Publishers' page

Publishers' page

Publishers' page

Publishers' page

Contents

Acknowledgments

Several lecturers have been using preliminary versions of this text and contributed valuable comments and corrections to its present, final form. I want to specifically thank Tore Langholm, Valentinas Kriaučiukas and Uwe Wolter.

The first part, on the history of logic, is to a high degree a compilation of various internet resources. I gratefully acknowledge the use of much detailed information from Encyclopedia Britannica (`www.britannica.com`), Wikipedia (`en.wikipedia.org`), Leibnitiana (`www.gwleibniz.com`), Stanford Encyclopedia (`plato.stanford.edu`), and apologize the owners of other sources which I might have inadverently left out. The arrangement, development and conclusions of this part, however, reflect only the author's views and should not be attributed to other sources.

The History of Logic

Chapter 1
Sets, Functions, Relations

- Sets and Functions
  - Set building operations
  - Some equational laws
- Relations and Sets with Structures
  - Properties of relations
  - Ordering relations
- Infinities
  - Countability vs. uncountability

1: Sets and Functions

◇——— a Background Story ———◇

A *set* is an arbitrary collection of arbitrary objects, called its *members*. One should take these two occurrences of "arbitrary" seriously. Firstly, sets may be finite, e.g., the set C of cars on the parking lot outside the building, or infinite, e.g. the set N of numbers greater than 5.

Secondly, any objects can be members of sets. We can talk about sets of cars, blood-cells, numbers, Roman emperors, etc. We can also talk about the set X whose elements are: my car, your mother and number 6. (Not that such a set necessarily is useful for any purpose, but it is possible to collect these various elements into one set.) In particular sets themselves can be members of other sets. I can, for instance, form the set whose elements are: my favorite pen, my four best friends and *the set* N. This set will have 6 elements, even though the set N itself is infinite.

A set with only one element is called a *singleton*, e.g., the set containing only planet Earth. There is one special and very important set – the *empty* set – which has no members. If it seems startling, you may think of the set of all square circles or all numbers x such that $x < x$. This set is mainly a mathematical convenience – defining a set by describing the properties of its members in an involved way, we may not know from the very begining what its members are. Eventually, we may find that no such objects exist, that is, that we defined an empty set. It also makes many formulations simpler since, without the assumption of its existence, one would often had to take special precautions for the case a set happened to contain no elements.

It may be legitimate to speak about a set even if we do not know exactly its members. The set of people born in 1964 may be hard to determine exactly but it is a well defined object because, at least in principle, we can determine membership of any object in this set. Similarly, we will say that the set R of red objects is well defined even if we certainly do not know all its members. But confronted with a new object, we can determine if it belongs to R or not (assuming, that we do not dispute the meaning of the word "red".)

There are four basic means of specifying a set.

(1) If a set is finite and small, we may list all its elements, e.g., $S = \{1, 2, 3, 4\}$ is a set with four elements.
(2) A set can be specified by determining a property which makes objects qualify as its elements. The set R of red objects is specified in this way. The set S can be described as 'the set of natural numbers greater than 0 and less than 5'.
(3) A set may be obtained from other sets. For instance, given the set S and the set $S' = \{3, 4, 5, 6\}$ we can form a new set $S'' = \{3, 4\}$ which is the *intersection* of S and S'. Given the sets of odd $\{1, 3, 5, 7, 9...\}$ and even numbers $\{0, 2, 4, 6, 8...\}$ we can form a new set $\mathbb{N}$ by taking their *union*.
(4) Finally, a set can be determined by describing the rules by which its elements may be generated. For instance, the set $\mathbb{N}$ of natural numbers $\{0, 1, 2, 3, 4, ...\}$ can be described as follows: 0 belongs to $\mathbb{N}$ and if n belongs to $\mathbb{N}$, then also $n + 1$ belongs to $\mathbb{N}$ and, finally, nothing else belongs to $\mathbb{N}$.

In this chapter we will use mainly the first three ways of describing sets. In particular, we will use various set building operations as in point 3. In the later chapters, we will constantly encouter sets described by the last method. One important point is that the properties of a set are entirely independent from the way the set is described. Whether we just say 'the set of natural numbers' or the set $\mathbb{N}$ as defined in point 2. or 4., we get the same set. Another thing is that *studying and proving* properties of a set may be easier when the set is described in one way rather than another.

◇————————————————————◇

Definition 1.1 Given some sets S and T we write:

$x \in S$	-	x is a member (element) of S	
$S \subseteq T$	-	S is a subset of T	for all x : if $x \in S$ then $x \in T$
$S \subset T$	-	$S \subseteq T$ and $S \neq T$	for all x : if $x \in S$ then $x \in T$ and for some x : $x \in T$ and $x \notin S$

Set building operations :

$\varnothing$	-	empty set	for any x : $x \notin \varnothing$
$S \cup T$	-	union of S and T	$x \in S \cup T$ iff $x \in S$ or $x \in T$
$S \cap T$	-	intersection of S and T	$x \in S \cap T$ iff $x \in S$ and $x \in T$
$S \setminus T$	-	difference of S and T	$x \in S \setminus T$ iff $x \in S$ and $x \notin T$
$\overline{S}$	-	complement of S; given a universe U of all elements	$\overline{S} = U \setminus S$
$S \times T$	-	Cartesian product of S and T	$x \in S \times T$ iff $x = \langle s, t\rangle$ and $s \in S$ and $t \in T$
$\wp(S)$	-	the power set of S	$x \in \wp(S)$ iff $x \subseteq S$

Also, $\{x \in S : \mathsf{Prop}(x)\}$ denotes the set of those $x \in S$ which have the specified property Prop.

Remark.
Sets may be members of other sets. For instance $\{\varnothing\}$ is the set with one element – which is the empty set $\varnothing$. In fact, $\{\varnothing\} = \wp(\varnothing)$. It is different from the set $\varnothing$ which has no elements. $\{\{a, b\}, a\}$ is a set with two elements: a and the set $\{a, b\}$. Also $\{a, \{a\}\}$ has two different elements: a and $\{a\}$. In particular, the power set contains only sets as elements: $\wp(\{a, \{a, b\}\}) = \{\varnothing, \{a\}, \{\{a, b\}\}, \{a, \{a, b\}\}\}$.

In the definition of Cartesian product, we used the notation $\langle s, t\rangle$ to denote an *ordered* pair whose first element is s and second t. In set theory, all possible objects are modelled as sets. An ordered pair $\langle s, t\rangle$ is then represented as the set with two elements – both being sets – $\{\{s\}, \{s, t\}\}$. Why not $\{\{s\}, \{t\}\}$ or, even simpler, $\{s, t\}$? Because elements of a set are not ordered. Thus $\{s, t\}$ and $\{t, s\}$ denote the same set. Also, $\{\{s\}, \{t\}\}$ and $\{\{t\}, \{s\}\}$ denote the same set (but different from the set $\{s, t\}$). In ordered pairs, on the other hand, the order does matter – $\langle s, t\rangle$ and $\langle t, s\rangle$ are different pairs. This ordering is captured by the representation $\{\{s\}, \{s, t\}\}$. We have here a set with two elements $\{A, B\}$ where $A = \{s\}$ and $B = \{s, t\}$. The relationship between these two elements tells us which is the first and which the second: $A \subset B$ identifies the member of A as the first element of the pair, and then the element of $B \setminus A$ as the second one. Thus $\langle s, t\rangle = \{\{s\}, \{s, t\}\} \neq \{\{t\}, \{s, t\}\} = \langle t, s\rangle$.

The set operations $\cup$, $\cap$, and $\setminus$ obey some well known laws:

1. Idempotency
$$A \cup A = A$$
$$A \cap A = A$$

2. Associativity
$$(A \cup B) \cup C = A \cup (B \cup C)$$
$$(A \cap B) \cap C = A \cap (B \cap C)$$

3. Commutativity
$$A \cup B = B \cup A$$
$$A \cap B = B \cap A$$

4. Distributivity
$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$$
$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$$

5. deMorgan
$$\overline{(A \cup B)} = \overline{A} \cap \overline{B}$$
$$\overline{(A \cap B)} = \overline{A} \cup \overline{B}$$

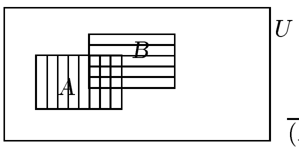
6. Complement
$$A \cap \overline{A} = \varnothing$$
$$A \setminus B = A \cap \overline{B}$$

7. Emptyset
$$\varnothing \cup A = A \qquad\qquad \varnothing \cap A = \varnothing$$

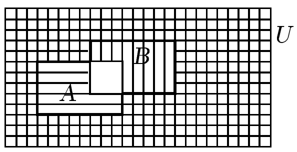
8. Consistency principles

a) $A \subseteq B$ iff $A \cup B = B$ b) $A \subseteq B$ iff $A \cap B = A$

Remark 1.2 **[Venn's diagrams]**
It is very common to represent sets and set relations by means of Venn's diagrams – overlapping figures, typically, circles or rectangles. On the the left in the figure below, we have two sets A and B in some universe U. Their intersection $A \cap B$ is marked as the area belonging to both by both vertical and horisontal lines. If we take A to represent Armenians and B bachelors, the darkest region in the middle represents Armenian bachelors. The region covered by only vertical, but not horisontal, lines is the set difference $A \setminus B$ – Armenians who are not bachelors. The whole region covered by either vertical or horisontal lines represents all those who are either Armenian or are bachelors.



$\overline{(A \cup B)} = \overline{A} \cap \overline{B}$



Now, the white region is the complement of the set $A \cup B$ (in the universe U) – all those who are neither Armenians nor bachelors. The diagram to the right is essentially the same but was constructed in a different way. Here, the region covered with vertical lines is the complement of A – all non-Armenians. The region covered with horisontal lines represents all non-bachelors. The region covered with *both* horisontal and vertical lines is the intersection of these two complements – all those who are neither Armenians nor bachelors. The two diagrams illustrate the first de Morgan law since the white area on the left,

$\overline{(A \cup B)}$, is exactly the same as the area covered with both horisontal and vertical lines on the right, $\overline{A} \cap \overline{B}$.

Venn's diagrams may be handy tool to visualize simple set operations. However, the equalities above can be also seen as a (not yet quite, but almost) formal system allowing one to derive various other set equalities. The rule for performing such derivations is 'substitution of equals for equals', known also from elementary arithemtics. For instance, the fact that, for an arbitrary set $A : A \subseteq A$ amounts to a single application of rule 8.a): $A \subseteq A$ iff $A \cup A = A$, where the last equality holds by 1. A bit longer derivation shows that $(A \cup B) \cup C = (C \cup A) \cup B$:

$$(A \cup B) \cup C \stackrel{3}{=} C \cup (A \cup B) \stackrel{2}{=} (C \cup A) \cup B.$$

In exercises we will encounter more elaborate examples.

In addition to the set building operations from the above defintion, one often encounters also *disjoint union* of sets A and B, written $A \uplus B$ and defined as $A \uplus B = (A \times \{0\}) \cup (B \times \{1\})$. The idea is to use 0, resp. 1, as indices to distinguish the elements originating from A and from B. If $A \cap B = \varnothing$, this would not be necessary, but otherwise the "disjointness" of this union requires that the common elements be duplicated. E.g., for $A = \{a, b, c\}$ and $B = \{b, c, d\}$, we have $A \cup B = \{a, b, c, d\}$ while $A \uplus B = \{\langle a, 0\rangle, \langle b, 0\rangle, \langle c, 0\rangle, \langle b, 1\rangle, \langle c, 1\rangle, \langle d, 1\rangle\}$, which can be thought of as $\{a_0, b_0, c_0, b_1, c_1, d_1\}$.

Definition 1.3 Given two sets S and T, a *function* f from S to T, $f : S \to T$, is a subset of $S \times T$ such that

- whenever $\langle s, t\rangle \in f$ and $\langle s, t'\rangle \in f$, then $t = t'$, and
- for each $s \in S$ there is some $t \in T$ such that $\langle s, t\rangle \in f$.

A subset of $S \times T$ that satisfies the first condition above but not necessarily the second, is called a *partial* function from S to T.

For a function $f : S \to T$, the set S is called the *source* or *domain* of the function, and the set T its *target* or *codomain*.

The second point of this definition means that function is *total* – for each argument (element $s \in S$), the function has some value, i.e., an element $t \in T$ such that $\langle s, t\rangle \in f$. Sometimes this requirement is dropped and one speaks about *partial functions* which may have no value for some arguments but we will be for the most concerned with total functions.

Example 1.4

Let $\mathbb{N}$ denote the set of natural numbers $\{0, 1, 2, 3, ...\}$. The mapping

$f : \mathbb{N} \to \mathbb{N}$ defined by $f(n) = 2n$ is a function. It is the set of all pairs $f = \{\langle n, 2n\rangle : n \in \mathbb{N}\}$. If we let M denote the set of all people, then the set of all pairs $father = \{\langle m, m'\text{s father}\rangle : m \in M\}$ is a function assigning to each person his/her father. A mapping '*children*', assigning to each person his/her children is not a function $M \to M$ for two reasons. For the first, a person may have no children, while saying "function" we mean a total function. For the second, a person may have more than one child. These problems may be overcome if we considered it instead as a function $M \to \wp(M)$ assigning to each person the set (possibly empty) of all his/her children. □

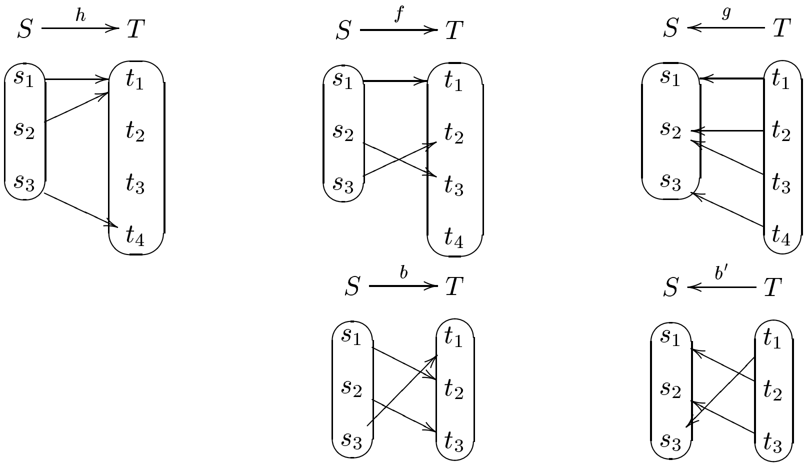
Notice that although intuitively we think of a function as a mapping assigning to each argument some value, the definition states that it is actually a set (a subset of $S \times T$ is a set.) The restrictions put on this set are exactly what makes it possible to think of this set as a mapping. Nevertheless, functions – being sets – can be elements of other sets. We may encounter situations involving sets of functions, e.g. the set T^S of all functions from set S to set T, which is just the set of all subsets of $S \times T$, each satisfying the conditions of the definition 1.3.

