^* \times A \times A^*$ *with* $\langle w,a,u\rangle \in \mathcal{ST}(A)(R_{push})$ *iff* $u = w \circ a$;
$\mathcal{ST}(A)(c_{new}) = \epsilon$, *the empty stack.*

4.2.16: Definition (Queues)
Let τ_{queue} *be the vocabulary consisting of two unary function symbols* F_{get}, F_{front}, *one ternary relation symbol* R_{put} *and a constant symbol* c_{new}. *Let* A *be a set.* $\mathcal{QU}(A)$ *is the* τ_{queue}*–structure given by*
$\mathcal{QU}(A)(\mathbf{Var}) = A^*$;
$\mathcal{QU}(A)(F_{get})(\epsilon) = \epsilon$ *and for every* $a \in A$ *and* $w \in A^*$;
$\mathcal{QU}(A)(F_{get})(w \circ a) = w$;
$\mathcal{QU}(A)(F_{front})(\epsilon) = \epsilon$ *and for every* $a \in A$ *and* $w \in A^*$;
$\mathcal{QU}(A)(F_{front})(w \circ a) = a$;

$\mathcal{QU}(A)(R_{put}) \subseteq A^* \times A \times A^*$ *with* $\langle w, a, u\rangle \in \mathcal{QU}(A)(R_{put})$ *iff* $u = a \circ w$;
$\mathcal{QU}(A)(c_{new}) = \epsilon$, *the empty queue.*

4.2.17: Remark
When dealing with data structures it is observed that both R_{push} *and* R_{put} *are interpreted as functions from* $A^* \times A$ *to* A^*. *In our definition of structures functions are* always total functions on the universe. *This is why we modelled their graph as ternary relations.*

Recall from section 2.5 that $CART(A)$ denotes the smallest set which contains A and is closed under the formation of ordered pairs.

4.2.18: Definition (Lists)
Let τ_{list} *be the vocabulary consisting of two unary function symbols* F_{head}, F_{tail}, *one binary function symbol* F_{cons} *and a constant symbol* c_0. *Let* A *be a set.* $\mathcal{LI}(A)$ *is the* τ_{list}*-structure given by*
$\mathcal{LI}(A)(\mathbf{Var}) = CART(A) \cup \{\emptyset\}$;
$\mathcal{LI}(A)(F_{head})(x) = \emptyset$ *for every* $x \in A \cup \{\emptyset\}$;
$\mathcal{LI}(A)(F_{head})(\langle x, y\rangle) = x$;
$\mathcal{LI}(A)(F_{tail})(x) = \emptyset$ *for every* $x \in A \cup \{\emptyset\}$;
$\mathcal{LI}(A)(F_{tail})(\langle x, y\rangle) = y$;
$\mathcal{LI}(A)(F_{cons})(x, y) = \langle x, y\rangle$;
$\mathcal{LI}(A)(c_0) = \emptyset$.

4.2.19: Exercise (Trees)
Define similarly trees

4.2.5 $\in$–Structures

In this subsection we introduce some structures modelling sets. Let τ_{set} be the vocabulary consisting of one binary relation symbol $R_\in$ and one constant symbol c_0.

4.2.20: Definition ($\in$–structures)

(i) Let A *be a set with* $\emptyset \in A$. *We denote by* $\mathcal{SET}(A)$ *the* τ_{set}*-structure with* $\mathcal{SET}(A)(\mathbf{Var}) = A$, $\mathcal{SET}(A)(c_0) = \emptyset$ *and* $\langle x, y\rangle \in \mathcal{SET}(A)(R_\in)$ *iff* $x \in y$.

(ii) For $\mathbf{HF}(A)$ *we write* $\mathcal{HF}(A)$ *instead of* $\mathcal{SET}(\mathbf{HF}(A))$. *In particular, for* $\mathbf{HF}$ *we write* $\mathcal{HF}$ *instead of* $\mathcal{SET}(\mathbf{HF})$.

4.3 Syntax and Semantics of First Order Logic

In this section we define the syntax and semantics of First Order Logic with equality.

4.3.1 Syntax of First Order Logic

4.3.1: Definition (Logical Symbols)
Logical symbols *are elements of the set* $Logsymb = \{\wedge, \vee, \rightarrow, \neg, \forall, \exists, \approx\}$. Separator symbols *are elements of the set* $Sepsymb = \{(,), ,\}$.

$\wedge$ *is read as 'and',* $\vee$ *is read as 'or',* $\neg$ *is read as 'not',* $\forall$ *is read as 'for all' and is called* universal quantifier, $\exists$ *is read as 'there is' and is called* existential quantifier, $\approx$ *is read as 'equals' and* $\rightarrow$ *is read as 'arrow'. We avoid reading* $\rightarrow$ *as 'implies', as sometimes suggested in the literature.*

4.3.2: Definition (Variables)
For every $i \in \mathbf{N}$, v_i *is a* variable. *The set of all variables is denoted by* **Var**. *Note that* **Var** *is the dummy symbol we had introduced in definition 4.1.4.*

Well–formed formulas of First Order Logic are strings containing symbols which are in $Logsymb \cup Sepsymb \cup \mathbf{Var} \cup \tau_\omega$.

4.3.3: Notation
We shall use a vector notation for variables. Thus $\vec{x}$ *stands for* $(x_1, \ldots, x_k)$ *for some variables* $x_1 = v_{j_1}, \ldots, x_k = v_{j_k}$.

We first define the set of τ-terms. Terms are complex names of elements, built from variables, constant symbols and function symbols.

4.3.4: Definition (τ-Terms)
Let τ *be a vocabulary. The set* $Terms(\tau)$ *of* τ-terms *is defined inductively as follows:*

Basis:

(i) $\mathbf{Var} \subseteq Terms(\tau)$.

(ii) For every $c_\alpha \in \tau$ *we have* $c_\alpha \in Terms(\tau)$.

Closure: *If* $t_1, \ldots, t_n$ *are terms and* $F_{n,\alpha} \in \tau$ *then* $F_{n,\alpha}(t_1, \ldots, t_n) \in Terms(\tau)$.

The variables occuring in the string $t \in Terms(\tau)$ *are called* free *variables. We denote the set of free variables in* t *by* $Free(t)$.

4.3.5: Notation
We shall also use vectorial notation for terms. For $\vec{t} = (t_1, \ldots, t_n)$ *we then can write instead of* $F_{n,\alpha}(t_1, \ldots, t_n)$ *simply* $F_{n,\alpha}(\vec{t})$.

4.3.6: Theorem (Unique Readability for Terms, an Exercise)
Let $t \in Term(\tau)$. *Then one of the following is true:*

(i) $t = c_\alpha$ *for some unique* $c_\alpha \in \tau$;

(ii) $t = v_i$ *for some unique* $v_i \in \mathbf{Var}$;

(iii) There is a unique $F_{n,\alpha} \in \tau$ *and there are unique terms* $t_1, \ldots, t_n$ *such that* $t = F_{n,\alpha}(t_1, \ldots, t_n)$.

Proof. The proof is very similar to the corresponding theorem for propositional logic. Note however, that we have not bothered to define the tree version of terms. The various statements in the theorem do correspond to the nodes of the tree, and the induction proceeds along those statements. ∎

4.3.7: Definition (Atomic τ-FOL-Formulas)

(i) If t_1 *and* t_2 *are* τ*-Terms then* $(t_1 \approx t_2)$ *is an* atomic τ-FOL-formula, *and* $Free(t_1 \approx t_2) = Free(t_1) \cup Free(t_2)$

(ii) If $R_{j,\alpha} \in \tau$ *and* $t_1, \ldots, t_j$ *are* τ*-Terms then* $R_{j,\alpha}(t_1 \ldots, t_j)$ *is an* atomic τ-FOL-formula, *and* $Free\, R_{j,\alpha}(t_1 \ldots, t_j) = Free\,(t_1) \cup \cdots \cup Free\,(t_j)$

4.3.8: Definition (τ-FOL-Formulas)

(i) If ϕ *is an atomic* τ*-FOL-formula then* ϕ *is a* τ-FOL-formula.

(ii) If ϕ *is a* τ*-FOL-formula then* $\neg\phi$ *is an* τ-FOL-formula, *and*

$$Free\,((\neg\phi)) = Free\,(\phi)$$

(iii) If ϕ *and* ψ *are* τ*-FOL-formulas then* $(\phi \vee \psi)$, $(\phi \wedge \psi)$, $(\phi \to \psi)$, *are all* τ-FOL-formulas, *and* $Free\,((\phi \vee \psi)) = Free\,((\phi \wedge \psi)) = Free\,((\phi \to \psi)) = Free\,(\phi) \cup Free\,(\psi)$.

(iv) If ϕ *is a* τ*-FOL-formula and* $x \in$ **Var** *is a variable then* $(\forall x\phi)$ *and* $(\exists x\phi)$ *are* τ-FOL-formulas, *and* $Free\,((\forall x\phi)) = Free\,((\exists x\phi)) = Free\,(\phi) \setminus \{x\}$.
We say also, that all of the occurences of x *in* ϕ *are* bound *by the quantifier* $\forall$ *or* $\exists$.

We denote the set of τ*-FOL–formulas by* $FOL(\tau)$.

4.3.9: Remark
Although our strict formation rules of FOL–formulas requires that the application of a quantifier is followed by enclosing the resulting formula with parentheses, we shall later drop this convention. Hence, in liberalized form,

*if ϕ is a τ-FOL-formula and $x \in$ **Var** is a variable then $\forall x\phi$ and $\exists x\phi$ are* τ-FOL-formulas.

4.3.10: Definition (Sentences)
If ϕ is a τ-FOL-formula and $Free(\phi) = \emptyset$, then ϕ is a τ-FOL-sentence. We denote the set of τ-FOL–sentences by $SENT(\tau)$.

4.3.11: Definition (Subformulas)
Let ϕ, ψ be τ-FOL-formulas. ψ is a subformula *of ϕ if there are words α, β such that $\alpha \circ \psi \circ \beta = \phi$*

4.3.12: Theorem (Unique Readability for Formulas, an Exercise)
Let ϕ be a τ-formula. Then one of the following is true:

(i) ϕ is an atomic formula and begins with a relation symbol $R_{n,\alpha} \in \tau$.

(ii) ϕ is an atomic formula and begins with (followed by some term t.

(iii) ϕ begins with $(\forall$ and there is a unique variable v_i and a unique subformula ψ of ϕ such that $\phi = (\forall v_i \psi)$;

(iv) ϕ begins with $(\exists$ and there is a unique variable v_i and a unique subformula ψ of ϕ such that $\phi = (\exists v_i \psi)$;

(v) ϕ begins with $\neg$ and there is a unique subformula ψ of ϕ such that $\phi = \neg\psi$;

(vi) ϕ begins with (and there are unique subformulas ψ_1, ψ_2 such that either $\phi = (\psi_1 \vee \psi_2)$ or $\phi = (\psi_1 \wedge \psi_2)$; $\phi = (\psi_1 \to \psi_2)$.

Proof. Use the Unique Readability for terms and proceed by induction on the construction of the formula. We have not bothered to give the tree presentation of formulas, but propose it in the exercise below. ∎

4.3.13: Remark
The reader should convince himself that the unique readability also holds for the liberalized notion of FOL–formulas.

4.3.14: Theorem (Exercise:)

(i) Define, analogously as for **WFF** *and* **WFTF**, *the set of τ-tree–formulas $\tau -$* **TFOF**. *The case of the quantifiers requires some kind of additional marking of the variables to be bound which requires two kinds of links.*

(ii) Define writing and reading of formulas as the translations of $\tau -$ **TFOF** *into $\tau -$* **FOF** *and vice versa.*

(iii) Using the above definitions state and prove the Unique Readability Theorem for First Order Logic.

4.3.15: Remark
Note that the notion of τ-FOL–formulas is context free. However, checking whether a variable v_i is free (or bound) in some formula ϕ is not context free but context sensitive.

4.3.2 Semantics of First Order Logic

Semantics is the solid foundation of meaning. In the case of First Order Logic meaning depends on a given τ-structure $\mathcal{A}$ and is assigned to τ-formulas by means of a *meaning function*. This meaning function is built in stages. We first give meaning to the variables by an *assignment* z which maps variables into the domain of the structure $\mathcal{A}$. Then we extend this assignment function to give meaning to τ-terms. Finally we define the meaning function M with three arguments, which depends on a formula ϕ, a structure $\mathcal{A}$ and an assignment z, and has the boolean values 0 or 1. In detail this looks as follows:

4.3.16: Definition (Variable Assignment)

(i) Given a τ-structure $\mathcal{A}$ with domain $\mathcal{A}(\mathbf{Var}) = A$ an assignment *z is a mapping $z : \mathbf{Var} \mapsto A$ from the set of variables to elements of the domain.*

(ii) We denote the set of all assignments $z : \mathbf{Var} \mapsto A$ by $Ass(\mathcal{A})$.

(iii) Let z_1 and z_2 be two assignments of $Ass(\mathcal{A})$. We write $z_1 =_i z_2$ if for every $j \neq i$ we have that $z_1(v_j) = z_2(v_j)$.

4.3.17: Definition
Given a τ-structure $\mathcal{A}$ and an assignment z, the meaning function for τ-terms is a function $MT : Terms(\tau) \times Ass(\mathcal{A}) \mapsto A$, defined inductively as follows:

Basis:

(i) $MT(v_i, z) = z(v_i)$;

(ii) If $c_\alpha \in \tau$ then $MT(c_\alpha, z) = c_\alpha$;

Closure:
If $F_{n,\alpha} \in \tau$, $t_1, \ldots, t_n$ and t are τ-terms with $t = F_{n,\alpha}(t_1, \ldots, t_n)$ then

$$MT(t, z) = \mathcal{A}(F_{n,\alpha})(MT(t_1, z), \ldots, MT(t_n, z)).$$

4.3.18: Exercise
Write down some examples.

4.3.19: Definition
The semantic for our languages is described by means of the meaning function $M(\phi, \mathcal{A}, z)$ *where* ϕ *is a formula,* $\mathcal{A}$ *a structure and* z *an assignment. The meaning of the formulas is defined inductively as follows:*

Basis:

(i) $M((t_1 \approx t_2), \mathcal{A}, z) = 1$ *iff* $MT(t_1, z) = MT(t_2, z)$;

(ii) $M(R_{j,\alpha}(t_1 \ldots, t_j), \mathcal{A}, z) = 1$ *iff* $(MT(t_1, z), \ldots, MT(t_j, z)) \in \mathcal{A}(R_{j,\alpha})$;

Closure:

(i) $M(\neg\phi, \mathcal{A}, z) = TT_{\neg}(M(\phi, \mathcal{A}, z))$

(ii) $M((\phi \vee \psi), \mathcal{A}, z) = TT_{\vee}(M(\phi, \mathcal{A}, z), M(\psi, \mathcal{A}, z))$;

(iii) $M((\phi \wedge \psi), \mathcal{A}, z) = TT_{\wedge}(M(\phi, \mathcal{A}, z), M(\psi, \mathcal{A}, z))$;

(iv) $M((\phi \rightarrow \psi), \mathcal{A}, z) = TT_{\rightarrow}(M(\phi, \mathcal{A}, z), M(\psi, \mathcal{A}, z))$;

(v) $M((\exists v_i \phi), \mathcal{A}, z) = 1$ *iff there exists* $z_1 \in Ass(\mathcal{A})$ *such that* $z =_i z_1$ *and* $M(\phi, \mathcal{A}, z_1) = 1$;

(vi) $M((\forall v_i \phi), \mathcal{A}, z) = 1$ *if for every* $z_1 \in Ass(\mathcal{A})$ *such that* $z =_i z_1$ *we have that* $M(\phi, \mathcal{A}, z_1) = 1$

4.3.20: Remark
Note that this definition works because of the Unique Readability Theorem for First Order Logic.

4.3.21: Proposition
If ϕ *is a sentence, then* $M(\phi, \mathcal{A}, z)$ *is independent of* z, *i.e. for every* $z, z_1 \in Ass(\mathcal{A})$ $M(\phi, \mathcal{A}, z) = M(\phi, \mathcal{A}, z_1)$

The next theorem states that isomorphic τ-structures can not be distinguished by τ-formulas. In other words, the meaning functions MT and M are not sensitive to change of structures by isomorphic copies.

4.3.22: Theorem (The Isomorphisms Property)
Let $\mathcal{A}, \mathcal{B}$ *be two* τ*-structures,* $X \subseteq \mathcal{A}(\mathbf{Var})$ *and* $Y \subseteq \mathcal{B}(\mathbf{Var})$. *Let* $f : X \mapsto Y$ *be a partial isomorphism with domain* X *and range* Y. *Let* $z : \mathbf{Var} \mapsto X$ *be an assignment and* z_1 *be the assignment defined by* $z_1(v_i) = f(z(v_i))$.

(i) Let t *be a* τ*-term. Then* $f(MT(t, \mathcal{A}, z)) = MT(t, \mathcal{B}, z_1)$.

(ii) Let ϕ *be a* τ*-formula. Then* $M(\phi, \mathcal{A}, z) = M(\phi, \mathcal{B}, z_1)$.

4.4 Basic Semantic Concepts

In this section we present the basic semantic concepts such as logical consequence, logical equivalence, satisfiability and tautologies for first order logic. As particular examples and applications of the notion of logical equivalence we study the effect of substitution of subformulas and some normal forms for first order logic.

4.4.1 Validity, Logical Equivalence and Logical Consequence

4.4.1: Definition
Let Σ be a set of τ-formulas and ϕ and ψ be two τ-formulas.