Remark 1.5 **[Notation]**
A function f associates with each element $s \in S$ a *unique* element $t \in T$. We write this t as $f(s)$ – the value of f at point s.
When S is finite (and small) we may sometimes write a function as a set $\{\langle s_1, t_1\rangle, \langle s_2, t_2\rangle, ..., \langle s_n, t_n\rangle\}$ or else as $\{s_1 \mapsto t_1, s_2 \mapsto t_2, ..., s_n \mapsto t_n\}$.
If f is given then by $f[s \mapsto p]$ we denote the function f' which is the same as f for all arguments $x \neq s : f'(x) = f(x)$, while $f'(s) = p$.

Definition 1.6 A function $f : S \to T$ is
injective iff whenever $f(s) = f(s')$ then $s = s'$;
surjective iff for all $t \in T$ there exists an $s \in S$ such that $f(s) = t$;
bijective, or a **set-isomorphism**, iff it is both injective and surjective.

Injectivity means that no two distinct elements from the source set are mapped to the same element in the target set; surjectivity that each element in the target is an image of some element from the source.

Example 1.7
The function $father : M \to M$ is injective – everybody has exactly one (biological) father, but it is not surjective – not everybody is a father of somebody. The following drawing gives some examples:



Here h is neither injective nor surjective, f is injective but not surjective, $g : T \to S$ is surjective but not injective. b and b' are both injective and surjective, i.e, bijective. $\square$

Soon we will see the particular importance of bijections. There may exist several different set-isomorphisms between two sets S and T. If there is at least one set-isomorphism, we say that the two sets are isomorphic and write $S \rightleftharpoons T$. The following lemma gives another criterion for a function to be a bijection.

Lemma 1.8 $f : S \to T$ is a set-isomorphism if and only if there is an *inverse* function $f^{-1} : T \to S$, such that for all $s \in S : f^{-1}(f(s)) = s$ and for all $t \in T : f(f^{-1}(t)) = t$.

Proof. We have to show two implications:
only if) If f is iso, we can define f^{-1} simply as the set of pairs $f^{-1} \stackrel{\text{def}}{=} \{\langle t, s\rangle : t = f(s)\}$.
if) If $f(s) = f(s')$ then $s = f^{-1}(f(s)) = f^{-1}(f(s')) = s'$, i.e., f is injective. Then, for any $t \in T$ we have an $s = f^{-1}(t)$ for which $f(s) = f(f^{-1}(t)) = t$, i.e., f is surjective. **QED** (1.8)

In the example 1.7 both b and b' were bijective. In fact, they acted as mutual inverses satisfying the conditions of the above lemma: for each $s \in S; b'(b(s)) = s$ and for each $t \in T : b(b'(t)) = t$.

Definition 1.9 For any set U and $A \subseteq U$, the *characteristic function* of A (relatively to U), denoted f_A, is the function

$$\{\langle x, \mathbf{1}\rangle \mid x \in A\} \ \cup \ \{\langle x, \mathbf{0}\rangle \mid x \in A'\}.$$

Hence $f_A(x) = \mathbf{1}$ iff $x \in A$. Note that f_A is a function from U to $\{\mathbf{1}, \mathbf{0}\}$, where $\{\mathbf{1}, \mathbf{0}\}$ is a(ny) set with exactly two elements.

Let $f\_ : \wp(U) \to 2^U$ denote the function sending each subset A of U to its characteristic function f_A (the notation 2^U stands for the set of all functions from U to a two-element set, e.g., to $\{\mathbf{1}, \mathbf{0}\}$).

It is clear that if $A \neq B$ then $f_A \neq f_B$. The first inequality means that there is an x such that either $x \in A \setminus B$ or $x \in B \setminus A$. In either case we will have that $f_A(x) \neq f_B(x)$. Thus $f\_$ is injective.

On the other hand, every function $f \in 2^U$ is the characteristic function of some subset of U, namely, of the set $A_f = \{x \in U : f(x) = \mathbf{1}\}$. That is, $f\_$ is surjective. Together, these two facts mean that we have a set-isomorphism $f\_ : \wp(U) \rightleftharpoons 2^U$.

2: Relations

Definition 1.10 A *binary relation* R between sets S and T is a subset $R \subseteq S \times T$.
A *binary relation on a set* S is a subset of $S \times S$, and an n-ary relation on S is a subset of $S_1 \times S_2 \times ... \times S_n$, where each $S_i = S$.

Definition 1.10 makes *any* subset of $S \times T$ a relation. The definition 1.3 of function, on the other hand, required this set to satisfy some additional properties. Hence a function is a special case of relation, namely, a relation which relates each element of S with exactly one element of T.

Binary relations are sets of *ordered* pairs, i.e., $\langle s, t\rangle \neq \langle t, s\rangle$ – if $s \in S$ and $t \in T$, then the former belongs to $S \times T$ and the latter to $T \times S$, which are different sets. For sets, there is no ordering and thus $\{s, t\} = \{t, s\}$. It is common to write the fact that s and t stand in a relation R, $\langle s, t\rangle \in R$, as sRt or as $R(s, t)$.

In general, relations may have arbitrary arities, for instance a subset $R \subseteq S \times T \times U$ is a ternary relation, etc. As a particular case, a *unary* relation on S is simply a subset $R \subseteq S$. In the following we are speaking only about binary relations (unless explicitly stated otherwise).

Example 1.11
Functions, so to speak, map elements of one set onto the elements of another

set (possibly the same). Relations *relate* some elements of one set with some elements of another. Recall the problem from example 1.4 with treating *children* as a function $M \to M$. This problem is overcome by treating *children* as a binary relation on the set M of all people. Thus $children(p, c)$ holds if an only if c is a child of p. Explicitly, this relation is $children = \{\langle p, c\rangle : p, c \in M \text{ and } c \text{ is a child of } p\}$. □

Definition 1.12 Given two relations $R \subseteq S \times T$ and $P \subseteq T \times U$, their *composition* is the relation $R ; P \subseteq S \times U$, defined as the set of pairs

$$R ; P = \{\langle s, u\rangle \in S \times U : \text{there is a } t \in T \text{ with } R(s, t) \text{ and } P(t, u)\}.$$

As functions are special cases of relations, the above definition allows us to form the composition $g ; f$ of functions $g : S \to T$ and $f : T \to U$, namely, the function from S to U given by the equation $(g ; f)(s) = f(g(s))$.

Definition 1.13 A relation $R \subseteq S \times S$ is :

connected	iff	for all pairs of distinct $s, t \in S : R(s, t)$ or $R(t, s)$		
reflexive	iff	for all $s \in S : R(s, s)$		
irreflexive	iff	for no $s \in S : R(s, s)$		
transitive	iff	when $R(s_1, s_2)$ and $R(s_2, s_3)$	then	$R(s_1, s_3)$
symmetric	iff	when $R(s, t)$	then	$R(t, s)$
asymmetric	iff	when $R(s, t)$	then	not $R(t, s)$
antisymmetric	iff	when $R(s, t)$ and $R(t, s)$	then	$s = t$
equivalence	iff	it is reflexive, transitive and symmetric.		

Numerous connections hold between these properties: every irreflexive, transitive relation is also asymmetric, and every asymmetric relation is also antisymmetric. Note also that just one relation is both connected, symmetric and reflexive, namely the universal relation $S \times S$ itself.

The most common (and smallest) example of equivalence is the *identity* relation $id_S = \{\langle s, s\rangle : s \in S\}$. The relation $\rightleftharpoons$ – existence of a set-isomorphism – is also an equivalence relation on any set (collection) of sets. An equivalence relation $\sim$ on a set S allows us to *partition* S into disjoint *equivalence classes* $[s] = \{s' \in S : s' \sim s\}$.

Given a relation $R \subseteq S \times S$, we can form its *closure* with respect to one or more of these properties. For example, the *reflexive closure* of R, written $\underline{R}$, is the relation $R \cup id_S$. (It may very well happen that R already is reflexive, $R = \underline{R}$, but then we do have that $\underline{R} = \underline{\underline{R}}$.) The *transitive closure* of R, written R^+, can be thought of as the infinite union

$$R \cup (R ; R) \cup (R ; R ; R) \cup (R ; R ; R ; R) \cup (R ; R ; R ; R ; R) \cup \ldots$$

and can be defined as the least relation R^+ such that $R^+ = R \cup (R^+ ; R)$.

3: Ordering Relations

Of particular importance are the *ordering* relations which we will use extensively in chapter 2 and later. Often we assume an implicit set S and talk only about relation R. However, it is important to realize that when we are talking about a relation R, we are actually considering a *set with structure*, namely a pair $\langle S, R\rangle$.

Definition 1.14 $\langle S, R\rangle$ is a *quasiorder* (or *preorder*), QO, iff R is

transitive : $R(x, y) \wedge R(y, z) \rightarrow R(x, z)$
reflexive : $R(x, x)$

$\langle S, R\rangle$ is a *weak partial order*, wPO, iff R is

a quasiorder : $R(x, y) \wedge R(y, z) \rightarrow R(x, z)$
: $R(x, x)$
antisymmetric : $R(x, y) \wedge R(y, x) \rightarrow x = y$

$\langle S, R\rangle$ is a *strict partial order*, sPO, iff R is

transitive : $R(x, y) \wedge R(y, z) \rightarrow R(x, z)$
irreflexive : $\neg R(x, x)$

A *total order*, TO, is a PO which is connected : $x \neq y \rightarrow R(x, y) \vee R(y, x)$

A QO allows loops, for instance the situations like $R(a_1, a_2)$, $R(a_2, a_3)$, $R(a_3, a_1)$ for distinct a_1, a_2, a_3. PO forbids such situations: by applications of transitivity we have

- $R(a_2, a_1)$ as well as $R(a_1, a_2)$, which for wPO's imply $a_1 = a_2$.
- $R(a_1, a_1)$, which is impossible for sPO's.

Obviously, given a wPO, we can trivially construct its strict version (by making it irreflexive) and vice versa. We will therefore often say "partial order" or PO without specifying which one we have in mind.

Instead of writing $R(x, y)$ for a PO, we often use the infix notation $x \leq y$ for a wPO, and $x < y$ for the corresponding *sPO*. In other words, $x \leq y$ means $x < y$ or $x = y$, and $x < y$ means $x \leq y$ and $x \neq y$.

Example 1.15

Consider the set of all people and their ages.

(1) The relation 'x is older than y' is an sPO on the set of all people : it is transitive, irreflexive (nobody is older than himself) and asymmetric: if 'x is older than y' then 'y is not older than x'.
(2) The relation 'x is not younger than y' is a QO. It is not a wPO since it is not antisymmetric. (There are different people of the same age.)

The weak version of the relation in 1. is the relation 'x is older than y or x, y is the same person', which is clearly different from the relation in 2. The relation in 1. is not a TO – of two *different* persons there may be none who is older than the other. □

Example 1.16
Given a set of symbols, for instance $\Sigma = \{a, b, c\}$, the set Σ^* contains all *finite* strings over Σ, e.g., a, aa, $abba$, $aaabbababab$, ... There are various natural ways of ordering Σ^*. Let $s, p \in \Sigma^*$

(1) Define $s \prec_Q p$ iff $length(s) < length(p)$. This gives an sPO. The weak relation $s \preceq_Q p$ iff $length(s) \leq length(p)$ will be a QO but not a wPO since now any subset containing strings of equal length will form a loop.
(2) Define $s \prec_P p$ iff s is a prefix of p. (Prefix is an initial segment.) We now obtain a wPO because any string is its own prefix and if both s is a prefix of p and p is a prefix of s the two must be the same string. This is not, however, a TO : neither of the strings a and bc is a prefix of the other.
(3) Suppose that the set Σ is totally ordered, for instance, let Σ be the Latin alphabet with the standard ordering $a \prec b \prec c \prec d$.... We may then define the *lexicographic* TO on Σ^* as follows:
$s \prec_L p$ iff either s is a prefix of p or else the two have a longest common prefix u (possibly empty) such that $s = uv$, $t = uw$ and $head(v) \prec head(w)$, where $head$ is the first symbol of the argument string. This defines the usual ordering used, for instance, in dictionaries. □

Homomorphisms .... [optional]
The following notions will not be used extensively, but it is often important to realize that the functions between ordered sets should consider not only the elements of the sets but also the relation between these elements.

Definition 1.17 Given two orderings (of any kind) $P = \langle S_1, R_1 \rangle$ and $Q = \langle S_2, R_2 \rangle$, a *homomorphism* $h : P \to Q$ is an *order preserving* function, i.e., a function $h : S_1 \to S_2$ such that $R_1(x, y)$ implies $R_2(h(x), h(y))$.
An *order-isomorphism* is a set-isomorphism $h : S_1 \rightleftharpoons S_2$ such that both h and h^{-1} are homomorphisms.

Thus, an order-isomorphism is a set-isomorphism which, in addition, preserves the structure of the isomorphic relations. One often encounters two ordered sets which are set-isomorphic but not order-isomorphic.

Example 1.18
Given two 4-element sets A and B, consider two sPO's $\mathbf{A} = \langle A, \prec_A \rangle$ and $\mathbf{B} =$

$\langle B, \prec_B \rangle$ as shown below:

A

$$\begin{tikzcd} & a_2 \arrow[dr, dash, "\prec_A"] & \\ a_1 \arrow[ur, dash, "\prec_A"] \arrow[dr, dash, "\prec_A"'] & & a_4 \\ & a_3 \arrow[ur, dash, "\prec_A"'] & \end{tikzcd}$$

B

$$b_1 \prec_B b_2 \prec_B b_3 \prec_B b_4$$

	$f(\_)$	$h_1(\_)$	$h_2(\_)$
a_1	b_2	b_1	b_2
a_2	b_1	b_2	b_3
a_3	b_4	b_3	b_3
a_4	b_3	b_4	b_4

Any injective $f : A \to B$ is also a bijection, i.e., $A \rightleftharpoons B$. However, a homomorphism $h : \mathbf{A} \to \mathbf{B}$ must also satisfy the additional condition, e.g., $a_1 \prec_A a_2$ requires that also $h(a_1) \prec_B h(a_2)$. Thus, for instance, f from the table is not a homomorphism $\mathbf{A} \to \mathbf{B}$ while h_1 and h_2 are.

Now, although the underlying sets A and B are isomorphic, there is no *order-isomorphism* between $\mathbf{A} = \langle A, \prec_A \rangle$ and $\mathbf{B} = \langle B, \prec_B \rangle$ because there is no *homomorphism* from the latter to the former. **B** is connected, i.e., a total order while **A** is not – any homomorphism would have to preserve the relation $\prec_B$ between arbitrary two elements of B, but in A there is no relation $\prec_A$ between a_2 and a_3.
.... [end optional]

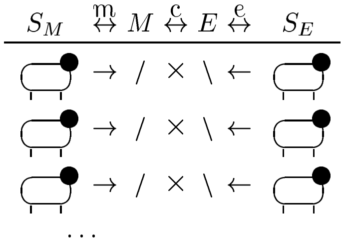
4: Infinities

◇——— A Background Story ———◇

Imagine a primitive shepherd who possesses no idea of number or counting – when releasing his sheep from the cottage in the morning he wants to find the means of figuring out if, when he collects them in the evening, all are back or, perhaps, some are missing.

He can, and most probably did, proceed as follows. Find a stick and let the sheep leave one by one. Each time a sheep passes through the gate, make a mark – something like / – on the stick. When all the sheep have left, there will be as many marks on the stick as there were sheep. On their return, do the same: let them go through the gate one by one. For each sheep, erase one mark – e.g., set a \making one / into ×. When all the sheep are inside, check if there are any /-marks left. If no, i.e., there are only × on the stick, then everything is ok – as many sheep returned home as had left in the morning. If yes, then some sheep are missing.

Notice, that the shepherd still does not know "how many" sheep he has – he still does not have the idea of a number. But, perhaps a bit paradoxically, he has the idea of two equal numbers*! This idea is captured by the correspondance, in fact, several functions: the first is a morning-function m which for each sheep from the set S_M of all* sheep, assigns a new / on the stick – all the marks obtained in the morning form a set M. The other, evening-function e, assigns to each returning sheep (from the set of returning sheep S_E) another mark \. Superimposing the evening marks \ onto the morning marks /, i.e., forming $\times$-marks, amounts to comparing the number of elements in the sets M and E by means of a function c assigning to each returning sheep marked by \, a morning sheep /.



In fact, this simple procedure may be considered as a basis of counting – comparing the number of elements in various sets. In order to ensure that the two sets, like S_M and M, have equal number of elements, we have to insist on some properties of the involved function m. Each sheep must be given a mark (m must be total) and for two distinct sheep we have to make two distinct marks (m must be injective). The third required property – that of surjectivity – follows automatically in the above procedure, since the target set M is formed only along as we mark the sheep. In short, the shepherd knows that there are *as many* sheep *as* the morning marks on the stick because he has a bijective function between the respective sets. For the same reason, he knows that the sets of returning sheep and evening marks have the same number of elements and, finally, establishing a bijection between the sets M and E, he rests satisfied. (Violating a bit the profiles of the involved functions, we may say that the composite function $e\,;c\,;m : S_E \to E \to M \to S_M$ turns out to be bijective.)

You can now easily imagine what is going on when the shepherd discovers that some sheep are missing in the evening. He is left with some /-marks which cannot be converted into $\times$-marks. The function $c : E \to M$ is injective but not surjective. This means that the set E has strictly fewer elements than the set M.

These ideas do not express immediately the concept of a number as we are used to it. (They can be used to do that.) But they do express our intuition about one number being equal to, smaller or greater than another number. What is, perhaps, most surprising is that they work equally well when the involved sets are infinite, thus allowing us to compare "the number of elements" in various infinite sets.

♢————————————————————♢

In this section, we consider only sets and set functions, in particular, set-isomorphisms.

Definition 1.19 Two sets S and T have *equal cardinality* iff they are set-isomorphic $S \rightleftharpoons T$.
The *cardinality* of a set S, $|S|$, is the equivalence class $[S] = \{T : T \rightleftharpoons S\}$.

This is not an entirely precise definition of cardinality which, as a matter of fact, is the *number associated with* such an equivalence class. The point is that it denotes the intuitive idea of *the number of elements* in the set – all set-isomorphic sets have the same number of elements.

Definition 1.20 $|S| \leq |T|$ iff there exists an injective function $f : S \to T$.

It can be shown that this definition is consistent, i.e., that if $|S_1| = |S_2|$ and $|T_1| = |T_2|$ then there exists an injective function $f_1 : S_1 \to T_1$ iff there exists an injective function $f_2 : S_2 \to T_2$. A set S has cardinality stricly less than a set T, $|S| < |T|$, iff there exists an injective function $f : S \to T$ *but there exists no* such surjective function.

Example 1.21
$|\varnothing| = 0$, $|\{\varnothing\}| = 1$, $|\{\{\varnothing\}\}| = 1$, $|\{\varnothing, \{\varnothing\}\}| = 2$.
$|\{a, b, c\}| = |\{\bullet, \#, +\}| = |\{0, 1, 2\}| = 3$.
For finite sets, all operations from Definition 1.1 yield sets with possibly different cardinality:

1.	$\lvert\{a,b\} \cup \{a,c,d\}\rvert = \lvert\{a,b,c,d\}\rvert$	$\lvert S \cup T\rvert \geq \lvert S\rvert$
2.	$\lvert\{a,b\} \cap \{a,c,d\}\rvert = \lvert\{a\}\rvert$	$\lvert S \cap T\rvert \leq \lvert S\rvert$
3.	$\lvert\{a,b\} \setminus \{a,c,d\}\rvert = \lvert\{b\}\rvert$	$\lvert S \setminus T\rvert \leq \lvert S\rvert$
4.	$\lvert\{a,b\} \times \{a,d\}\rvert = \lvert\{\langle a,a\rangle, \langle a,d\rangle, \langle b,a\rangle\langle b,d\rangle\}\rvert$	$\lvert S \times T\rvert = \lvert S\rvert * \lvert T\rvert$
5.	$\lvert\wp(\{a,b\})\rvert = \lvert\{\varnothing, \{a\}, \{b\}, \{a,b\}\}\rvert$	$\lvert\wp(S)\rvert > \lvert S\rvert$

□

From certain assumptions ("axioms") about sets it can be proven that the relation $\leq$ on cardinalities has the properties of a weak TO, i.e., it is reflexive (obvious), transitive (fairly obvious), antisymmetric (not so obvious) and total (less obvious).

.... [optional]

As an example of how intricate reasoning may be needed to establish such "not quite but almost obvious" facts, we show that $\leq$ is antisymmetric.

Theorem 1.22 [Schröder-Bernstein] For arbitrary sets X, Y, if there are injections $i : X \to Y$ and $j : Y \to X$, then there is a bijection $f : X \to Y$ (i.e., if $|X| \leq |Y|$ and $|Y| \leq |X|$ then $|X| = |Y|$).

Proof. If the injection $i : X \to Y$ is surjective, i.e., $i(X) = Y$, then i is a bijection and we are done. Otherwise, we have $Y_0 = Y \setminus i(X) \neq \varnothing$ and we apply j and i repeatively as follows

$$\begin{array}{ll}
Y_0 = Y \setminus i(X) & X_0 = j(Y_0) \\
Y_{n+1} = i(X_n) & X_{n+1} = j(Y_{n+1}) \\
Y^* = \bigcup_{n=0}^{\omega} Y_n & X^* = \bigcup_{n=0}^{\omega} X_n
\end{array}$$

$$i.e., \quad \begin{tikzcd}
Y_0 \arrow[dr, "j"] & Y_1 \arrow[dr, "j"] & Y_2 \arrow[dr, "j"] & {} \arrow[dr, dotted, "j"] & \ldots & Y^* \\
X \arrow[u, "i"] & X_0 \arrow[u, "i"] & X_1 \arrow[u, "i"] & X_2 \arrow[u, dotted, "i"] & \ldots & X^*
\end{tikzcd}$$

So we can divide both sets into disjoint components as in the diagram below.

$$\begin{tikzcd}
Y \arrow[d, shift right, "j"'] & = & Y^* \arrow[d, Leftrightarrow, "j"] & \cup & (Y \setminus Y^*) \arrow[d, Leftrightarrow, "i"'] \\
X \arrow[u, shift right, "i"'] & = & X^* & \cup & (X \setminus X^*)
\end{tikzcd}$$

We show that the respective restrictions of j and i are bijections. First, $j : Y^* \to X^*$ is a bijection (it is injective, and the following equation shows that it is surjective):

$$j(Y^*) = j(\bigcup_{n=0}^{\omega} Y_n) = \bigcup_{n=0}^{\omega} j(Y_n) = \bigcup_{n=0}^{\omega} X_n = X^*$$

By lemma 1.8, $j^- : X^* \to Y^*$, defined by $j^-(x) = y : j(y) = x$ is a bijection too. Furthermore:

$$i(X^*) = i(\bigcup_{n=0}^{\omega} X_n) = \bigcup_{n=0}^{\omega} i(X_n) = \bigcup_{n=0}^{\omega} Y_{n+1} = \bigcup_{n=1}^{\omega} Y_n = Y^* \setminus Y_0. \quad (1.23)$$

Now, the first of the following equalities holds since i is injective, the second by (1.23) and since $i(X) = Y \setminus Y_0$ (definition of Y_0), and the last since $Y_0 \subseteq Y^*$:

$$i(X \setminus X^*) = i(X) \setminus i(X^*) = (Y \setminus Y_0) \setminus (Y^* \setminus Y_0) = Y \setminus Y^*,$$

i.e., $i : (X \setminus X^*) \to (Y \setminus Y^*)$ is a bijection. We obtain a bijection $f : X \to Y$ defind by

$$f(x) = \begin{cases} i(x) & if\ x \in X \setminus X^* \\ j^-(x) & if\ x \in X^* \end{cases} \qquad \textbf{QED} \ (1.22)$$

A more abstract proof

The construction of the sets X^* and Y^* in the above proof can be subsumed under a more abstract formulation implied by the Claims 1. and 2. below. In particular, Claim 1. has a very general form.

Claim 1. For any set X, if $h : \wp(X) \to \wp(X)$ is monotonic, i.e. such that, whenever $A \subseteq B \subseteq X$ then $h(A) \subseteq h(B)$; then there is a set $T \subseteq X : h(T) = T$.

We show that $T = \bigcup\{A \subseteq X : A \subseteq h(A)\}$.

a) $T \subseteq h(T)$: for for each $t \in T$ there is an $A : t \in A \subseteq T$ and $A \subseteq h(A)$. But then $A \subseteq T$ implies $h(A) \subseteq h(T)$, and so $t \in h(T)$.

b) $h(T) \subseteq T$: from a) $T \subseteq h(T)$, so $h(T) \subseteq h(h(T))$ which means that $h(T) \subseteq T$ by definition of T.

Claim 2. Given injections i, j define $\_^* : \wp(X) \to \wp(X)$ by $A^* = X \setminus j(Y \setminus i(A))$. If $A \subseteq B \subseteq X$ then $A^* \subseteq B^*$.

Follows trivially from injectivity of i and j. $A \subseteq B$, so $i(A) \subseteq i(B)$, so $Y \setminus i(A) \supseteq Y \setminus i(B)$, so $j(Y \setminus i(A)) \supseteq j(Y \setminus i(B))$, and hence $X \setminus j(Y \setminus i(A)) \subseteq X \setminus j(Y \setminus i(B))$.

3. Claims 1 and 2 imply that there is a $T \subseteq X$ such that $T = T^*$, i.e., $T = X \setminus j(Y \setminus i(T))$. Then $f : X \to Y$ defined by $f(x) = \begin{cases} i(x) & \text{if } x \in T \\ j^{-1}(x) & \text{if } x \notin T \end{cases}$ is a bijection. We have $X = j(Y \setminus i(T)) \cup T$ and $Y = (Y \setminus i(T)) \cup i(T)$, and obviously j^{-1} is a bijection between $j(Y \setminus i(T))$ and $Y \setminus i(T)$, while i is a bijection between T and $i(T)$. . . . . [end optional]

Cardinality of each finite set is a natural number. The apparently empty Definition 1.19 becomes more significant when we look at the infinite sets.

Definition 1.24 A set S is *infinite* iff there exists a proper subset $T \subset S$ such that $S \rightleftharpoons T$.

Example 1.25

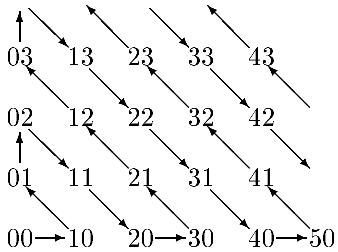
Denote the cardinality of the set of natural numbers by $|\mathbb{N}| \stackrel{\text{def}}{=} \aleph_0$. (Sometimes it is also written ω, although axiomatic set theory distinguishes between the *cardinal number* $\aleph_0$ and the *ordinal number* ω. Ordinal number is a more fine-grained notion than cardinal number, but we shall not worry about this.) We have, for instance, that $|\mathbb{N}| = |\mathbb{N} \setminus \{0\}|$, as shown below to the left. In fact, the cardinality of $\mathbb{N}$ is the same as the cardinality of the even natural numbers! It is easy to see that the pair of functions $f(n) = 2n$ and $f^{-1}(2n) = n$ as shown to the righ:

$$\begin{array}{cccccc} \{ & 0 & 1 & 2 & 3 & \ldots \\ & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \ldots \\ \{ & 1 & 2 & 3 & 4 & \ldots \end{array} \qquad\qquad \begin{array}{ccccccccc} & \{0 & 1 & 2 & 3 & 4 & 5 & & \ldots \\ f\downarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \uparrow f^{-1} & \\ & \{0 & 2 & 4 & 6 & 8 & 10 & & \ldots \end{array}$$

In general, when $|S| = |T| = \aleph_0$ and $|P| = n < \aleph_0$, we have

$$|S \cup T| = \aleph_0 \qquad |S \setminus P| = \aleph_0 \qquad |S \times T| = \aleph_0$$

The following drawing illustrates a possible set-isomorphisms $\mathbb{N} \rightleftharpoons \mathbb{N} \times \mathbb{N}$:



□

A set-isomorphism $S \rightleftharpoons \mathbb{N}$ amounts to an *enumeration* of the elements of S. Thus, if $|S| \leq \aleph_0$ we say that S is *enumerable* or *countable*; in case of equality, we say that it is countably infinite. Now, the question "are there any uncountable sets?" was answered by the founder of modern set theory:

Theorem 1.26 [Georg Cantor] For any set $A : |A| < |\wp(A)|$.