(i) Let $\mathcal{A}$ be a τ-structure and z be an assignment. We extend the meaning function for formulas to sets of formulas by setting $M(\Sigma, \mathcal{A}, z) = 1$ iff for every $\phi \in \Sigma$ we have that $M(\phi, \mathcal{A}, z) = 1$.

(ii) We say that ϕ is a logical consequence *of Σ, and we write $\Sigma \models \phi$, if for every τ-structure $\mathcal{A}$ and every assignment z such that $M(\Sigma, \mathcal{A}, z) = 1$ we have also that $M(\phi, \mathcal{A}, z) = 1$. If $\Sigma = \emptyset$ we write simply $\models \phi$ instead of $\emptyset \models \phi$. If $\Sigma = \{\psi\}$ is a singleton we write, by abuse of notation, also $\psi \models \phi$ instead of $\{\psi\} \models \phi$.*

(iii) Similarly we write $\Sigma \models \Sigma_1$ if $\Sigma \models \phi$ for every $\phi \in \Sigma_1$.

(iv) We say that ϕ and ψ are logically equivalent, and we write $\phi \equiv \psi$, if $\phi \models \psi$ and if $\psi \models \phi$.

(v) ϕ is a tautology (or valid) if it is a consequence of the empty set, i.e. $\models \phi$.

(vi) Σ is satisfiable if there is a τ-structure $\mathcal{A}$ and an assignment z such that $M(\Sigma, \mathcal{A}, z) = 1$

4.4.2: Notation
Let Σ, Σ_1 be sets of τ-formulas, $\mathcal{A}$ be a τ-structure and z be an assignment.

(i) We write $\mathcal{A}, z \models \Sigma_1$ for $M(\Sigma_1, \mathcal{A}, z) = 1$ and $\mathcal{A} \models \Sigma_1$ if for every assignment z $M(\Sigma_1, \mathcal{A}, z) = 1$.

(ii) We write $\Sigma \models \Sigma_1$ if for every $\mathcal{A}$ with $\mathcal{A} \models \Sigma$ we have also $\mathcal{A} \models \Sigma_1$.

(iii) Note that this notation is consistent with $\Sigma \models \Sigma_1$ defined above via logical consequence.

4.4.3: Proposition

(i) ϕ is a tautology iff $\neg\phi$ is not satisfiable.

(ii) $\Sigma \models (\phi \to \psi)$ *iff* $\Sigma \cup \{\phi\} \models \psi$.

(iii) $\Sigma \models (\phi \wedge \neg\phi)$ *iff* Σ *is not satisfiable.*

(iv) Σ *is not satisfiable iff for every* ϕ *we have that* $\Sigma \models \phi$.

(v) $\phi \equiv \psi$ *iff both* $\models (\phi \to \psi)$ *and* $\models (\psi \to \phi)$

4.4.4: Proposition (Some Tautologies and Equivalences)

(i) Let $B(p_1, \ldots, p_n)$ *be a propositional formula in* **WFF** *which is a propositional tautology. Let* $\phi_1, \ldots, \phi_n$ *be* τ*-formulas. Let* ϕ *be the* τ*-formula* $B(\phi_1, \ldots, \phi_n)$ *obtained from* B *by replacing each occurrence of the propositional variable* p_i *by* ϕ_i. *Then* ϕ *is a tautology (among first order formulas).*

(ii) Let ϕ_1, ψ_1 *and* ψ_2 *be* τ*-formulas such that* ψ_1 *is a subformula of* ϕ_1 *and* $\psi_1 \equiv \psi_2$. *Let* ϕ_2 *be the formula obtained from* ϕ_1 *by replacing* ψ_1 *by* ψ_2. *Then* $\phi_1 \equiv \phi_2$.

Proof. (i) First we prove by induction on the structure of B that $M(\phi, \mathcal{A}, z) = M_{PL}(B, z_1)$ where z_1 is the propositional assignment defined by $z_1(p_i) = M(\phi_i, \mathcal{A}, z)$. Then we use that B is a propositional tautology to conclude that ϕ is a first order tautology.
(ii) Let $\phi_1 = \alpha \circ \psi_1 \circ \beta$. We proceed by backward induction on α and induction on β and use the Unique Readability Theorem. ∎

4.4.5: Proposition (Moving quantifiers in formulas)
Let ϕ, ψ *be* τ*-formulas. Then*

(i) $\forall v_i(\phi \wedge \psi) \equiv (\forall v_i\phi \wedge \forall v_i\psi)$;

(ii) $\exists v_i(\phi \vee \psi) \equiv (\exists v_i\phi \vee \exists v_i\psi)$;

(iii) If v_i *does not occur free in* ψ *then* $(\forall v_i\phi \wedge \psi) \equiv \forall v_i(\phi \wedge \psi)$ *and* $(\forall v_i\phi \vee \psi) \equiv \forall v_i(\phi \vee \psi)$;

(iv) If v_i *does not occur free in* ψ *then* $(\exists v_i\phi \vee \psi) \equiv \exists v_i(\phi \vee \psi)$ *and* $(\exists v_i\phi \wedge \psi) \equiv \exists v_i(\phi \wedge \psi)$;

(v) $\neg\forall v_i\phi \equiv \exists v_i\neg\phi$ *and* $\neg\exists v_i\phi \equiv \forall v_i\neg\phi$;

4.4.6: Definition (Substitution of Terms)
Let s : **Var** $\mapsto Term(\tau)$ *be a function. Let* $t \in Term(\tau)$ *and* ϕ *be a* τ*-formula. We define* $st(t, s)$ *and* $subst(\phi, s)$ *inductively in two stages.*

(i) $st(c_\alpha, s) = c_\alpha$;

(ii) $st(v_i, s) = s(v_i)$;

(iii) $st(F_{n,\alpha}(t_1, \ldots, t_n), s) = F_{n,\alpha}(st(t_1, s), \ldots, st(t_n, s))$;

(iv) $subst((t_1 \approx t_2), s) = st(t_1, s) \approx st(t_2, s)$;

(v) $subst(R_{n,\alpha}(t_1, \ldots, t_n), s) = R_{n,\alpha}(st(t_1, s), \ldots, st(t_n, s))$;

(vi) $subst(\neg\phi, s) = \neg subst(\phi, s)$;

(vii) $subst((\phi \wedge \psi), s) = (subst(\phi, s) \wedge subst(\psi, s))$;

(viii) $subst((\phi \vee \psi), s) = (subst(\phi, s) \vee subst(\psi, s))$;

(ix) $subst((\phi \to \psi), s) = (subst(\phi, s) \to subst(\psi, s))$;

(x) $subst(\forall v_i \phi, s) = \forall v_i subst(\phi, s_1)$ *with* $s_1(v_i) = v_i$ *and* $s_1(v_j) = s(v_j)$ *for all* $i \neq j$.

(xi) $subst(\exists v_i \phi, s) = \exists v_i subst(\phi, s_1)$ *with* $s_1(v_i) = v_i$ *and* $s_1(v_j) = s(v_j)$ *for all* $i \neq j$.

4.4.7: Proposition (Renaming of Bound Variables)
Let $s : \mathbf{Var} \mapsto \mathbf{Var}$ *be a function and* ϕ *a* τ*-formula. Assume that* $s(v_j)$ *does not occur at all in* ϕ *and* $s(v_i) = v_i$ *for every* $i \neq j$. *Then*

(i) $\forall v_j \phi \equiv \forall s(v_j) subst(\phi, s)$ *and*

(ii) $\exists v_j \phi \equiv \exists s(v_j) subst(\phi, s)$.

Proof. The proof uses the fact that $M(\phi, \mathcal{A}, z)$ is not dependent on $z(s(v_j))$. ∎

The condition that $s(v_j)$ does not occur at all in ϕ in proposition 4.4.7 is not the weakest possible.

4.4.8: Exercise
Analyse what other conditions on the occurrence of $s(v_j)$ *in* ϕ *suffice for the conclusion of proposition 4.4.7 to remain true. Is the condition that* $s(v_j)$ *is not in* $Free(\phi)$ *sufficient ?*

Note that in the proof rules in section 4.6 additional conditions on substitutions are imposed.

4.4.9: Proposition (Replacing equivalent subformulas)
Let ϕ *be a formula with subformula* θ, *i.e.* $\phi = \alpha \circ \theta \circ \beta$. *Let* θ_1 *be a formula such that* $\theta \equiv \theta_1$ *and* $Free(theta) = Free(theta_1)$. *Then*

(i) $\phi_1 = \alpha \circ \theta_1 \circ \beta$ *is a well formed formula and*

(ii) $\phi \equiv \phi_1$.