Proof. The construction applied here shows that the contrary assumption – $A \rightleftharpoons \wp(A)$ – leads to a contradiction. Obviously, $|A| \leq |\wp(A)|$, since the inclusion defined by $f(a) = \{a\}$ is an injective function $f : A \to \wp(A)$. So assume the equality $|A| = |\wp(A)|$, i.e., a corresponding $F : A \to \wp(A)$ which is both injective and surjective. Define the subset of A by $B \stackrel{\mathsf{def}}{=} \{a \in A : a \notin F(a)\}$. Since $B \subseteq A$, so $B \in \wp(A)$ and, since F is surjective, there is a $b \in A$ such that $F(b) = B$. Is b in B or not? Each of the two possible answers yields a contradiction:

(1) $b \in F(b)$ means $b \in \{a \in A : a \notin F(a)\}$, which means $b \notin F(b)$
(2) $b \notin F(b)$ means $b \notin \{a \in A : a \notin F(a)\}$, which means $b \in F(b)$.

QED (1.26)

Corollary 1.27 There is no greatest cardinal number.

In particular, $\aleph_0 = |\mathbb{N}| < |\wp(\mathbb{N})| < |\wp(\wp(\mathbb{N}))| < ...$ Theorem 1.26 proves that there exist uncountable sets, but are they of any interest? Another theorem of Cantor shows that such sets have been around in mathematics for quite a while.

Theorem 1.28 The set $\mathbb{R}$ of real numbers is uncountable.

Proof. Since $\mathbb{N} \subset \mathbb{R}$, we know that $|\mathbb{N}| \leq |\mathbb{R}|$. The *diagonalisation* technique introduced here by Cantor, reduces the assumption that $|\mathbb{N}| = |\mathbb{R}|$ ad absurdum. If $\mathbb{R} \rightleftharpoons \mathbb{N}$ then, certainly, we can enumerate any subset of $\mathbb{R}$. Consider only the closed interval $[0,1] \subset \mathbb{R}$. If it is countable, we can list all its members, writing them in decimal expansion (each r_{ij} is a digit):

$$\begin{array}{rl}
n_1 = & 0.\ \mathbf{r_{11}}\ r_{12}\ r_{13}\ r_{14}\ r_{15}\ r_{16}\ .... \\
n_2 = & 0.\ r_{21}\ \mathbf{r_{22}}\ r_{23}\ r_{24}\ r_{25}\ r_{26}\ .... \\
n_3 = & 0.\ r_{31}\ r_{32}\ \mathbf{r_{33}}\ r_{34}\ r_{35}\ r_{36}\ .... \\
n_4 = & 0.\ r_{41}\ r_{42}\ r_{43}\ \mathbf{r_{44}}\ r_{45}\ r_{46}\ .... \\
n_5 = & 0.\ r_{51}\ r_{52}\ r_{53}\ r_{54}\ \mathbf{r_{55}}\ r_{56}\ .... \\
n_6 = & 0.\ r_{61}\ r_{62}\ r_{63}\ r_{64}\ r_{65}\ \mathbf{r_{66}}\ .... \\
\vdots &
\end{array}$$

Form a new real number r by replacing each $\mathbf{r_{ii}}$ with another digit, for instance, let $r = 0.r_1r_2r_3r_4...$, where $r_i = \mathbf{r_{ii}} + 1 \bmod 10$. Then r cannot be any of the listed numbers $n_1, n_2, n_3, ...$ For each such number n_i has a digit $\mathbf{r_{ii}}$ at its i-th position which is different from the digit r_i at the i-th position in r. **QED** (1.28)

"Sets" which are not sets

In Definition 1.1 we introduced several set building operations. The power set operation $\wp(\_)$ has proven particularly powerful. However, the most peculiar one is the *comprehension* operation, namely, the one allowing us to form a set of elements satisfying some property $\{x : \text{Prop}(x)\}$. Although apparently very natural, its unrestricted use leads to severe problems.

Russell's Paradox
Define the set $U = \{x : x \notin x\}$ and say if $U \in U$. Each possible answer leads to absurdity:

(1) $U \in U$ means that U is one of x in U, i.e., $U \in \{x : x \notin x\}$, so $U \notin U$
(2) $U \notin U$ means that $U \in \{x : x \notin x\}$, so $U \in U$. □

The problem arises because in the definition of U we did not specify what kind of x's we are gathering. Among many solutions to this paradox, the most commonly accepted is to exclude such definitions by requiring that x's which are to satisfy a given property when collected into a new set must already belong to some other set. This is the formulation we used in Definition 1.1, where we said that if S is a set then $\{x \in S : \text{Prop}(x)\}$ is a set too. The "definition" of $U = \{x : x \notin x\}$ does not conform to this format and hence is not considered a valid description of a set.

Exercises 1.

EXERCISE 1.1 Given the following sets

$$
\begin{array}{ll}
S1 = \{\{\varnothing\}, A, \{A\}\} & S6 = \varnothing \\
S2 = A & S7 = \{\varnothing\} \\
S3 = \{A\} & S8 = \{\{\varnothing\}\} \\
S4 = \{\{A\}\} & S9 = \{\varnothing, \{\varnothing\}\} \\
S5 = \{A, \{A\}\} &
\end{array}
$$

Of the sets S1-S9, which

(1) are members of S1 ?
(2) are members of S4 ?
(3) are members of S9 ?
(4) are subsets of S1 ?
(5) are subsets of S4 ?
(6) are subsets of S9 ?

EXERCISE 1.2 Let $A = \{a, b, c\}$, $B = \{c, d\}$, $C = \{d, e, f\}$.

(1) Write the sets: $A \cup B$, $A \cap B$, $A \cup (B \cap C)$
(2) Is a a member of $\{A, B\}$, of $A \cup B$?
(3) Write the sets $A \times B$ and $B \times A$.

EXERCISE 1.3 Using the set theoretic equalities (page 5), show that:

(1) $A \cap (B \setminus A) = \varnothing$
(2) $((A \cup C) \cap (B \cup \overline{C})) \subseteq (A \cup B)$

Show first some lemmas:
$a)$ $A \cap B \subseteq A$
$b)$ if $A \subseteq B$ then $A \subseteq B \cup C$
$c)$ if $A_1 \subseteq X$ and $A_2 \subseteq X$ then $A_1 \cup A_2 \subseteq X$

Expand then the expression $(A \cup C) \cap (B \cup \overline{C})$ to one of the form $X_1 \cup X_2 \cup X_3 \cup X_4$, show that each $X_i \subseteq A \cup B$ and use lemma c).

EXERCISE 1.4 Let $S = \{0, 1, 2\}$ and $T = \{0, 1, \{0, 1\}\}$. Construct $\wp(S)$ and $\wp(T)$.

EXERCISE 1.5 Given two infinite sets
$S = \{5, 10, 15, 20, 25, ...\}$ and
$T = \{3, 4, 7, 8, 11, 12, 15, 16, 19, 20, ...\}$.

(1) Specify each of these sets by defining the properties P_S and P_T such that $S = \{x \in \mathbb{N} : P_S(x)\}$ and $T = \{x \in \mathbb{N} : P_T(x)\}$
(2) For each of these sets specify two other properties P_{S1}, P_{S2} and P_{T1}, P_{T2}, such that $S = \{x \in \mathbb{N} : P_{S1}(x)\} \cup \{x \in \mathbb{N} : P_{S2}(x)\}$ and similarly for T.

EXERCISE 1.6 Prove the claims cited below Definition 1.13, that

(1) Every sPO (irreflexive, transitive relation) is asymmetric.
(2) Every asymmetric relation is antisymmetric.
(3) If R is connected, symmetric and reflexive, then $R(s, t)$ for every pair s, t. What about a relation that is both connected, symmetric and transitive? In what way does this depend on the cardinality of S?

EXERCISE 1.7 Let C be a collection of sets. Show that equality $=$ and existence of set-isomorphism $\rightleftharpoons$ are equivalence relations on $C \times C$ as claimed under Definition 1.13. Give an example of two sets S and T such that $S \rightleftharpoons T$ but $S \neq T$ (they are set-isomorphic but not equal).

EXERCISE 1.8 For any (non-empty) collection of sets C, show that

(1) the inclusion relation $\subseteq$ is a wPO on C
(2) $\subset$ is its strict version
(3) $\subseteq$ is not (necessarily) a TO on C.

EXERCISE 1.9 If $|S| = n$ for some natural number n, what will be the cardinality of $\wp(S)$?

EXERCISE 1.10 Let A be a countable set.

(1) If also B is countable, show that:
    (a) the disjoint union $A \uplus B$ is countable (specify its enumeration, assuming the existence of the enumerations of A and B);
    (b) the union $A \cup B$ is countable (specify an injection into $A \uplus B$).
(2) If B is uncountable, can $A \times B$ ever be countable?

EXERCISE 1.11 Let A be a countable set. Show that A has countably many *finite* subsets, proceeding as follows:

(1) Show first that for any $n \in \mathbb{N}$, the set $\wp^n(A)$ of finite subsets – with exactly n elements – of A is countable.
(2) Using a technique similar to the one from Example 1.25, show that the union $\bigcup_{n \in \mathbb{N}} \wp^n(A)$ of all these sets is countable.

Chapter 2
Induction

- Well-founded Orderings
  - General notion of Inductive proof
- Inductive Definitions
  - Structural Induction

1: Well-Founded Orderings

◇——— a Background Story ———◇

Ancients had many ideas about the basic structure and limits of the world. According to one of them our world – the earth – rested on a huge tortoise. The tortoise itself couldn't just be suspended in a vacuum – it stood on the backs of several elephants. The elephants stood all on a huge disk which, in turn, was perhaps resting on the backs of some camels. And camels? Well, the story obviously had to stop somewhere because, as we notice, one could produce new sub-levels of animals resting on other objects resting on yet other animals, resting on ... indefinitely. The idea is not well founded because such a hierarchy has no well defined begining, it hangs in a vacuum. Any attempt to provide the last, the most fundamental level is immediately met with the question "And what is beyond that?"

The same problem of the lacking foundation can be encoutered when one tries to think about the begining of time. When was it? Physicists may say that it was Big Bang. But then one immediately asks "OK, but what was before?". Some early opponents of the Biblical story of creation of the world – and thus, of time as well – asked "What did God do before *He created time?". St. Augustine, realising the need for a definite answer which, however, couldn't be given in the same spirit as the question, answered "He prepared the hell for those asking such questions."*

One should be wary here of the distinction between the begining and the end, or else, between moving backward and forward. For sure, we imagine that things, the world may continue to exist indefinitely in *the future – this idea does not cause much trouble. But our intuition is uneasy with things which do not have any begining, with chains of events extending indefinitely backwards, whether it is a backward movement along the dimension of time or causality.*

Such non well founded *chains are hard to imagine and even harder to do anything with – all our thinking, activity, constructions have to start from some begining. Having an idea of a begining, one will often be able to develop it into a description of the ensuing process. One will typically say: since the begining was so-and-so, such-and-such had to follow since it is implied by the properties of the begining. Then, the properties of this second stage, imply some more, and so on. But having nothing to start with, we are left without foundation to perform any intelligible acts.*

Mathematics has no problems with chains extending infinitely in both directions. Yet, it has a particular liking for chains which do have a begining, for orderings which are well-founded. As with our intuition and activity otherwise, the possibility of ordering a set in a way which identifies its least, first, starting elements, gives a mathematician a lot of powerful tools. We will study in this chapter some fundamental tools of this kind. As we will see later, almost all our presentation will be based on well-founded orderings.

$\diamondsuit$ ——————————————————————— $\diamondsuit$

Definition 2.1 Let $\langle S, \leq \rangle$ be a PO and $T \subseteq S$.

- $x \in T$ is a *minimal element* of T iff there is no element smaller than x, i.e., for no $y \in T : y < x$
- $\langle S, \leq \rangle$ is *well-founded* iff each non-empty $T \subseteq S$ has a minimal element.

The set of natural numbers with the standard ordering $\langle \mathbb{N}, \leq \rangle$ is well-founded, but the set of all integers with the natural extension of this ordering $\langle \mathbb{Z}, \leq \rangle$ is not – the subset of all negative integers does not have a $\leq$-minimal element. Intuitively, well-foundedness means that the ordering has a "basis", a set of minimal "starting points". This is captured by the following lemma.

Lemma 2.2 A PO $\langle S, \leq \rangle$ is well-founded iff there is no infinite decreasing sequence, i.e., no sequence $\{a_n\}_{n \in \mathbb{N}}$ of elements of S such that $a_n > a_{n+1}$.

Proof. We have to show two implications.

$\Leftarrow$) If $\langle S, \leq \rangle$ is not well-founded, then let $T \subseteq S$ be a subset without a minimal element. Let $a_1 \in T$ – since it is not minimal, we can find $a_2 \in T$ such that $a_1 > a_2$. Again, a_2 is not minimal, so we can find $a_3 \in T$ such that $a_2 > a_3$. Continuing this process we obtain an infinite descending sequence $a_1 > a_2 > a_3 > ....$

$\Rightarrow$) If there is such a sequence $a_1 > a_2 > a_3 > ...$ then, obviously, the set $\{a_n : n \in \mathbb{N}\} \subseteq S$ has no minimal element. **QED** (2.2)

Example 2.3

Consider again the orderings on finite strings as defined in example 1.16.

(1) The relation $\prec_Q$ is a well-founded sPO; there is no way to construct an infinite sequence of strings with ever decreasing lengths!

(2) The relation $\prec_P$ is a well-founded PO : any subset of strings will contain element(s) such that none of their prefixes (except the strings themselves) are in the set. For instance, a and bc are $\prec_P$-minimal elements in $S = \{ab, abc, a, bcaa, bca, bc\}$.

(3) The relation $\prec_L$ is not well-founded, since there exist infinite descending sequences like

$$\ldots \prec_L \; aaaab \; \prec_L \; aaab \; \prec_L \; aab \; \prec_L \; ab \; \prec_L \; b.$$

In order to construct any such descending sequence, however, there is a need to introduce ever longer strings as we proceed towards infinity. Hence the alternative ordering below is also of interest.

(4) The relation $\prec_Q$ was defined in example 1.16. Now define $s \prec_{L'} p$ iff $s \prec_Q p$ or ($length(s) = length(p)$ and $s \prec_L p$). Hence sequences are ordered primarily by length, secondarily by the previous lexicographic order. The ordering $\prec_{L'}$ is indeed well-founded and, in addition, connected, i.e., a well-founded TO. □

Definition 2.4 $\langle S, \leq \rangle$ is a well-ordering, WO, iff it is a TO and is well-founded.

Notice that well-founded ordering is *not* the same as well-ordering. The former can still be a PO which is not a TO. The requirement that a WO $= \langle S, \leq \rangle$ is a TO implies that each (sub)set of S has not only a minimal element but also a *unique* minimal element.

Example 2.5

The set of natural numbers with the "less than" relation, $\langle \mathbb{N}, < \rangle$, is an sPO. It is also a TO (one of two distinct natural numbers must be smaller than the other) and well-founded (any non-empty set of natural numbers contains a least element). □

Although sets like $\mathbb{N}$ or $\mathbb{Z}$ have the natural orderings, these are not the only possible orderings of these sets. In particular, for a given set S there may be several different ways of imposing a well-founded ordering on it.

Example 2.6
The set of integers, $\langle \mathbb{Z}, < \rangle$, is a TO but not well-founded – negative numbers have no minimal element.

That $<$ and $\leq$ fail to be WO on $\mathbb{Z}$ does not mean that $\mathbb{Z}$ cannot be made into a WO. One has to come up with another ordering. For instance, let $|x|$ for an $x \in \mathbb{Z}$ denote the absolute value of x (i.e., $|x| = x$ if $x \geq 0$ and $|x| = -x$ if $x < 0$.) Say that $x \prec y$ if either $|x| < |y|$ or ($|x| = |y|$ and $x < y$). This means that we order $\mathbb{Z}$ as $0 \prec -1 \prec 1 \prec -2 \prec 2....$ This $\prec$ is clearly a WO on $\mathbb{Z}$.

Of course, there may be many different WO's on a given set. Another WO on $\mathbb{Z}$ could be obtained by swapping the positive and negative integers with the same absolute values, i.e., $0 \prec' 1 \prec' -1 \prec' 2 \prec' -2....$ □

1.1: Inductive Proofs on Well-founded Orderings

Well-founded orderings play a central role in many contexts because they allow one to apply a particularly convenient proof technique – proof by induction – which we now proceed to study.

◇——— a Background Story ———◇

Given some set S, a very typical problem is to show that all elements of S satisfy some property, call it P, i.e., to show that for all $x \in S$: $P(x)$. How one can try to prove such a fact depends on how the set S is described.

A special case is when S is finite and has only few elements – in this case, we can just start proving $P(x)$ for each x separately.

A more common situation is that S has infinitely many elements. Let $S = \{2i : i \in \mathbb{Z}\}$ and show that each $x \in S$ is an even number. Well, this is trivial by the way we have defined the set. Let x be an arbitrary element of S. Then, by definition of S, there is some $i \in \mathbb{Z}$ such that $x = 2i$. But this means precisely that x is an even number and, since x was assumed arbitrary, the claim holds for all $x \in S$.

Of course, in most situations, the relation between the definition of S and the property we want to prove isn't that simple. Then the *question arises: "How to ensure that we check the property for* all *elements of S and that we can do it in finite time (since otherwise we would never finish our proof)?" The idea of proof by induction answers this question in a particular way. It tells us that we have to find some well-founded ordering of the elements of S and then proceed in a prescribed fashion: one shows the statement for the minimal elements and then proceeds to greater elements in the ordering. The trick is that the strategy ensures that only finitely many steps of the proof are needed in order to conclude that the statement holds for* all *elements of S.*

The inductive proof strategy is not guaranteed to work in all cases and, particularly, it depends heavily on the choice of the ordering. It is, nevertheless, a very powerfull proof technique which will be of crucial importance for all the rest of the material we will study.

$\diamond$————————————————————$\diamond$

The most general and abstract statement of the inductive proof strategy is as follows.

Theorem 2.7 Let $\langle S, \leq \rangle$ be a well-founded ordering and $T \subseteq S$.
Assume the following condition: for all $y \in S$: if (for all $x \in S : x < y \to x \in T$) then $y \in T$. Then $T = S$.

> **Proof.** Assume that T satisfies the condition but $T \neq S$, i.e., $S \setminus T \neq \varnothing$. Since S is well-founded, $S \setminus T$ must have a minimal element y. Since y is minimal in $S \setminus T$, any $x < y$ must be in T. But then the condition implies $y \in T$. This is a contradiction – we cannot have both $y \in T$ and $y \in S \setminus T$ – showing that $T = S$. **QED** (2.7)

This theorem of *induction* is the basis for the following proof strategy for showing properties of sets on which some well-founded ordering has been defined.

Idea 2.8 [**Inductive proof**] Let $\langle S, \leq \rangle$ be well-founded and P be a predicate. Suppose we want to prove that each element $x \in S$ has the property P – that $P(x)$ holds for all $x \in S$. I.e., we want to prove that the sets $T = \{x \in S : P(x)\}$ and S are equal. Proceed as follows:

INDUCTION :: Let x be an arbitrary element of S, and assume that for all $y < x : P(y)$ holds. Prove that this implies that also $P(x)$ holds.

CLOSURE :: If you managed to show this, you may conclude $S = T$, i.e., $P(x)$ holds for all $x \in S$.

Observe that the hypothesis in the INDUCTION step, called the *induction hypothesis*, IH, allows us to assume $P(y)$ for all $y < x$. Since we are working with a well-founded ordering, there are some minimal elements x for which no such y exists. For these minimal x's, we then have no hypothesis and simply have to show that the claim $P(x)$ holds for them without any assumptions. This part of the proof is called the BASIS of induction.

Example 2.9

Consider the natural numbers greater than 1, i.e. the set $\mathbb{N}_2 = \{n \in \mathbb{N} : n \geq 2\}$. We want to prove the *prime number theorem*: for each $n \in \mathbb{N}_2 : P(n)$, where $P(n)$ stands for 'n is a product of prime numbers'. First we have to decide which well-founded ordering on $\mathbb{N}_2$ to use – the most obvious first choice is to try the natural ordering $<$, that is, we prove the statement on the well-founded ordering $\langle \mathbb{N}_2, < \rangle$:

BASIS :: Since 2 – the minimal element in $\mathbb{N}_2$ – is a prime number, we have $P(2)$.

IND. :: So let $n > 2$ and assume IH: that $P(k)$ holds for every $k < n$. If n is prime, $P(n)$ holds trivially. So, finally, assume that n is a non-prime number greater than 2. Then $n = x * y$ for some $2 \leq x, y < n$. By IH, $P(x)$ and $P(y)$, i.e., x and y are products of primes. Hence, n is a product of primes.

CLSR. :: So $P(n)$ for all $n \in \mathbb{N}_2$. □

Example 2.10

For any number $x \neq 1$ and for any $n \in \mathbb{N}$, we want to show: $1 + x + x^2 + ... + x^n = \frac{x^{n+1}-1}{x-1}$. There are two different sets involved (of x's and of n's), so we first try the easiest way – we attempt induction on the well-founded ordering $\langle \mathbb{N}, < \rangle$, which is simply called "induction on n":

BASIS :: For $n = 0$, we have $1 = \frac{x-1}{x-1} = 1$.

IND. :: Let $n' > 0$ be arbitrary, i.e., $n' = n + 1$. We expand the left hand side of the equality:

$$\begin{aligned}
1 + x + x^2 + ... + x^n + x^{n+1} &= (1 + x + x^2 + ... + x^n) + x^{n+1} \\
\text{(by IH since } n < n' = n+1) &= \frac{x^{n+1} - 1}{x - 1} + x^{n+1} \\
&= \frac{x^{n+1} - 1 + (x-1)x^{n+1}}{x - 1} \\
&= \frac{x^{n+1+1} - 1}{x - 1}
\end{aligned}$$

Notice that here we have used much weaker IH – the hypothesis that the claim holds for $n = n' - 1$ (implied by IH) is sufficient to establish the induction step. CLOSURE yields the claim *for all* $n \in \mathbb{N}$. □

The proof rule from the idea 2.8 used in the above examples may be written more succinctly as

$$\frac{\forall x(\forall y(y < x \to P(y)) \to P(x))}{\forall x P(x)} \tag{2.11}$$

where "$\forall x$" is short for "for all $x \in \mathbb{N}_2$" (in 2.9), resp. "for all $x \in \mathbb{N}$" (in 2.10) and the horizontal line indicates that the sentence below can be inferred from the sentence above.

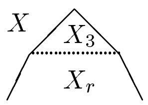
Example 2.12

A *convex n-gon* is a polygon with n sides and where each interior angle is less than 180°. A triangle is a convex 3-gon and, as you should know from the basic geometry, the sum of the interior angles of a triangle is 180°. Now, show by induction that the sum of interior angles of any convex n-gon is $(n-2)180°$.

The first question is: induction *on what*? Here it seems natural to try induction on n, i.e., on the number of sides. (That is, we consider a well-founded ordering on n-gons in which $X < Y$ iff X has fewer sides than Y.)

The basis case is: let X be an arbitrary triangle, i.e., 3-gon. We use the known result that the sum of interior angles of any triangle is indeed 180°.

For the induction step: let $n > 3$ be arbitrary number and X an arbitrary convex n-gon. Selecting two vertices with one common neighbour vertex between them, we can always divide X into a triangle X_3 and $(n-1)$-gon X_r, as indicated by the dotted line on the drawing below.



X_r has one side less than X so, by induction hypothesis, we have that the sum of its angles is $(n-3)180°$. Also by IH, the sum of angles in X_3 is 180°. At the same time, the sum of the angles in the whole X is simply the sum of angles in X_3 and X_r. Thus it equals $(n-3)180° + 180° = (n-2)180°$ and the proof is complete. □

The simplicity of the above examples is due to not only the fact that the problems are easy but also that the ordering to be used is very easy to identify. In general, however, there may be different orderings on a given set and then the first question about inductive proof concerns the choice of appropriate ordering.

Example 2.13
We want to prove that for all integers $z \in \mathbb{Z}$:
$$\begin{cases} 1+3+5+...+(2z-1) = z^2 & if\ z > 0 \\ (-1)+(-3)+(-5)+...+(2z+1) = -(z^2) & if\ z < 0 \end{cases}.$$
We show examples of two proofs using different orderings.

(1) For the first, this looks like two different statements, so we may try to prove them separately for positive and negative integers. Let's do it:

BASIS :: For $z = 1$, we have $1 = 1^2$.
IND. :: Let $z' > 1$ be arbitrary, i.e., $z' = z + 1$ for some $z > 0$.

$$\begin{aligned} 1+3+...+(2z'-1) &= 1+3+...+(2z-1)+(2(z+1)-1) \\ \text{(by IH since } z < z' = z+1) &= z^2+2z+1 \ = \ (z+1)^2 \ = \ (z')^2 \end{aligned}$$

The proof for $z < 0$ is entirely analogous, but now we have to reverse the ordering: we start with $z = -1$ and proceed along the negative integers only considering $z \prec z'$ iff $|z| < |z'|$, where $|z|$ denotes the absolute value of z (i.e., $|z| = -z$ for $z < 0$). Thus, for $z, z' < 0$, we have actually that $z \prec z'$ iff $z > z'$.

BASIS :: For $z = -1$, we have $-1 = -(-1)^2$.
IND. :: Let $z' < -1$ be arbitrary, i.e., $z' = z - 1$ for some $z < 0$.

$$\begin{aligned} -1-3+...+(2z'+1) &= -1-3+...+(2z+1)+(2(z-1)+1) \\ \text{(by IH since } z \prec z') &= -|z|^2-2|z|-1 \\ &= -(|z|^2+2|z|+1) \ = \ -(|z|+1)^2 \\ &= -(z-1)^2 \ = \ -(z')^2 \end{aligned}$$

The second part of the proof makes it clear that the well-founded ordering on the whole $\mathbb{Z}$ we have been using was not the usual $<$. We have, in a sense, ordered both segments of positive and negative numbers independently in the following way (the arrow $x \to y$ indicates the ordering $x \prec y$):

$$1 \longrightarrow 2 \longrightarrow 3 \longrightarrow 4 \longrightarrow 5 \longrightarrow \cdots$$

$$-1 \longrightarrow -2 \longrightarrow -3 \longrightarrow -4 \longrightarrow -5 \longrightarrow \cdots$$

The upper part coincides with the typically used ordering $<$ but the lower one was the matter of more specific choice.

Notice that the above ordering is different from another, natural one, which orders two integers $z \prec' z'$ iff $|z| < |z'|$. This ordering (shown below) makes, for instance, $3 \prec' -4$ and $-4 \prec' 5$. The ordering above did not relate any pair of positive and negative intergers with each other.

$$\begin{tikzcd}
1 \arrow[r] \arrow[dr] & 2 \arrow[r] \arrow[dr] & 3 \arrow[r] \arrow[dr] & 4 \arrow[r] \arrow[dr] & 5 \arrow[r] \arrow[dr] & \cdots \\
-1 \arrow[r] \arrow[ur] & -2 \arrow[r] \arrow[ur] & -3 \arrow[r] \arrow[ur] & -4 \arrow[r] \arrow[ur] & -5 \arrow[r] \arrow[ur] & \cdots
\end{tikzcd}$$

(2) The problem can be stated a bit differently. We have to show that

$$\text{for all } z \in \mathbb{Z} : \begin{cases} 1+3+5+\ldots+(2z-1) = z^2 & \text{if } z > 0 \\ -(1+3+5+\ldots+(2|z|-1)) = -(z^2) & \text{if } z < 0 \end{cases}.$$

Thus formulated, it becomes obvious that we only have one statement to prove: we show the first claim (in the same way as we did it in point 1.) and then apply trivial arithmetics to conclude from $x = y$ that also $-x = -y$.

This, indeed, is a smarter way to prove the claim. Is it induction? Yes it is. We prove it first for all positive integers – by induction on the natural ordering of the positive integers. Then, we take an arbitrary negative integer z and observe (assume induction hypothesis!) that we have already proved $1+3+\ldots+(2|z|-1) = |z|^2$. The well-founded ordering on $\mathbb{Z}$ we are using in this case orders first all positive integers along $<$ (for proving the first part of the claim) and then, for any $z < 0$, puts z after $|z|$ but unrelated to other $n > |z| > 0$ – the induction hypothesis for proving the claim for such a $z < 0$ is, namely, that the claim holds for $|z|$. The ordering is shown on the left:

$$\begin{tikzcd}
1 \arrow[r] \arrow[d] & 2 \arrow[r] \arrow[d] & 3 \arrow[r] \arrow[d] & 4 \arrow[r] \arrow[d] & \cdots \\
-1 & -2 & -3 & -4 & \cdots
\end{tikzcd}
\qquad
\begin{tikzcd}
 & & & & & -1 \\
 & & & & & -2 \\
1 \arrow[r] & 2 \arrow[r] & 3 \arrow[r] & 4 \arrow[r] & \cdots \arrow[uur] \arrow[ur] \arrow[r] \arrow[dr] \arrow[ddr] & -3 \\
 & & & & & -4 \\
 & & & & & \cdots
\end{tikzcd}$$

As a matter of fact, the structure of this proof allows us to view the used ordering in yet another way. We first prove the claim for *all* positive integers. Thus, when proving it for an arbitrary negative integer $z < 0$, we can assume the stronger statement than the one we are actually using, i.e., that the claim holds for *all* positive integers. This ordering puts all negative integers after all positive ones as shown on the right in the figure above.