Proof. Use unique readability and induction on the structure of ϕ. ∎

4.4.2 Normal Forms

First order formulas are built by using the boolean operations and quantification in arbitrary order. The purpose of Normal Forms is to restrict the order of these building steps, similar as in the case of **CNF** for propositional logic. The most important normal form for first order logic is the Prenex Normal Form. The word "prenex" refers to "pre" (latin: before) and "nexus" (latin: bound) and is used in logic to indicate that all the variable binding appears in the formula before any other logical operations.

4.4.10: Definition (Quantifier free formulas)
The set $\mathbf{QF}(\tau)$ *of* quantifier free τ-formulas *is built inductively as follows:*
Basis: *Atomic* τ*-formulas are quantifier free.*
Closure: *If* $\phi, \psi \in \mathbf{QF}(\tau)$ *then so are* $\neg\phi, (\phi \wedge \psi), (\phi \vee \psi), (\phi \rightarrow \psi)$.

4.4.11: Definition (Prenex Normal Form)
The set $\mathbf{PNF}(\tau)$ *of* τ-formulas in prenex normal form *is built inductively as follows:*
Basis: $\mathbf{QF}(\tau) \subseteq \mathbf{PNF}(\tau)$.
Closure: *If* $\phi \in \mathbf{PNF}(\tau)$ *then so are* $\forall v_i \phi$ *and* $\exists v_i \phi$.

4.4.12: Theorem (Prenex Normal Form Theorem)
For every τ*-formula* ϕ *there is a* τ*-formula* ψ *such that:*

(i) $\phi \equiv \psi$;

(ii) $\psi \in \mathbf{PNF}(\tau)$ *and*

(iii) ϕ *and* ψ *have the same free variables.*

(iv) Furthermore the length of ψ *is linear in the length of* ϕ.

Proof. The proof is by induction on ϕ. If $\phi \in \mathbf{PNF}(\tau)$ there is nothing to prove. In the other cases we use the proposition on moving quantifiers 4.4.5, renaming bound variables 4.4.7 and replacing equivalent subformulas 4.4.9. ∎

4.4.13: Definition (Universal and Existential Formulas)
A τ*–formula* ϕ *in* $\mathbf{PNF}(\tau)$ *which has only universal (existential) quantifiers is called a* universal (existential) formula.

4.4.3 Models and Theories *

In this section we introduce some definitions useful in the discussion of examples.

4.4.14: Definition (Definable Classes of Models)

(i) *We denote by* $Str(\tau)$ *the class of all* τ*–structures.*

(ii) *For every set of* τ*–sentences* Σ *we denote by*

$$MOD(\Sigma) = \{\mathcal{A} \in Str(\tau) : \textit{for every assignment } z\ M(\Sigma, \mathcal{A}, z) = 1\}$$

$MOD(\Sigma)$ *is called the* class of all models of Σ*.*

(iii) *Let* $K \subseteq Str(\tau)$*.* K *is called* (finitely) definable *if there is a (finite) set* Σ *of* τ*–sentences such that* $K = MOD(\Sigma)$*.*

Note that $\mathcal{A} \in MOD(\Sigma)$ is equivalent to $\mathcal{A} \models \Sigma$.

4.4.15: Definition (Elementary equivalence and elementary substructures)
Let $\mathcal{A}, \mathcal{B} \in Str(\tau)$*.*

(i) $\mathcal{A}, \mathcal{B}$ *are said to be* elementarily equivalent*, and we write* $\mathcal{A} \equiv \mathcal{B}$*, if for every* $\phi \in SENT(\tau)$ $\mathcal{A} \models \phi$ *iff* $\mathcal{B} \models \phi$*.*

(ii) $\mathcal{A}$ *are said to be* elementarily substructure of $\mathcal{B}$*, and we write* $\mathcal{A} \prec \mathcal{B}$*, if* $\mathcal{A}$ *is a substructure of* $\mathcal{B}$ *and for every assignment* $z : \mathbf{Var} \to \mathcal{A}(\mathbf{Var})$ *and for every* $\phi \in FOL(\tau)$ $\mathcal{A}, z \models \phi$ *iff* $\mathcal{B}, z \models \phi$*.*

4.5 Visit to the Museum: The Meaning Function and Definability

In this section we discuss some examples of what formulas can say. Learning how to read and understand the language of First Order Logic is essential for the intuitive grasp of logic and its mathematical treatment. But it should not be confused with reasoning about First Order Logic (and other logics). The former is like using a language to talk and communicate, the latter is like reasoning about such languages. And as the communication of reasoning about languages is done in language, so the reasoning about logic is done in mathematical language which again can be modelled in logic.

4.5.1 The Logical Lieu

Logical formulas say something about the structures in which they hold. If the τ–formula ϕ has no free variables and $\mathcal{A}$ is a τ–structure, the value of $M(\phi, \mathcal{A}, z)$ is

If the τ–formula $\phi(v_1, \ldots, v_n)$ has exactly $v_1, \ldots, v_n$ as its free variables and $\mathcal{A}$ is a τ–structure, the value of $M(\phi, \mathcal{A}, z)$ depends on $z(v_1), \ldots, z(v_n)$ and defines a subset of $\mathcal{A}(\mathbf{Var})^n$, similarly to $x^2 + y^2 + z^2 = 1$, which defines the unit circle in $\mathbf{R}^3$. The latter is called in elemntary geometry the 'geometrical lieu' defined by $x^2+y^2+z^2 = 1$. In analogy to this we now introduce the 'logical lieu' defined by a first order formula in a structure.

4.5.1: Definition (Logical Lieu)
Let $\phi(v_1, \ldots, v_n)$ a τ–formula with exactly $v_1, \ldots, v_n$ as its free variables and $\mathcal{A}$ be a τ–structure.

(i) We denote by $\phi(\mathcal{A})$ the set

$$\{\langle a_1, \ldots, a_n\rangle \in \mathcal{A}(\mathbf{Var})^n :$$

$$\textit{there is a } z \textit{ with } M(\phi, \mathcal{A}, z) = 1 \textit{ and } z(v_1) = a_1, \ldots, z(v_n) = a_n\}.$$

(ii) Let $X \subseteq \mathcal{A}(\mathbf{Var})^n$. We say that ϕ defines the set *X over $\mathcal{A}$ or that X is the* logical lieu *defined by ϕ over $\mathcal{A}$ if $X = \phi(\mathcal{A})$.*

The rest of this section is devoted to examples.

4.5.2 Ordered Fields

Recall that τ_{arith} is the vocabulary $\{F_+, F_*, R_<, c_0, c_1\}$ and $\mathcal{R}$ $(\mathcal{Q}, \mathcal{N})$ is the Arithmetic Structure of the Real (Rational, Natural) Numbers and $\mathcal{Z}$ of the Integers. The structures $\mathcal{R}$ and $\mathcal{Q}$ are what is called a *field*, i.e. addition and multiplication satisfy the following (set theoretic) properties:

Commutativity $x + y = y + x, x * y = y * x$;

Associativity $x + (y + z) = (x + y) + z, x * (y * z) = (x * y) * z$;

Neutral Element $0 + x = x, 1 * x = x, 0 * x = 0$;

Inverse Element For every x there is a y such that $x + y = 0$,
for every $x \neq 0$ there is a y such that $x * y = 1$;

Distributivity $x * (y + z) = (x * y) + (x * z)$.

4.5.2: Exercise

(i) Translate the above properties into τ_{arith}-formulas and verify that your translation is satisfied both in $\mathcal{R}$ and $\mathcal{Q}$.

(ii) Which of the properties have a translation which is not in Prenex Normal Form ? Can you find a translation which is in Prenex Normal Form?

(iii) Which of the above properties are not satisfied in $\mathcal{N}$?

(iv) Which of the above properties are not satisfied in $\mathcal{Z}$?

The structures $\mathcal{R}$ and $\mathcal{Q}$ are both what is called an *ordered field*, i.e. addition and multiplication satisfy the following additional (set theoretic) properties:

Asymmetry If $x < y$ then not $y < x$;

Transitivity if $x < y$ and $y < z$ then $x < z$;

Linearity for every x, y either $x < y$ or $y < x$ or $x = y$;

Monotonicity If $x < y$ then for every z also $x + z < y + z$;
if $0 < x$ and $0 < y$ then also $0 < x * y$.

4.5.3: Exercise

(i) Translate the above properties into τ_{arith}-formulas and verify that your translation is satisfied both in $\mathcal{R}$ and $\mathcal{Q}$.

(ii) Which of the properties have a translation which is not in Prenex Normal Form ? Can you find a translation which is in Prenex Normal Form?

(iii) Which of the above properties are not satisfied in $\mathcal{N}$?

(iv) Which of the above properties are not satisfied in $\mathcal{Z}$?

4.5.4: Exercise

(i) Find a τ_{field}–formula $\phi_{pos}(v_1)$ with one free variable (and without $R_<$) which defines the positive elements of $\mathcal{R}$. (Hint: The squares)

(ii) Find a τ_{field}–formula $\phi_{ord}(v_1, v_2)$ with two free variables (and without $R_<$) which defines the linear order on $\mathcal{R}$. (Hint: Use ϕ_{pos})

(iii) Determine the logical lieu of ϕ_{pos} and ϕ_{ord} in $\mathcal{Q}, \mathcal{Z}, \mathcal{N}$.