None of the orderings we encountered in this example was total. $\square$

2: Inductive Definitions

We have introduced the *general* idea of inductive proof over an *arbitrary* well-founded ordering defined on an arbitrary set. The idea of induction – a kind of stepwise construction of the whole from a "basis" by repetitive applications of given rules – can be applied not only for constructing proofs but also for constructing, that is defining, sets. We now illustrate this technique of *definition by induction* and then (subsection 2.3) proceed to show how it gives rise to the possibility of using a special case of the inductive proof strategy – the *structural induction* – on sets defined in this way.

◇—— a Background Story ——◇

Suppose I make a simple statement, for instance, (1) 'John is a nice person'. Its truth may be debatable – some people may think that, on the contrary, he is not nice at all. Pointing this out, they might say – "No, he is not, you only think that he is". So, to make my statemet less definite I might instead say (2) 'I think that 'John is a nice person''. In the philosophical tradition one would say that (2) expresses a reflection over (1) – it expresses the act of reflecting over the first statement. But now, (2) is a new statement, and so I can reflect over it again: (3) 'I think that 'I think that 'John is nice'''. It isn't perhaps obvious why I should make this kind of statement, but I certainly can make it and, with some effort, perhaps even attach some meaning to it. Then, I can just continue: (4) 'I think that (3)', (5) 'I think that (4)', etc. The further (or higher) we go, the less idea we have what one might possibly intend with such expressions. Philosophers used to spend time analysing their possible meaning – the possible meaning of such repetitive acts of reflection over reflection over reflection ... over something. In general, they agree that such an infinite regress does not yield anything intuitively meaningful and should be avoided.

In the daily discourse, we hardly ever attempt to carry such a process beyond the level (2) – the statements at the higher levels do not make any meaningful contribution to a conversation. Yet they are possible for purely linguistic reasons – each statement obtained in this way is grammatically correct. And what is 'this way'? Simply:

Basis :: Start with some statement, e.g., (1) 'John is nice'.

Step :: Whenever you have produced some statement (n) – at first, it is just (1), but after a few steps, you have some

> higher statement (n) – you may produce a new statement by prepending (n) with 'I think that ...'. Thus you obtain a new, (n+1), statement 'I think that (n)'.

Anything you obtain according to this rule happens to be grammatically correct and the whole infinite chain of such statements consitutes what philosophers call an infinite regress.

The crucial point here is that we do not start with some set which we analyse. We are *defining a new set* – the set of statements $\{(1), (2), (3), ...\}$ – in a peculiar way. The idea of induction – stepwise construction from a "basis" – is not applied for proving properties of a given set but for defining a new one.

◇————————————————————◇

One may often encounter sets described by means of abbreviations like $E = \{0, 2, 4, 6, 8, ...\}$ or $T = \{1, 4, 7, 10, 13, 16, ...\}$. The abbreviation ... indicates that the author assumes that you have figured out what the subsequent elements will be – and that there will be infinitely many of them. It is assumed that you have figured out *the rule* by which to generate all the elements. The same sets may be defined more precisely with the explicit reference to the respective rule:

$$E = \{2 * n : n \in \mathbb{N}\} \quad \text{and} \quad T = \{3 * n + 1 : n \in \mathbb{N}\} \tag{2.14}$$

Another way to describe these rules is as follows. The set E is defined by:

Basis :: $0 \in E$ and,
Step :: whenever an $x \in E$, then also $x + 2 \in E$.
Clsr. :: Nothing else belongs to E.

The other set is defined similarly:

Basis :: $1 \in T$ and,
Step :: whenever an $x \in T$, then also $x + 3 \in T$.
Clsr. :: Nothing else belongs to T.

Here we are not so much defining the whole set by one static formula (as we did in (2.14)), as we are specifying the rules for *generating new* elements from some elements which we have already included in the set. Not all formulae (static rules, as those used in (2.14)) allow equivalent formulation in terms of such generation rules. Yet, quite many sets of interest can be defined by means of such generation rules – quite many sets can be introduced by means of *inductive definitions*. Inductively defined sets will play a central role in all the subsequent chapters.

Idea 2.15 [Inductive definition of a set] An *inductive definition* of a set S consists of

BASIS :: List some (at least one) elements $B \subseteq S$.
IND. :: Give one or more rules to construct new elements of S from already existing elements.
CLSR. :: State that S consists exactly of the elements obtained by the basis and induction steps.

The closure condition is typically assumed rather than stated explicitly, and we will not mention it either.

Example 2.16
The finite strings Σ^* over an alphabet Σ from Example 1.16, can be defined inductively, staring with the empty string, ϵ, i.e., the string of length 0, as follows:

BASIS :: $\epsilon \in \Sigma^*$
IND. :: if $s \in \Sigma^*$ then $xs \in \Sigma^*$ for all $x \in \Sigma$

Constructors are the empty string ϵ and the operations prepending an element in front of a string $x\_$, for all $x \in \Sigma$. Notice that 1-element strings like x will be here represented as $x\epsilon$. □

Example 2.17
The finite non-empty strings Σ^+ over alphabet Σ are defined by starting with a different basis.

BASIS :: $x \in \Sigma^+$ for all $x \in \Sigma$
IND. :: if $s \in \Sigma^+$ then $xs \in \Sigma^+$ for all $x \in \Sigma$ □

Often, one is not interested in all possible strings over a given alphabet but only in some subsets. Such subsets are called *languages* and, typically, are defined by induction.

Example 2.18
Define the set of strings $\mathbb{N}$ over $\Sigma = \{0, s\}$:

BASIS :: $0 \in \mathbb{N}$
IND. :: If $n \in \mathbb{N}$ then $sn \in \mathbb{N}$

This language is the basis of the formal definition of natural numbers. The constructors are 0 and the operation of appending the symbol 's' to the left. (The 's' signifies the "successor" function corresponding to $n + 1$.)

Notice that we do not obtain the set $\{0, 1, 2, 3...\}$ but $\{0, s0, ss0, sss0...\}$, which is a kind of unary representation of natural numbers. Notice that, for instance, the strings $00s, s0s0 \notin \mathbb{N}$, i.e., $\mathbb{N} \neq \Sigma^*$. □

Example 2.19

(1) Let $\Sigma = \{a, b\}$ and let us define the language $L \subseteq \Sigma^*$ consisting only of the strings starting with a number of a's followed by the equal number of b's, i.e., $\{a^n b^n : n \in \mathbb{N}\}$.

    BASIS :: $\epsilon \in L$
    IND. :: if $s \in L$ then $asb \in L$

    Constructors of L are ϵ and the operation adding an a in the beginning and a b at the end of a string $s \in L$.

(2) Here is a more complicated language over $\Sigma = \{a, b, c, (,), \neg, \rightarrow\}$ with two rules of generation.

    BASIS :: $a, b, c \in L$
    IND. :: if $s \in L$ then $\neg s \in L$
    if $s, r \in L$ then $(s \rightarrow r) \in L$

    By the closure property, we can see that, for instance, '(' $\notin L$ and $(\neg b) \notin L$. □

In the examples from section 1 we saw that a given set may be *endowed* with various well-founded orderings. Having succeded in this, we can than use the powerful technique of proof by induction according to theorem 2.7. The usefulness of inductive *definitions* is related to the fact that such an ordering may be obtained for free – the resulting set obtains implicitly a well-founded ordering induced by the very definition as follows.[1]

Idea 2.20 **[Induced wf Order]** For an inductively defined set S, define a function $f : S \rightarrow \mathbb{N}$ as follows:

BASIS :: Let $S_0 = B$ and for all $b \in S_0 : f(b) \stackrel{\text{def}}{=} 0$.
IND. :: Given S_i, let S_{i+1} be the union of S_i and all the elements $x \in S \setminus S_i$ which can be obtained according to one of the rules from some elements $y_1, \ldots, y_n$ of S_i. For each such new $x \in S_{i+1} \setminus S_i$, let $f(x) \stackrel{\text{def}}{=} i + 1$.
CLSR. :: The actual ordering is then $x \prec y$ iff $f(x) < f(y)$.

[1] In fact, an inductive definition imposes at least two such orderings of interest, but here we consider just one.

The function f is essentially counting the minimal number of steps – consecutive applications of the rules allowed by the induction step of Definition 2.15 – needed to obtain a given element of the inductive set.

Example 2.21
Refer to Example 2.16. Since the induction step amounts there to increasing the length of a string by 1, following the above idea, we would obtain the ordering on strings $s \prec p$ iff $length(s) < length(p)$. □

2.1: "1-1" Definitions

A common feature of the above examples of inductively defined sets is the impossibility of deriving an element in more than one way. For instance in Example 2.17, the only way to derive the string abc is to start with c and then add b and a to the left in sequence. One apparently tiny modification changes this state of affairs:

Example 2.22
The finite non-empty strings over alphabet Σ can also be defined inductively as follows.

Basis :: $x \in \Sigma^+$ for all $x \in \Sigma$
Ind. :: if $s \in \Sigma^+$ and $p \in \Sigma^+$ then $sp \in \Sigma^+$ □

According to this example, abc can be derived by concatenating either a and bc, or ab and c. We often say that the former definitions are 1-1, while the latter is not. Given a 1-1 inductive definition of a set S, there is an easy way to define new functions on S – again by induction.

Idea 2.23 [**Inductive function definition**] Suppose S is defined inductively from basis B and a certain set of construction rules. To define a function f on elements of S do the following:

Basis :: Identify the value $f(x)$ for each x in B.
Ind. :: For each way an $x \in S$ can be constructed from one or more $y_1, \ldots, y_n \in S$, show how to obtain $f(x)$ from the values $f(y_1), \ldots, f(y_n)$.
Clsr. :: If you managed to do this, then the closure property of S guarantees that f is defined for all elements of S.

The next few examples illustrate this method.

Example 2.24

We define the *length* function on finite strings by induction on the definition in example 2.16 as follows:

$$\begin{array}{rl} \text{Basis} :: & length(\epsilon) = 0 \\ \text{Ind.} :: & length(xs) = length(s) + 1 \end{array} \qquad \Box$$

Example 2.25

We define the concatenation of finite strings by induction on the definition from example 2.16:

$$\begin{array}{rl} \text{Basis} :: & \epsilon \cdot t = t \\ \text{Ind.} :: & xs \cdot t = x(s \cdot t) \end{array} \qquad \Box$$

Example 2.26

In Example 2.16 strings were defined by a left append (prepend) operation which we wrote as juxtaposition xs. A corresponding right append operation can be now defined inductively.

$$\begin{array}{rl} \text{Basis} :: & \epsilon \vdash y = y\epsilon \\ \text{Ind.} :: & xs \vdash y = x(s \vdash y) \end{array}$$

and the operation of reversing a string:

$$\begin{array}{rl} \text{Basis} :: & \epsilon^R = \epsilon \\ \text{Ind.} :: & (xs)^R = s^R \vdash x \end{array} \qquad \Box$$

The right append operation $\vdash$ does not quite fit the format of idea 2.23 since it takes two arguments – a symbol as well as a string. It is possible to give a more general version that covers such cases as well, but we shall not do so here. The definition below also apparently goes beyond the format of idea 2.23, but in order to make it fit we merely have to think of addition, for instance in $m+n$, as an application of the one-argument function *add* n to the argument m.

Example 2.27

Using the definition of $\mathbb{N}$ from Example 2.18, we can define the plus operation for all $n, m \in \mathbb{N}$:

$$\begin{array}{rl} \text{Basis} :: & 0 + n = n \\ \text{Ind.} :: & s(m) + n = s(m + n) \end{array}$$

It is not obvious that this is the usual addition. For instance, does it hold that $n + m = m + n$? We shall verify this in an exercise.

We can use this definition to calculate the sum of two arbitrary natural numbers represented as elements of $\mathbb{N}$. For instance, $2 + 3$ would be processed as follows:

$$ss0 + sss0 \mapsto s(s0 + sss0) \mapsto ss(0 + sss0) \mapsto ss(sss0) = sssss0 \qquad \Box$$

Note carefully that the method of inductive function definition 2.23 is guaranteed to work only when the set is given by a 1-1 definition. Imagine that we tried to define a version of the length function in example 2.24 by induction on the definition in example 2.22 as follows: $len(x) = 1$ for $x \in \Sigma$, while $len(ps) = len(p) + 1$. This would provide us with alternative (and hence mutually contradictive) values for $len(abc)$, depending on which way we choose to derive abc. Note also that the two equations $len(x) = 1$ and $len(ps) = len(p) + len(s)$ provide a working definition, but in this case it takes some reasoning to check that this is indeed the case.

2.2: Inductive Definitions, Recursive Programming

If you are not familiar with the basics of programming you may skip this subsection and go directly to subsection 2.3. No new concepts are introduced here but merely illustrattions of the relation between the two areas from the title.

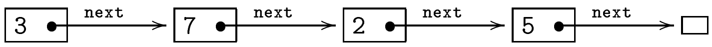
All basic structures known from computer science are defined inductively – any instance of a `List`, `Stack`, `Tree`, etc., is generated in finitely many steps by applying some basic constructor operations. These operations themselves may vary from one programming language to another, or from one application to another, but they always capture the inductive structure of these data types. We give here but two simple examples which illustrate the inductive nature of two basic data structures and show how this leads to the elegant technique of recursive programming.

1. Lists

A `List` (to simplify matters, we assume that we store only integeres as data elements) is a sequence of 0 or more integers. The idea of a sequence or, more generally, of a linked structure is captured by pointers between objects storing data. Thus, one would define objects of the form

```
List
 int x;
 List next;
```

so that, for instance, the list $\langle 3, 7, 2, 5 \rangle$ would contain 4 `List` objects, plus the additional `null` object at the end:



The declaration of the `List` objects tells us that a `List` is:

(1) either a `null` object (the default value for pointers)
(2) or an integer (stored in the current `List` object) followed by another `List` object.

But this is exactly an inductive definition, namely, the one from example 2.16, the only difference being that of the language used.