4.5.3 The Natural Numbers

Let τ_{peano} be the vocabulary $\{F_+, F_*, S, c_0, c_1\}$ consisting of two binary function symbols, one unary function symbol and two constant symbols. Let $\mathcal{N}_{peano}$ be the τ_{peano}–structure defined as follows: $\mathcal{N}_{peano}(\mathbf{Var}) = \mathbf{N}$,
$\mathcal{N}_{peano}(F_+)$ the usual addition on the natural numbers,
$\mathcal{N}_{peano}(F_*)$ the usual multiplication on the natural numbers,
$\mathcal{N}_{peano}(F_{succ})$ the usual successor function $(+1)$ on the natural numbers,
$\mathcal{N}_{peano}(c_0) = 0$, $\mathcal{N}_{peano}(c_1) = 1$.

4.5.5: Exercise

(i) Find a formula $\phi \in \mathbf{FOL}(\tau_{peano})$ which defines the usual order of the natural numbers on $\mathcal{N}_{peano}$.

(ii) Find a formula $\phi \in \mathbf{FOL}(\tau_{arith})$ which defines the successor function of the natural numbers on $\mathcal{N}$.

(iii) Show that for every formula $\phi \in \mathbf{FOL}(\tau_{peano})$ there is formula $\psi \in \mathbf{FOL}(\tau_{arith})$ with the same free variables such that for every assignment $z : \mathbf{Var} \to \mathbf{N}$ we have that $M(\phi, \mathcal{N}_{peano}, z) = M(\psi, \mathcal{N}, z)$.

4.5.6: Exercise

(i) Find a formula $\phi \in \mathbf{FOL}(\tau_{peano})$ with two free variables which defines the usual divisibility relation 'm divides n'.

(ii) Find a formula $\phi \in \mathbf{FOL}(\tau_{arith})$ with one free variable which defines the set of primes.

(iii) Find a formula $\phi \in \mathbf{FOL}(\tau_{peano})$ without free variables which expresses that there are infinitely many primes in $\mathbf{N}$.

The natural numbers satisfy the *Induction Principle*:
For every $X \subseteq \mathbf{N}$ such that $0 \in X$ and whenever $n \in X$ the $n + 1 \in X$, then $X = \mathbf{N}$. We shall see in section 4.7 that the Induction Principle is not expressible in first order logic. The best we can do in first order logic in describing the Induction Principle consists of writing down each instance for it for which X is definable as a formula.

Let $\phi(v_0, v_1, \ldots, v_k)$ be a τ_{peano} formula with $v_0, v_1, \ldots, v_k$ as its free variables. Let Ind_ϕ be the following formula:

$$\forall v_1, \ldots, \forall v_k[(\phi(c_0, v_1, \ldots, v_k) \wedge (\forall v_0 \phi(v_0, v_1, \ldots, v_k) \to$$

$$\phi(F_s ucc(v_0), v_1, \ldots, v_k))) \to \forall v_0 \phi(v_0, v_1, \ldots, v_k)].$$

4.5.7: Exercise

(i) Verify that $\mathcal{N}_{peano} \models Ind_\phi$ *for every* τ_{peano} *formula.*

(ii) Let $Ind^+{}_\phi$ *be the following formula:*

$$\forall v_1, \ldots, \forall v_k[(\phi(c_0, v_1, \ldots, v_k) \wedge (\forall v_0 \phi(v_0, v_1, \ldots, v_k) \rightarrow$$

$$\phi(F_+(v_0, c_1), v_1, \ldots, v_k))) \rightarrow \forall v_0 \phi(v_0, v_1, \ldots, v_k)].$$

Verify that $\mathcal{N}_{peano} \models Ind^+{}_\phi$ *for every* τ_{peano} *formula.*

4.5.4 Graphs and Orders

Let $\mathcal{G} =< V, E >$ be a graph.

4.5.8: Exercise

(i) Find a τ_{graph}*-formula without free variables which says that there is a cycle of length* k*.*

(ii) Find a τ_{graph}*-formula with two free variables which says that there is a path of length* k *between two elements* $x, y \in V$*.*

(iii) Find a τ_{graph}*-formula with one free variable which says that a vertex* $x \in V$ *has out–degree (in–degree)* k*.*

Let $\mathcal{G} =< V, E >$ be a linear order. Let $a, b \in V$. We say that a *is smaller (bigger) than* b if $\langle a, b\rangle \in E$ ($\langle b, a\rangle \in E$). a is a *successor (predecessor) of* b if a is the smallest (biggest) element bigger (smaller) than b. An element a is a *first (last)* element if there are no elements in V which are smaller (bigger) than a. $\mathcal{G}$ is *discrete* if every element of V is either a first element, last element or both a successor and a predecessor. $\mathcal{G}$ is *dense* if between every two elements of V such that $\langle a, b\rangle \in E$ there is $c \in V$ with $\langle a, c\rangle \in E$ and $\langle c, b\rangle \in E$.

4.5.9: Exercise
Write τ_{graph}*–formulas which define the above concepts.*

4.5.5 Words and Sets of Words as Structures

Let $\mathcal{WFF}$ the structure of well formed propositional formulas as defined in section 4.2.

4.5.10: Exercise

(i) For every of the axioms $A1, A2, A3$ of Propositional Logic find a formula ϕ_{A1} (ϕ_{A2}, ϕ_{A3}) with two (three, one) free variables which defines the instances of this axiom over $\mathcal{WFF}$.

(ii) Use the above to find sentences which say that the axioms are sound.

(iii) Find, analogously, a sentence ϕ_{MP} which expresses that Modus Ponens is sound.

4.5.6 Data Structures of Computer Science

Stacks and Queues differ only in as much as what one puts in last into a stack gets out first (LIFO), whereas in queues what puts in first gets out first as well (FIFO).

4.5.11: Exercise

(i) Find a τ_{stack}–formula ϕ_{atom} with one free variable which defines the set A in $\mathcal{ST}(A)$.

(ii) Find a τ_{queue}–formula ψ_{atom} with one free variable which defines the set A in $\mathcal{QU}(A)$.

(iii) Find formulas which define the set of stacks (queues) of depth k, for every $k \in \mathbf{N}$.

(iv) Define a similar formula for lists which defines the set of atoms.

Let ϕ_{LIFO} be the formula

$$\phi_{LIFO} = \forall v_1 \forall v_2 \forall v_3((\phi_{atom}(v_2) \wedge R_{push}(v_1, v_2, v_3)) \rightarrow (v_1 \approx F_{pop}(v_3)))$$

and let ϕ_{FIFO} be the formula

$$\forall v_1 \forall v_2 \forall v_3 \forall v_4(\phi_{FIFO.1} \wedge \phi_{FIFO.2})$$

and $\phi_{FIFO.1} =$

$$((\phi_{atom}(v_2) \wedge R_{put}(c_{new}, v_2, v_3)) \rightarrow (c_{new} \approx F_{get}(v_3)))$$

$\phi_{FIFO.2} =$

$$((\phi_{atom}(v_2) \wedge \neg(v_1 \approx c_{new}) \wedge R_{put}(v_1, v_2, v_3) \wedge R_{put}(F_{get}(v_1), v_2, v_4)) \rightarrow (v_4 \approx F_{get}(v_3))).$$

Note that we have ommitted some parentheses for the sake of readibility.

4.5.12: Exercise
Discuss to what extent the formulas ϕ_{LIFO} and ϕ_{FIFO} are related to the LIFO (FIFO) property of stacks (queues).

Let ψ_1, ψ_2, ψ_3 be the following τ_{list}-formulas

$$\psi_1 = \exists v_3(R_{cons}(v_1, v_2, v_3) \wedge (F_{tail}(v_3) \approx v_2)),$$

$$\psi_2 = \exists v_3(R_{cons}(v_1, v_2, v_3) \wedge (F_{head}(v_3) \approx v_1)),$$

$$\psi_3 = \forall v_1 \forall v_2(\psi_1 \wedge \psi_2).$$

4.5.13: Exercise
Discuss the logical lieus defined by ψ_1, ψ_2 and ψ_3 in the structure $\mathcal{LI}(A)$ for various sets A.

4.5.7 $\in$–Structures

4.5.14: Exercise
Formulate the laws and principles of sets from sections 2.1–2.4 as $\tau_\in$–formulas.

4.5.8 Substructures

Recall the definition of universal and existential sentences from definition 4.4.13 and prove the following

4.5.15: Proposition
Let $\mathcal{A}$ and $\mathcal{B}$ be τ–structures and let $\mathcal{A} \subseteq \mathcal{B}$. Let ϕ be a τ–sentence.

(i) If ϕ is universal and $\mathcal{B} \models \phi$ then $\mathcal{A} \models \phi$

(ii) If ϕ is existential and $\mathcal{A} \models \phi$ then $\mathcal{B} \models \phi$

4.5.16: Exercise
Let ϕ_{dense} be the τ_{graph}-sentence which says that a linear order is dense. Show that ϕ_{dense} is neither equivalent to some universal nor to some existential formula.

4.5.17: Exercise
Let $\tau_{emptyset}$ be the empty vocabulary and let $\mathcal{A}, \mathcal{B} \in Str(\tau_{emptyset}$. Show the following:

(i) If $\mathcal{A}$ is finite, then $\mathcal{A} \equiv \mathcal{B}$ iff $\mathcal{A}(\mathbf{Var})$ and $\mathcal{B}(\mathbf{Var})$ have the same number of elements.

(ii) If $\mathcal{A}$ is finite, then $\mathcal{A} \prec \mathcal{B}$ iff $\mathcal{A}(\mathbf{Var}) = \mathcal{B}(\mathbf{Var})$.

(iii) *Let $\mathcal{A}$ be infinite and $\phi \in FOL(\tau_{emptyset})$. Show by induction on ϕ that for every ϕ there is a qunatifier free formula $\psi \in FOL(\tau_{emptyset})$ with the same free variables as ϕ such that for every assignment $z : \mathbf{Var} \to \mathcal{A}(\mathbf{Var})$ we have that $\mathcal{A}, z \models \phi$ iff $\mathcal{A}, z \models \psi$.*

(iv) *If $\mathcal{A}$ is infinite, then $\mathcal{A} \equiv \mathcal{B}$ iff $\mathcal{B}$ is infinite.*

(v) *If $\mathcal{A}$ is infinite, then $\mathcal{A} \prec \mathcal{B}$ iff $\mathcal{A}(\mathbf{Var}) \subseteq \mathcal{B}(\mathbf{Var})$.*

Hint. For (iv) and (v) use (iii). ∎

4.6 Hilbert–style Deduction for First Order Logic

In this section we present the first deduction method for First Order Logic: Hilbert–style deduction (deduction sequences). In section 4.8 we shall see another deduction method, Resolution with Unification. Hilbert–style deduction models to some extent human reasoning as practized by formally trained mathematicians. Resolution is more machine friendly and is extremely popular in Automated Theorem Proving and Logic Programming.

4.6.1 Hilbert-style Axioms and Inference Rules

In this section we want to characterize the notion of logical consequence of First Order Logic syntactically. For simplicity we define our deduction sequences only for formulas built with the connectives $\to$, the logical constant $\mathbf{F}$ and the quantifier $\forall$. The remaining connectives and quantifier are used as abbreviations (macros) according to the following list:

(i) $\neg\phi$ stands for $\phi \to \mathbf{F}$;

(ii) $(\phi \vee \psi)$ stands for $((\phi \to \mathbf{F}) \to \psi)$;

(iii) $(\phi \wedge \psi)$ stands for $((\phi \to (\psi \to \mathbf{F})) \to \mathbf{F})$;

(iv) $\exists v_i \phi$ stands for $\neg\forall v_i \neg\phi$;

4.6.1: Definition (The axioms)

(i) *Let $B(p_1, \ldots, p_n)$ be a propositional formula in $\mathbf{WFF}_{\to,\mathbf{F}}$ which is a propositional tautology. Let $\phi_1, \ldots, \phi_n$ be τ-formulas. Let ϕ be the τ-formula $B(\phi_1, \ldots, \phi_n)$ obtained from B by replacing each occurrence of the propositional variable p_i by ϕ_i. Then ϕ is an axiom.*

(ii) *If ϕ and ψ are τ-formulas, v_i be a variable which does not occur free in ϕ. Then*
$(\forall v_i(\phi \to \psi) \to (\phi \to \forall v_i \psi))$ is an axiom.

(iii) If ϕ is a τ-formula, $s : \mathbf{Var} \mapsto Term(\tau)$ be a function with $s(v_i) = t$ and $s(v_j) = v_j$ for every $j \neq i$, then $(\forall v_i \phi(v_i) \rightarrow subst(\phi, s))$ is an axiom, provided that no variable v_j $(j \neq i)$ that occurs in t has a bounded occurence in ϕ.

4.6.2: Definition (Proof Sequences)
Let Σ be a set of τ-formulas and $\phi_1, \ldots, \phi_n$ be τ-formulas. We say that $\phi_1, \ldots, \phi_n$ is a proof sequence over Σ *if for each $i \leq n$ one of the following holds:*

(i) $\phi_i \in \Sigma$;

(ii) ϕ_i is an axiom;

(iii) (Modus ponens) There are $k, l < i$ such that $\phi_l = (\phi_k \rightarrow \phi_i)$;

(iv) (Generalization) There is $k < i$ such that $\phi_k = \psi(v_l)$ and $\phi_i = \forall v_m subst(\psi(v_l), s)$, provided that

(iv.a) v_l not free in any of the formulas of Σ,

(iv.b) $s : \mathbf{Var} \mapsto \mathbf{Var}$ is a function such that $s(v_l) = v_m$ and $s(v_h) = v_h$ for every $h \neq l$ and v_m does not occur at all in ψ.

We write $\Sigma \vdash \phi$ if there is a proof sequence $\phi_1, \ldots, \phi_n$ over some $\Sigma_0 \subseteq \Sigma$ such that $\phi_n = \phi$.

4.6.3: Definition (Deducible formulas)

Let Σ be a set of τ-formulas. We define the set $Ded(\Sigma)$ by $\phi \in Ded(\Sigma)$ iff $\Sigma \vdash \phi$.

4.6.4: Proposition (Soundness of Proof Sequences)

If $\phi \in Ded(\Sigma)$ then $\Sigma \models \phi$.

Proof:. For the case of the axioms obtained from tautologies of Propositional Logic and for Modus Ponens the proof is the same as for Propositional Logic. Axiom (ii) follows from proposition 4.4.5 on moving quantifiers and axiom (iii) from proposition 4.4.7 on substituting bound variables. We are left with the generalization rule (exercise). ∎

4.6.5: Notation
Let $\phi(v_i)$ be a τ–formula. We shall freely write $\phi(v_j)$ for $subst(\phi(v_i), s)$ with $s(v_i) = v_j$ when ever it should be clear from the context what is meant.

4.6.6: Exercise

(i) Show that $\phi(v_i) \not\vdash \forall v_i \phi(v_i)$.

(ii) Let ϕ *be a* τ*-formula. Pick your favourite* ϕ *and write it down explicitely. Find a proof sequence for* $(\forall v_i \phi(v_i) \to \forall v_j \phi(v_j))$ *for your specifically chosen* ϕ *and* v_j *not occurring in* ϕ.

4.6.7: Examples (Exercise)
Prove the following statements:

(i) $Ded(\emptyset)$ *is a subset of the first order tautologies.*

(ii) $Ded(\{\mathbf{F}\}) = \mathbf{FOL}(\tau)$ *for every vocabulary* τ.

(iii) Let Σ *be infinite and* $\phi \in Ded(\Sigma)$. *Then there is a finite subset* $\Sigma_0 \subseteq \Sigma$ *such that* $\phi \in Ded(\Sigma_0)$.

4.6.8: Definition
We say that a set of τ*-formulas* Σ *is* inconsistent *if* $\Sigma \vdash \mathbf{F}$. *If* Σ *is not inconsistent, we say* Σ *is* consistent.

4.6.9: Remark
Note, by the soundness of proof sequences, that if Σ *is inconsistent, then* Σ *is not satisfiable.*

4.6.2 Manipulations of Proof Sequences

The following are useful properties for the manipulation of proof sequences.

4.6.10: Proposition–Exercise
Let $\Sigma_0 \subseteq \Sigma$ *be two sets of* τ*-formulas and* ϕ *be a* τ*-formula.*

(i) If $\Sigma_0 \vdash \phi$ *then* $\Sigma \vdash \phi$;

(ii) If $\phi_1, \phi_2, \ldots, \phi_n$ *is a proof sequence over* Σ *then for each* $i \leq n$ *we have that* $\Sigma \vdash \phi_i$.

(iii) If $\Sigma \vdash \phi$ *and* $\Sigma \vdash (\phi \to \psi)$ *then* $\Sigma \vdash \psi$.

(iv) If $\Sigma \vdash (\phi \to (\theta \to \psi))$ *and* $\Sigma \vdash (\phi \to \theta)$ *then* $\Sigma \vdash (\phi \to \psi)$.

4.6.11: Proposition (Deduction Theorem)
Let Σ *be a set of* τ*-formulas and* ϕ, ψ *be two* τ*-formulas.*
$\Sigma \vdash (\phi \to \psi)$ *iff* $\Sigma \cup \{\phi\} \vdash \psi$.