1a. From inductive definition to recursion

The above is also a 1-1 definition and thus gives rise to natural recursive programming over lists. The idea of recursive programming is to traverse a structure in the order opposite to the way we imagine it built along its inductive definition. We start at some point of the structure and proceed to its subparts until we reach the basis case. For instance, the function computing length of a list is programmed recursively to the left:

```
int length(List L)                int sum(List L)
  IF (L is null) return 0;          IF (L is null) return 0;
  ELSE return 1+length(L.next);     ELSE return L.x+sum(L.next);
```

It should be easy to see that the pseudo-code on the left is nothing more than the inductive definition of the function from example 2.24. Instead of the mathematical formulation used there, it uses the operational language of programming to specify:

1. the value of the function in the basis case (which also terminates the recursion) and then
2. the way to compute the value in the non-basis case from the value for some subcase which brings recursion "closer to" the basis.

The same schema is applied in the function to the right which computes the sum of all integers in the list.

Notice that, abstractly, `List`s can be viewed simply as finite strings. You may rewrite the definition of concatenation from Example 2.25 for `List`s as represented here.

1b. Equality of Lists

Inductive 1-1 definition of a set (here of the set of `List` objects given by their declaration) gives also rise to the obvious recursive function for comparing objects for equality. Two lists are equal iff

i) they have the same structure (i.e., the same number of elements) and

ii) respective elements in both lists are equal

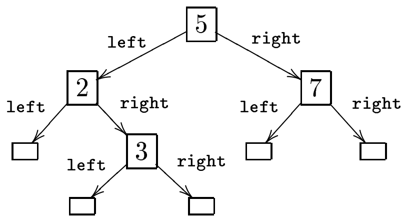
The corresponding pseudo-code for recursive function definition is as follows. The first two lines check point i) and ensure termination upon reaching the basis case. The third line checks point ii). If everything is ok so far, the recursion proceeds to check the rest of the lists:

```
boolean equal(List L, R)
  IF (L is null AND R is null) return TRUE;
  ELSE IF (L is null OR R is null) return FALSE;
  ELSE IF (L.x ≠ R.x) return FALSE;
  ELSE return equal(L.next, R.next);
```

2. Trees

Another very common data structure is binary tree `BT`. (Again, we simplify presentation by assuming that we only store integers as data.) Unlike in a list, each node (except the `null` ones) has two successors (called "children") left and right:

```
BT
  int x;
  BT left;
  BT right;
```



The inductive definition says that a binary tree `BT` is

(1) either a `null` object
(2) or an integer (stored in the current `BT` object) with two pointers (`left` and `right`) to other (always distinct) `BT` objects.

2a. From inductive definition to recursion

To compute the sum of integers stored in a given tree, we have to compute the sums stored in its left and right subtrees and add to them the value stored at the current node itself. The recursive function reflecting the inductive definition is as follows:

```
int sum(BT T)
  IF (T is null) return 0;
  ELSE return (sum(T.left) + T.x + sum(T.right));
```

Again, the first line detects the basis case, while the second one computes the non-basic case, descending recursively down the tree structure.

2b. Equality of Binary Trees

Using the *operational* intuition of inductive generation might suggest that the definition of binary tree is not 1-1. To generate the tree from the example drawing, we might first generate its left subtree and then the right one, or else other way around. The sequence of steps leading to the construction of the whole tree would not be unique.

However, this operational intuition, relaying on the *temporal ordering* of construction steps is not what matters for a definition to be 1-1. There is *only one logical* way to obtain this tree: we must have *both* its left *and* its right subtree, and *only then* we can make this tree. That is, there are *unique* elements (left and right subtree, and the integer to be stored in the root node itself) and the *unique* rule to be applied (put left subtree to the left, right to the right, and the integer in the node). The definition is 1-1.

Equality of binary trees follows naturally: two trees are equal iff

i) they have the same structure and
ii) respective elements stored at respective nodes in both trees are equal

Having the same structure amounts here to the following condition: trees $T1$ and $T2$ have the same structure iff:

- both are `null` or
- $T1 = (L1, x1, R1)$, $T2 = (L2, x2, R2)$, i.e., neither is `null`, and *both* $L1, L2$ have the same structure *and* $R1, R2$ have the same structure.

This is clearly an inductive definition of 'having the same structure', giving us the recursive pseudo-code for checking equality of two binary trees:

```
boolean equal(BT T1,T2)
  IF (T1 is null AND T2 is null) return TRUE;
  ELSE IF (T1 is null OR T2 is null) return FALSE;
  ELSE IF (T1.x ≠ T2.x) return FALSE;
  ELSE return ( equal(T1.left, T2.left) AND
                                  equal(T1.right,T2.right) );
```

Ending now this programming excursion, we return to the proof strategy arising from inductive definitions of sets which will be of crucial importance in later chapters.

2.3: Proofs by Structural Induction

Since, according to Idea 2.20, an inductive definition of a set induces a well-founded ordering, it allows us to perform inductive proofs of the properties of this set. This is called proof by *structural induction* – the word "structural" indicating that the proof, so to speak, proceeds along the structure of the set imposed by its inductive definition. In contrast to the definitions of functions on inductive sets 2.23, this proof strategy works for all inductively defined sets, regardless of whether the definitions are 1-1.

Proof by structural induction is just a special case of the inductive proof Idea 2.8. Simply, because any inductive definition of a set induces a well-founded ordering on this set according to the Idea 2.20. Using this ordering leads to the following proof strategy:

Idea 2.28 [**Proof by Structural Induction**] Suppose that, given a set S defined inductively from basis B, we want to prove that each element $x \in S$ has the property P – that $P(x)$ holds for all $x \in S$. Proceed as follows:

BASIS :: Show that $P(x)$ holds for all $x \in B$.
IND. :: For each way an $x \in S$ can be constructed from one or more $y_1, \ldots, y_n \in S$, show that the induction hypothesis : $P(y_1), \ldots, P(y_n)$ implies $P(x)$.
CLSR. :: If you managed to show this, then the closure property of S allows you to conclude that $P(x)$ holds for all $x \in S$.

It is straightforward to infer the proof rule in idea 2.28 from the one in idea 2.8. We assume that 2.8 holds. Then, assuming BASIS and INDUCTION step of 2.28, we merely has to prove the INDUCTION step in idea 2.8. So let x be an arbitrary member of S and assume the IH that $P(y)$ holds for all $y \prec x$. There are two possible cases: Either $f(x) = 0$, in which case $x \in S_0 = B$, and $P(x)$ follows from the BASIS part of idea 2.28. Otherwise $x \in S_{i+1} \setminus S_i$ for some $i \in \mathbb{N}$. Then x can be obtained from some $y_1, \ldots, y_n \in S_i$. Since these are all less than x in the sense of $\prec$, by the IH $P(y_1)$ and ... and $P(y_n)$. But then $P(x)$ follows from the INDUCTION part of idea 2.28, and the argument is complete.

The great advantage of inductively defined sets is that, in order to prove their properties, one need not inquire into the details of the induced well-founded ordering but merely has to follow the steps of the definition (as described in idea 2.28).

Example 2.29
The set of natural numbers was defined inductively in example 2.18. The induced well-founded ordering will say that $s^n0 \prec s^m0$ iff $n < m$. Thus, writing the natural numbers in the usual way $0, 1, 2, 3, \ldots$, and replacing sn by $n + 1$, the induced ordering is the standard ordering $<$ – the structural

induction on this set will be the usual mathematical induction. That is:

BASIS :: Show that $P(0)$ holds.
IND. :: Assuming the induction hypothesis $P(n)$, show that it implies also $P(n+1)$.

It is the most common form of induction used in mathematics. The proof in example 2.10 used this form, observed there as a "weaker induction hypothesis" (than the one allowed by the general formulation from idea 2.8). Using structural induction, we show that for all $n \in \mathbb{N}$:

$$1 + 2 + ... + n = \frac{n(n+1)}{2} .$$

BASIS :: $n = 0 : 0 = \dfrac{0(0+1)}{2} = 0$

IND. :: Assume the IH $1 + 2 + ... + n = \dfrac{n(n+1)}{2}$. Then

$$1 + 2 + ... + n + (n+1) = \frac{n(n+1)}{2} + (n+1) = \frac{(n+1)(n+2)}{2}$$
. □

The proof rule for natural numbers used above can be written more succinctly as follows.

$$\frac{P(0) \ \& \ \forall x(P(x) \rightarrow P(x+1))}{\forall x P(x)} \tag{2.30}$$

This rule can be sometimes more cumbersome to use than the rule (2.11). To see this, try to use it to prove the prime number theorem which was shown in example 2.9 using the general induction schema for natural numbers (i.e., the rule (2.11)). The difference between the two concerns, actually, only the "easiness" of carrying out the proof – as you are asked to show in exercise 2.9, the two proof rules have the same power.

Example 2.31
We show that the concatenation function from Example 2.25 is associative, i.e., that for any strings $s \cdot (t \cdot p) = (s \cdot t) \cdot p$. We proceed by structural induction on the first argument:

BASIS :: $\epsilon \cdot (t \cdot p) = t \cdot p = (\epsilon \cdot t) \cdot p$
IND. :: $xs \cdot (t \cdot p) = x(s \cdot (t \cdot p)) \stackrel{\text{IH}}{=} x((s \cdot t) \cdot p) = (x(s \cdot t)) \cdot p = (xs \cdot t) \cdot p$ □

Example 2.32
Define the set U inductively:

BASIS :: $\varnothing \in U$
IND. :: if $S \in U$ then $\wp(S) \in U$.

and show that for all $S \in U : S \notin S$. What is the ordering $\prec$ induced on the set U by this definition? Well, $\varnothing$ is the least element and then, for any set S we have that $S < \wp(S)$. $\prec$ is the transitive closure of this relation $<$. The nice thing about structural induction is that one actually need not have a full insight into the structure of the induced ordering but merely has to follow the inductive definition of the set.

BASIS :: Since, for all $X : X \notin \varnothing$ so, in particular, $\varnothing \notin \varnothing$.
IND. :: Assume IH : $S \notin S$. Contrapositively, assume that $\wp(S) \in \wp(S)$. This means that $\wp(S) \subseteq S$. On the other hand, $S \in \wp(S)$. But since $X \subseteq Y$ iff for all $x : x \in X \Rightarrow x \in Y$, we thus obtain that $S \in \wp(S)$ & $\wp(S) \subseteq S \Rightarrow S \in S$, contradiciting IH. □

Example 2.33
Show that the set L defined in Example 2.19.1 is actually the set $T = \{a^n b^n : n \in \mathbb{N}\}$.
1. We show first the inclusion $L \subseteq T$ by structural induction on L.

BASIS :: $\epsilon = a^0 b^0$.
IND. :: Assume that $s \in L$ satisfies the property, i.e., $s = a^n b^n$ for some $n \in \mathbb{N}$. The string produced from s by the rule will then be $asb = aa^n b^n b = a^{n+1} b^{n+1}$. Hence all the strings of L have the required form and $L \subseteq \{a^n b^n : n \in \mathbb{N}\}$.

Notice again that we did not ask about the precise nature of the induced ordering but merely followed the steps of the inductive definition of L in carrying out this proof. (The induced ordering $\prec$ on L is given by: $a^n b^n \prec a^m b^m$ iff $n < m$.)

2. On the other hand, any element of $\{a^n b^n : n \in \mathbb{N}\}$ can be generated in n steps according to the L's construction rule. This is shown by induction on n, that is, on the natural ordering $\langle \mathbb{N}, < \rangle$:

BASIS :: For $n = 0$, we have $a^0 b^0 = \epsilon \in L$.
IND. :: If IH : $a^n b^n \in L$, then $aa^n b^n b = a^{n+1} b^{n+1} \in L$ by the induction step of definition of L.

Hence $\{a^n b^n : n \in \mathbb{N}\} \subseteq L$. □

Example 2.34
Recall the language L defined in Example 2.19.2. Define the subset S of L :

$$\begin{array}{rl} \text{BASIS} :: & B = \{a, b, \neg c, (a \to (b \to c)), (c \to b)\} \subset S \\ \text{IND.} :: & \text{if } s \in S \text{ and } (s \to t) \in S \text{ then } t \in S \\ & \text{if } \neg t \in S \text{ and } (s \to t) \in S \text{ then } \neg s \in S \end{array}$$

We show that $S = B \cup \{(b \to c), \neg b, c\}$ constructing S step by step:

$S_0 =$ $B = \{a, b, \neg c, (a \to (b \to c)), (c \to b)\}$
$S_1 =$ $S_0 \cup \{(b \to c)\}$, taking a for s and $(b \to c)$ for t and applying the first rule
$= \{a, b, \neg c, (a \to (b \to c)), (c \to b), (b \to c)\}$
$S_2 =$ $S_1 \cup \{\neg b, c\}$ taking b for s and c for t and applying both rules
$= \{a, b, \neg c, (a \to (b \to c)), (c \to b), (b \to c), \neg b, c\}$
$S =$ Every application of a rule to some new combination of strings now yields a string which is already in S_2. Since no new strings can be produced, the process stops here and we obtain $S = S_2$. □

The last example will illustrate some more intricate points which often may appear in proofs by structural induction.

Example 2.35
Recall the definition of finite non-empty strings Σ^+ from Example 2.17. For the same alphabet Σ, define the set Σ' inductively as follows:

$$\begin{array}{rl} \text{BASIS} :: & x \in \Sigma' \text{ for all } x \in \Sigma \\ \text{IND-A} :: & \text{if } x \in \Sigma \text{ and } s \in \Sigma' \text{ then } xs \in \Sigma' \\ \text{IND-B} :: & \text{if } x \in \Sigma \text{ and } s \in \Sigma' \text{ then } sx \in \Sigma'. \end{array}$$

If we want to view this as a definition of finite non-empty strings, we have to first observe that the same string may be generated in various ways – this definition is not 1-1. For instance, abc may be obtained from bc by prepending a according to rule 1, or else from ab by appending c according to rule 2. To make sure that these operations yield the same result, we should augment the above definition with the equation:

$$x(sy) = (xs)y \tag{2.36}$$

Intuitively, the sets Σ^+ and Σ' are the same – we said that both are the set of finite non-empty string. But this is something we have to show. As is often the case, equality of two sets is shown by showing two inclusions.

$\Sigma^+ \subseteq \Sigma'$. This inclusion is trivial since anything which can be generated from the Basis and Induction rule in definition of Σ^+ can be generated by the same process following defnition of Σ'. (Strictly speaking, we show it by structural induction following the definition of Σ^+: every element of its basis is also in the basis of Σ'; and then, whatever can be obtained according to the rule from the definition of Σ^+ can be obtained by the first rule from the definition of Σ'.)

$\Sigma' \subseteq \Sigma^+$, i.e., for every $s : s \in \Sigma' \to s \in \Sigma^+$. We show the claim by structural induction on s, i.e., on the definition of Σ'.