Proof:. (i) Assume $\Sigma \vdash (\phi \to \psi)$. We have to prove that $\Sigma \cup \{\phi\} \vdash \psi$. By proposition 4.6.10 (i) above we have $\Sigma \cup \{\phi\} \vdash (\phi \to \psi)$ and, using modus ponens, $\Sigma \cup \{\phi\} \vdash \psi$.

(ii) Assume $\Sigma \cup \{\phi\} \vdash \psi$.
There are two cases:
1. $\Sigma \vdash \psi$ without using ϕ. As $(\psi \to (\phi \to \psi))$ is a tautology, we have, using Modus Ponens, $\Sigma \vdash (\phi \to \psi)$.
2. Otherwise, let $\Sigma_0 \subseteq \Sigma$ such that there is a proof sequence $\psi_1 \dots \psi_n$ over $\Sigma_0 \cup \{\phi\}$ for $\psi_n = \psi$. We have to show that there is a proof sequence over Σ_0 for $(\phi \to \psi)$. This would imply $\Sigma \vdash (\phi \to \psi)$. We proceede by induction on n.
Basis: $n = 1$. $\Sigma \not\vdash \psi$ so the only possibility left is $\psi = \phi$. In this case $(\phi \to \psi)$ is a tautology and we conclude $\Sigma_0 \vdash (\phi \to \psi)$.
Closure: If the last step in the proof sequence is justified by an axiom instance or an hypothesis from $\Sigma_0 \cup \{\phi\}$ we proceed as in the first case or the basis.
If the last step is an application of Modus ponens then there are $1 \leq k, l \leq n$ so that

$$\psi_l = (\psi_k \to \psi).$$

By induction hypothesis

$$\Sigma_0 \vdash (\phi \to \psi_k)$$

and

$$\Sigma_0 \vdash (\phi \to (\psi_k \to \psi)).$$

As in the proof of the deduction theorem for Propositional Logic we conclude

$$\Sigma_0 \vdash (\phi \to \psi).$$

If the last step is an application of the generalization rule with $\psi_k(v_i), 1 \leq k \leq n$ and $\psi = \psi_{n+1} = \forall v_j \psi_k(v_j)$, we have by induction hypothesis

$$\Sigma_0 \vdash (\phi \to \psi_k(v_i)).$$

Also v_i is not free in ϕ or any formula of Σ_0 (otherwise the application of the rule would not have been legal). This allows us to derive

$$\forall v_i(\phi \to \psi_k(v_i))$$

by generalization and then

$$\Sigma_0 \vdash (\phi \to \forall v_i \psi_k(v_i))$$

using the axiom instance

$$\forall v_i(\phi \to \psi_k(v_j)) \to (\phi \to \forall v_i \psi_k(v_i))$$

and Modus ponens. Now, remembering the tautology

$$(\forall v_i \psi_k(v_i) \to \forall v_j \psi_k(v_j))$$

we conclude

$$\Sigma_0 \vdash (\phi \to \forall v_j \psi_k(v_j)).$$

∎

4.6.12: Proposition–Exercise (Dychotomy Theorem)
Let Σ be a set of τ-formulas and ϕ, ψ be two τ-formulas.
If both $\Sigma \cup \{\phi\} \vdash \psi$ and $\Sigma \cup \{(\phi \to \mathbf{F})\} \vdash \phi$ then $\Sigma \vdash \psi$.

4.6.13: Hint
Use the Deduction Theorem.

Proof sequences capture the essence of proofs and can be used for similar formulas in the following sense:

4.6.14: Exercise
Let $\Sigma \subseteq FOL(\tau)$, $\phi \in FOL(\tau)$ and $s : \mathbf{Var} \to Term(\tau)$ be a substitution. Prove that, if $\Sigma \vdash \phi$, then

$$\{subst(\psi, s) : \psi \in \Sigma\} \vdash subst(\phi, s).$$

4.6.3 Completeness and Compactness

The following shows that the method of proof sequences is sufficiently powerful to obtain all tautologies, or, more generally, all logical consequences of a given set of formulas.

4.6.15: Theorem (Completeness theorem for Deductions)
Let Σ be a set of τ-formulas and ϕ be a τ-formula.
If $\Sigma \models \phi$ then $\Sigma \vdash \phi$.

The proof will be given in subsection 4.6.4.

4.6.16: Corollary (Compactness Theorem)
Let Σ be an infinite set of τ-formulas. Σ is satisfiable iff every finite subset $\Sigma_0 \subseteq \Sigma$ is satisfiable.

Proof:. By the completeness theorem above Σ is satisfiable iff Σ is consistent. Clearly, if Σ is consistent, so is every finite subset $\Sigma_0 \subseteq \Sigma$. Conversely, assume Σ is inconsistent. By definition and example 4.6.7 (iii) there is a finite subset $\Sigma_1 \subseteq \Sigma$ such that $\mathbf{F} \in Ded(\Sigma_1)$ and therefore Σ_1 is a finite inconsistent subset. ∎

As a first application of compactness we have

4.6.17: Theorem (Finitely Definable Classes)
Let $K \subseteq Str(\tau)$ and $\bar{K} = Str(\tau) \setminus K$. The following are equivalent:

(i) K *is finitely definable.*

(ii) $\bar{K}$ *is finitely definable.*

(iii) Both K *and* $\bar{K}$ *are definable.*

Proof:. Exactly as the corresponding theorem for Propositional Logic. ∎

More applications of the compactness theorem may be found in the next visit to the museum, section 4.7.

4.6.4 Proof of the Completeness Theorem

The proof of the completeness theorem 4.6.15 comes in several stages. We first observe that it suffices to prove that Σ is satisfiable iff Σ is consistent. In other words

4.6.18: Lemma
Assume that for every set of τ*-formulas* Σ*,* Σ *is satisfiable iff* Σ *is consistent. Then for every set of* τ*-formulas* Σ *and every* τ*–formula* ϕ $\Sigma \models \phi$ *iff* $\Sigma \vdash \phi$*.*

Proof. Use the deduction theorem. ∎

Next we need a set theoretic lemma:

4.6.19: Lemma
Let $\tau = \bigcup_{i \in \mathbf{N}} \tau_i$ *with* $\tau i \subseteq \tau_j$ *for* $i \leq j$*. Let* $\Sigma_i, i \in \mathbf{N}$ *be a family of* τ_i*-formulas such that for each* $i \in \mathbf{N}$ Σ_i *is consistent and* $\Sigma_i \subseteq \Sigma_{i+1}$*. Then* $\Sigma = \bigcup \Sigma_i$ *is consistent.*

Proof. Σ is consistent. For, otherwise, there is a finite deduction sequence showing inconsistence over some finite subset $X \subseteq \Sigma$. But then there is an $i \in \mathbf{N}$ with Σ_i inconsistent. ∎

Next we introduce the notion of maximally consistent sets of formulas.

4.6.20: Definition
Let Σ *be a set of* τ*–formulas.* Σ *is* maximally consistent *if* Σ *is consistent and for every* τ*–sentence* ϕ $\phi \in \Sigma$ *or* $\neg\phi \in \Sigma$*.*

4.6.21: Lemma
Let Σ *be a consistent set of* τ*–sentences. Then there is a maximally consistent set of* τ*–sentences* Σ^* *with* $\Sigma \subseteq \Sigma^*$*.*

Proof. Let $\{\phi_i\}$ be an enumeration of all τ–sentences.
Let $\Sigma_0 = \Sigma$.
Let $\Sigma_{n+1} = \Sigma_n \cup \{\phi_n\}$ if it is consistent, and Let $\Sigma_{n+1} = \Sigma_n \cup \{\neg\phi_n\}$ otherwise.
Use the Deduction theorem to show that for each $n \in \mathbf{N}$ Σ_n is consistent.
Now let Σ^* be the union of all the Σ_n. Σ^* is consistent by lemma 4.6.19.
To show that Σ^* is maximally consistent, let ψ be a τ–sentence. $\psi = \phi_k$ for some $k \in \mathbf{N}$. But then either ψ or $\neg\psi$ is in Σ_{k+1} and therefore in Σ^*. ∎

4.6.22: Examples

(i) Let $\mathcal{A}$ be a τ-structure and $Th(\mathcal{A}) = \{\phi \in \mathbf{FOL}(\tau) : \mathcal{A} \models \phi\}$. Then $Th(\mathcal{A})$ is maximally consistent. To see this we note first that $Th(\mathcal{A})$ is satisfiable (by $\mathcal{A}$), and therefore consistent, by the soundness of the deduction rules. Now let ϕ be a τ-formula without free variables. Clearly, either $\mathcal{A} \models \phi$ or $\mathcal{A} \models \neg\phi$. Therefore either $\phi \in Th(\mathcal{A})$ or $\neg\phi \in Th(\mathcal{A})$, which shows that $Th(\mathcal{A})$ is maximally consistent.

(ii) Let Σ be a consistent set of τ–formulas such that for every two τ-structures $\mathcal{A}$, $\mathcal{B}$ with $\mathcal{A} \models \Sigma$ and $\mathcal{B} \models \Sigma$ we have that $\mathcal{A} \equiv \mathcal{B}$, i.e for every τ-sentence ϕ we have that $\mathcal{A} \models \phi$ iff $\mathcal{B} \models \phi$. Then $Ded(\Sigma)$ is maximally consistent. (Exercise, similar to the above).

4.6.23: Definition
A constant *τ–term is a τ–term without free variables.*

4.6.24: Definition
Let Σ be a set of τ–sentences. We say that Σ has enough witnesses if for every τ–formula $\phi(v_i)$ with v_i the only free variable in ϕ there is a constant τ–term t such that if $\exists v_i\phi(v_i) \in \Sigma$ then $\phi(t) \in \Sigma$. Here $\phi(t)$ is an abbreviation for the result of substituting t for v_i in ϕ.

4.6.25: Lemma
Let Σ be a countable consistent set of τ–sentences. Then there is a countable vocabulary τ^+ and a countable consistent set of τ^+-sentences Σ^+ which has enough witnesses.

4.6.26: Lemma
Let Σ be a countable consistent set of τ–sentences. Then there is a countable vocabulary $\tau^\#$ and a a countable set of $\tau^\#$-sentences $\Sigma^\#$ which has enough witnesses and is maximally consistent.

Proof:. Use lemma 4.6.21 and lemma 4.6.25. ∎

4.6.27: Lemma
Let Σ be a countable set of τ–sentences not containing the equality symbol. If Σ has enough witnesses and is maximally consistent, then Σ is satisfiable.

The proof of the completeness theorem 4.6.15 now proceeds as follows.

- Assume Σ is consistent.
- Using, lemma 4.6.26 we can find Σ^* with $\Sigma \subseteq \Sigma^*$, such that Σ^* is maqximally consistent and has enough witnesses.
- Using lemma 4.6.27 we conclude there is a τ^*–structure $\mathcal{A}$ with $\mathcal{A} \models \Sigma^*$.
- Regarding $\mathcal{A}$ as a τ–structure we conclude that $\mathcal{A} \models \Sigma$.

4.6.5 The Case with Equality

We have shown the completeness theorem only for formulas with equality.

In this section we shall expand the deduction rules and the proof of the completeness theorem to formulas with equality.

4.6.28: Remark
Not done in the course. To be completed.

4.7 Visit to the Museum: Non–Definability

4.7.1 Finite Structures

4.7.1: Exercise

(i) Write down sentences ϕ_n over $\tau = \emptyset$ which say that there are at least n different elemnts.

(ii) Let $\Sigma_{inf} = \{\phi_n : n \in \mathbf{N}\}$. Show that Σ_{inf} is satisfiable.

(iii) Let $\Sigma \subseteq SENT(\tau)$ for some not empty τ. Assume that $MOD(\Sigma)$ contains arbitrarily large finite models. Show that $\Sigma \cup \Sigma_{inf}$ is finitely satisfiable.

4.7.2: Exercise
Use the previous exercise to show that

(i) The class of finite graphs is not definable.

(ii) The class of finite orders is not definable.

(iii) The class of finite fields is not definable.

Show the following

4.7.3: Proposition
Let $K \subseteq Str(\tau)$ be a class of finite structures. Show that the following are equivalent:

(i) K is definable.

(ii) K is finitely definable.

(iii) There is an $n \in \mathbf{N}$ such that every $\mathcal{A} \in K$ has less than n elements.

4.7.2 The Real Numbers

Recall that τ_{arith} is the vocabulary $\{F_+, F_*, R_<, c_0, c_1\}$ consisting of two binary function symbols, one binary relation symbol and two constant symbols and $\mathcal{R}$ as a τ_{arith}–structure is the Arithmetic Structure of the Real Numbers.

We write $\mathbf{n}$ for $\underbrace{1 + \ldots + 1}_{n}$. $\mathbf{n}$ represents the natural number n in $\mathcal{R}$. We write $\mathbf{t_n}$ for the term $\underbrace{F_+(c_1 F_+(\ldots + F_+(c_1, c_1)\ldots)}_{n-1}$. $\mathbf{t_n}$ is a term whose interpretation in $\mathcal{R}$ is the natural number n in $\mathcal{R}$. Now $\mathcal{R}$ satisfies the *Archimedean Property*, i.e. for every x, y with $0 < x$ and $0 < y$ there is an n such that $y < \mathbf{n} * x$.

4.7.4: Proposition
There is no τ_{arith}-formula ψ which expresses the Archimedean property. In other words, for every set of τ_{arith}-formulas Σ such that $\mathcal{R} \models \Sigma$ there is a τ_{arith}-structure $\mathcal{B}$ with $\mathcal{B} \models \Sigma$ which does not satisfy the Archimedean Property.

Proof. We use the compactness theorem. Let c_2 be a new constant symbol. Let $\Sigma = \{\mathbf{t_n} < c_2 : n \in \mathbf{N}\}$. Let $Th(\mathcal{R}) = \{\phi \in \mathbf{SEN}(\tau_{arith}) : \mathcal{R} \models \phi\}$.
Claim: $T = Th(\mathcal{R}) \cup \Sigma$ is satisfiable.
Let $\Sigma_m = \{\mathbf{t_n} < c_2 : n < m\}$ and $T_m = Th(\mathcal{R}) \cup \Sigma_m$. Let X be a finite subset of T. Then there is an m such that $X \subseteq T_m$. We define the $\tau_{arith} \cup \{c_2\}$–structure $\mathcal{R}_m$ as follows: All symbols of τ_{arith} are interpreted as for $\mathcal{R}$. The constant c_2 is interpreted as the interpretation of the term $\mathbf{t_m}$. Clearly, $\mathcal{R}_m \models T_m$, so X is satisfiable. Using the compactness theorem, we conclude that T is satisfiable. So let $\mathcal{B} \models T$. It is now easy to see, that $\mathcal{B}$ does not satisfy the Archimedian Property. ∎

4.7.3 The Natural Numbers

Recall that the τ_{arith}–structure $\mathcal{N}$ is the Arithmetic Structure of the Natural Numbers. The natural numbers satisfy the *Induction Principle*:

Induction Principle For every $X \subseteq \mathbf{N}$ such that $0 \in X$ and whenever $n \in X$ the $n+1 \in X$, then $X = \mathbf{N}$.

We now show that the Induction Principle is not expressible in first order logic. More precisely

4.7.5: Proposition
There is no τ_{peano}-formula ψ which expresses the Induction Principle. In other words, for every set of τ_{peano}-formulas Σ such that $\mathcal{N}_{peano} \models \Sigma$ there is a τ_{peano}-structure $\mathcal{B}$ with $\mathcal{B} \models \Sigma$ which does not satisfy the Induction Principle.

Proof. We use the compactness theorem. Let c_2 be a new constant symbol. Let $\Sigma = \{\mathbf{t_n} < c_2 : n \in \mathbf{N}\}$. Let $Th(\mathcal{N}_{peano}) = \{\phi \in \mathbf{SEN}(\tau_{peano}) : \mathcal{N}_{peano} \models \phi\}$.
Claim: $T = Th(\mathcal{N}_{peano}) \cup \Sigma$ is satisfiable.
Let $\Sigma_m = \{\mathbf{t_n} < c_2 : n < m\}$ and $T_m = Th(\mathcal{N}_{peano}) \cup \Sigma_m$. Let X be a finite subset of T. Then there is an m such that $X \subseteq T_m$. We define the $\tau_{peano} \cup \{c_2\}$–structure $\mathcal{N}_m$ as follows: All symbols of τ_{peano} are interpreted as for $\mathcal{N}_{peano}$. The constant c_2 is interpreted as the interpretation of the term $\mathbf{t_m}$. Clearly, $\mathcal{N}_m \models T_m$, so X is satisfiable. Using the compactness theorem, we conclude that T is satisfiable. So let $\mathcal{B} \models T$. It is now easy to see, that $\mathcal{B}$ does not satisfy they Induction Principle. Let $X \subseteq \mathcal{B}(\mathrm{Var})$ be the set $\{\mathbf{n} : n \in \mathbf{N}\}$. Clearly, X satisfies the hypothesis of the Induction Principle, but $\mathcal{B}(c_2) \notin X$. ∎

4.7.6: Remark
The same proof also works for $\mathcal{N}$ as a τ_{arith}-structure.

The best we can do in first order logic in describing the Induction Principle consists of writing down each instance for it for which X is definable as a formula. This is what we have done in section 4.5.

4.8 Unification and Resolution

In this section we discuss a second deduction method, Unification and Resolution, which extends the Reslotion Method of Propositional Logic.

4.8.1: Remark
Not treated in the course, to be completed