Basis :: if $x \in \Sigma$, then $x \in \Sigma'$ and also $x \in \Sigma^+$.

IH :: Assume the claim for s, i.e., $s \in \Sigma' \to s \in \Sigma^+$. Notice that, according to Idea 2.28, we have to show now the claim *for each way* a "new" element may be obtained. The two different rules in the inductive definition of Σ', give rise to two cases to be considered.

Ind-a :: $xs \in \Sigma'$ – by IH, $s \in \Sigma^+$, and so by the induction step of the definition of $\Sigma^+ : xs \in \Sigma^+$.

Ind-b :: $sx \in \Sigma'$ ...? There does not seem to be any helpful information to show that then $sx \in \Sigma^+$...
Let us therefore try *a new level* of induction, now for showing this particular case. We have the assumption of IH above, and proceed by sub-induction, again on s:

Basis2 :: $s = y \in \Sigma$ and $x \in \Sigma$ – but then $xy \in \Sigma^+$, by the induction step of its definition

IH2 :: The Basis has been shown for arbitrary x and $y = s$, and so we may assume that: for every $x \in \Sigma$ and for $s \in \Sigma'$: $sx \in \Sigma^+$. We have again two cases for s:

Ind2-a :: $ys \in \Sigma'$, and $(ys)x \in \Sigma'$ – then, by (2.36), $(ys)x = y(sx)$ while by IH2, $sx \in \Sigma^+$. But this suffices to conclude that $y(sx) \in \Sigma^+$.

Ind2-b :: $sy \in \Sigma'$, and $(sy)x \in \Sigma'$ – by IH2 we have that $sy \in \Sigma^+$, but what can we then do with the whole term $(sy)x$?
Well, $sy \in \Sigma^+$ is very significant – by the definition of Σ^+, this means that sy can be actually written as zt for some $z \in \Sigma$ and $t \in \Sigma^+$, i.e., $sy = zt$. Can we now conclude that $(sy)x = (zt)x \overset{(2.36)}{=} z(tx) \in \Sigma^+$? Not really, because for that we would need that $tx \in \Sigma^+$, and we do not know that. We might, perhaps, try yet another level of induction – now on t – but this should start looking suspicious.

What we know about tx is that i) $tx \in \Sigma'$ (since $zt \in \Sigma'$ so $t \in \Sigma'$) and that ii) $length(tx) = length(sy)$. If IH2 could be assumed for all such tx (and not only for the particular sy from which the actual $(sy)x$ is built), we would be done. And, as a matter of fact, we *are allowed to assume just that* in our (sub-)induction hypothesis IH2. Why? Because proving our induction step for the term of the form $(sy)x$ (or $(ys)x$) we can assume the claim for *all terms which are smaller* in the actual ordering. This is the same as the strong version of the induction principle, like the one used in example 2.9, as compared to the weaker version, like the one used in Example 2.10. The actual formulation of Idea 2.28 allows us only the weaker version of induction hypothesis – namely, the assumption of the claim only for the *immediate* predecessors of the element for which we are proving the claim in the induction step. However, if we inspect the actual ordering induced by the inductive definition according to Idea 2.20 and apply the general theorem 2.7, then we see that also this stronger version will be correct. □

We will encounter many examples of inductive proofs in the rest of the course. In fact, the most important sets we will consider will be defined inductively and most proofs of their properties will use structural induction.

3: TRANSFINITE INDUCTION ______________________ [optional]

As a final example, we give an inductive definition of ordinals (introduced by von Neumann and called also "ordinal numbers" though, as a matter of fact, they are sets), and show inductively some properties of this set. The example shows that induction is by no means restricted to finitely generated elements. It can be carried over to sets of arbitrary cardinality. Although the principle is the same as before, emphasising the context, one calls it "transfinite induction".

Define the collection of von Neumann's ordinals, $\mathbb{O}$, inductively as follows:

BASIS :: $\varnothing \in \mathbb{O}$

IND. :: 1) if $x \in \mathbb{O}$ then $x^+ = x \cup \{x\} \in \mathbb{O}$ – "successor" of an ordinal is an ordinal

2) if $x_i \in \mathbb{O}$ then $\bigcup x_i \in \mathbb{O}$ – arbitrary (possibly infinite) union of ordinals is an ordinal

We show a few simple facts about $\mathbb{O}$ using structural induction.

(A) For all $x \in \mathbb{O} : y \in x \Rightarrow y \subset x$.

BASIS :: $x = \varnothing$, and as there is no $y \in \varnothing$, the claim follows trivially.

IND :: 1) From IH : $y \in x \Rightarrow y \subset x$ show that $y \in x^+ \Rightarrow y \subset x^+$. $y \in x^+$ means that either a) $y \in x$, from which, by IH, we get $y \subset x$ and thus $y \subset x \cup \{x\}$, or b) $y = x$, and then $y \subseteq x$ but $y \neq x \cup \{x\}$, i.e., $y \subset x \cup \{x\}$.

2) IH : for all $x_i : y \in x_i \Rightarrow y \subset x_i$. If $y \in \bigcup x_i$, then there is some $k : y \in x_k$ and, by IH, $y \subset x_k$. But then $y \subset \bigcup x_i$.

(B) For all $x \in \mathbb{O} : y \in x \Rightarrow y \in \mathbb{O}$ (each element of an ordinal is an ordinal)

BASIS :: Since there is no $y \in \varnothing$, this follows trivially.
IND :: Let $x \in \mathbb{O}$ and 1) assume IH : $y \in x \Rightarrow y \in \mathbb{O}$. If $y \in x \cup \{x\}$ then either $y \in x$ and by IH $y \in \mathbb{O}$, or else $y = x$ but $x \in \mathbb{O}$.
2) IH : for all $x_i : y \in x_i \Rightarrow y \in \mathbb{O}$. If $y \in \bigcup x_i$, then there is some x_k for which $y \in x_k$. Then, by IH, $y \in \mathbb{O}$.

The second rule in the definition of $\mathbb{O}$ enables one to construct elements of $\mathbb{O}$ without any immediate predecessor. In the above proofs, the case 1) was treated by a simple induction assuming the claim about the immediate predecessor. The case 2), however, required the stronger version assuming the claim for all ordinals x_i involved in forming the new ordinal by union. This step amounts to transfinite induction – what we have proven holds for the set $\mathbb{O}$ whose small, initial segment is shown below, with the more standard notation indicated in the right column.

$$\begin{array}{rl}
\omega+\omega\left\{\begin{array}{l}\omega\left\{\begin{array}{l}\varnothing\\ \{\varnothing\}\\ \{\varnothing,\{\varnothing\}\}\\ \{\varnothing,\{\varnothing\},\{\varnothing,\{\varnothing\}\}\}\\ \vdots\end{array}\right.\\ \omega\\ \{\omega,\{\omega\}\}\\ \{\omega,\{\omega\},\{\omega,\{\omega\}\}\}\\ \vdots\end{array}\right. & \begin{array}{l}0\\ 1\\ 2\\ 3\\ \vdots\\ \omega\\ \omega+1\\ \omega+2\\ \vdots\end{array}\\
\begin{array}{l}2\omega\\ \{2\omega,\{2\omega\}\}\\ \vdots\end{array} & \begin{array}{l}2\omega\\ 2\omega+1\\ \vdots\end{array}
\end{array}$$

ω is the first such *limit* ordinal (with no immediate predecessor), $\omega+\omega = 2\omega$ the second, etc. – the sequence of ordinals continues indefinitely:

$$\begin{array}{l}
0, 1, 2, 3, 4, 5, \ldots \omega, \omega+1, \omega+2, \ldots \omega+\omega = \\
\quad 2\omega, 2\omega+1, 2\omega+2, \ldots 3\omega, 3\omega+1, \ldots 4\omega, 4\omega+1, \ldots \omega * \omega = \\
\quad \omega^2, \omega^2+1, \ldots \omega^3, \omega^3+1, \ldots \omega^4, \omega^4+1, \ldots \\
\quad \omega^\omega, \omega^\omega+1, \ldots \omega^{2\omega}, \ldots \omega^{3\omega}, \ldots \omega^{\omega*\omega}, \ldots \omega^{(\omega^3)}, \ldots \omega^{(\omega^\omega)}, \ldots
\end{array}$$

Ordinals play the central role in set theory, providing the paradigm for well-orderings (total well-founded orderings). Notice that this means that we may have several ordinals of the same cardinality (the concept of ordinal number includes ordering, that of a cardinal number does not). For instance, the ordinals

ω, $\omega+1$ and $1+\omega$, can be easily seen to have the same cardinality:

$$\begin{array}{rl} \omega = & 0 < 1 < 2 < 3 < 4 < 5 < ... \\ \omega + 1 = & 0 < 1 < 2 < 3 < 4 < 5 < ... < \omega \\ 1+\omega = & \bullet < 0 < 1 < 2 < 3 < 4 < 5 < ... \end{array}$$

The functions $f : \omega+1 \to 1+\omega$ defined by $f(\omega) = \bullet$ and $f(n) = n$ and $g : 1+\omega \to \omega$ defined by $g(\bullet) = 0$ and $g(n) = n+1$ are obviously bijections, i.e., set-isomorphisms. g is, in addition order-isomorphism, since it satisfies for all elements $x, y \in 1+\omega : x < y \Rightarrow g(x) < g(y)$. This means that these two ordinals represent essentially the same ordering. However, f does not preserve the ordering, since it maps the greatest element ω of the ordering $\omega+1$ onto the smallest element $\bullet$ of the ordering $1+\omega$. These two ordinals, although of the same cardinality ω, represent different ways of ordering all ω elements. The first makes first an infinite sequence and then adds a maximal element at the end of it. The second makes an infinite sequence and adds a new minimal element in front of it. Thus the former does possess a maximal element while the latter does not.

We thus see that $\omega+1 \neq 1+\omega$ – the inequality which holds in the ordinal arithmetics exactly because the ordinals represent not only numbers (or unstructured sets) but orderings...[end optional]

Exercises 2.

EXERCISE 2.1 Given the following inductively defined set $S \subset \mathbb{N} \times \mathbb{N}$:

BASIS :: $\langle 0, 0 \rangle \in S$
IND. :: If $\langle n, m \rangle \in S$ then $\langle s(n), m \rangle \in S$ and $\langle s(n), s(m) \rangle \in S$

Determine the property P which allows you to describe the set S as a set of the form
$\{\langle n, m \rangle : P(n, m)\}$.
Describe those elements of S which can be derived in more than one way.

EXERCISE 2.2 Let $\Sigma = \{a, b, c\}$, $\Gamma = \{\neg, \to, (,)\}$ and define the language WFF^{Σ} over $\Sigma \cup \Gamma$ inductively as follows:

BASIS :: If $A \in \Sigma$ then $A \in \mathsf{WFF}^{\Sigma}$
IND. :: If $A, B \in \mathsf{WFF}^{\Sigma}$ then $\neg A \in \mathsf{WFF}^{\Sigma}$ and $(A \to B) \in \mathsf{WFF}^{\Sigma}$.

(1) Which of the following strings belong to WFF^{Σ} :
$(a \to \neg b) \to c,\ a \to b \to c,\ \neg(a \to b),\ (\neg a \to b),\ \neg a \to b$?
(2) Replace Σ with $\Delta = \{*, \#\}$ and use the analogous definition of WFF^{Δ}. Which strings are in WFF^{Δ} :
$*\#,\ \neg *,\ \neg(\#\#),\ (* \to \#),\ * \to \#,\ * \leftarrow \#$?

EXERCISE 2.3 Describe the ordering induced on the Σ^+ of example 2.22 by the definitions of idea 2.20.

EXERCISE 2.4 Use induction on $\mathbb{N}$ to prove the following equations for all $1 \leq n \in \mathbb{N}$

(1) $1 + 2 + ... + n = \dfrac{n(n+1)}{2}$

(2) $(1 + 2 + ... + n)^2 = 1^3 + 2^3 + ... + n^3$
(In the induction step expand the expression $((1+2+...+n)+(n+1))^2$ and use 1.)

EXERCISE 2.5 Use induction to prove that a finite set with n elements has 2^n subsets (cf. Exercise 1.9).

EXERCISE 2.6 Let X be a country with finitely many cities. Show that for any such X, if every two cities in X have (at least) one one-way road between them, then there is some starting city x_0 and a route from x_0 which passes through every city exactly once.

EXERCISE 2.7 Using the definition of strings Σ^* from example 2.16 as a schema for structural induction, show that the reverse operation from example 2.26 is idempotent, i.e., $(s^R)^R = s$.
(Show first, by structural induction on s, the lemma: for all $y \in \Sigma, s \in \Sigma^* : (s \vdash y)^R = y(s^R)$.)

EXERCISE 2.8 Show that the operation $+$ on $\mathbb{N}$, defined in example 2.27, is commutative, i.e., that the identity $n + m = m + n$ holds for all $n, m \in \mathbb{N}$. This requires a *nested* induction, i.e.,

- the basis: for all $m \in \mathbb{N}$, $0 + m = m + 0$, and
- the induction: for all $m \in \mathbb{N}$, $s(n) + m = m + s(n)$, given that for all $m \in \mathbb{N}$, $n + m = m + n$

can themselves be proved only by the use of induction (on m).

EXERCISE 2.9 We have seen two different proof rules for induction on natural numbers – one (2.11) in example 2.9 and another (2.30) in example 2.29. Show that each can be derived from the other. Begin by stating clearly what exactly you are asked to prove!
(One part of this proof follows trivially from the general argument after Idea 2.28.)

Chapter 3
Turing Machines

- Alphabets, Strings, Languages
- Turing Machiness as accepting vs. computing devices
- A Universal Turing Machine
- Decidability

1: Alphabets and Languages

◇——————a Background Story ——————◇

The languages we use for daily communication are what we call *natural languages*. They are acquired in childhood and are suited to just about any communication we may have. They have developed along with the development of mankind and form a background for any culture and civilisation. *Formal languages*, on the other hand, are explicitly designed by people for a clear, particular purpose. A semi-formal language was used in the preceding chapters for the purpose of talking about sets, functions, relations, etc. It was only *semi*-formal because it was not fully defined. It was introduced along as we needed some notation for particular concepts.

Formal language is a fundament of formal logical system and we will later encouter several examples of formal languages. Its most striking feature is that, although designed for some purposes, it is an entity on its own. It is completely specified without necessarily referring to its possible meaning. It is a pure syntax. Similar distinction can be drawn with respect to natural languages. The syntax of a natural language is captured by the intuition of the *grammatically correct expressions*. `Quadrille drinks procrastination` is a grammatically correct expression consisting of the subject `quadrille`, verb in the proper form `drinks`, and object `procrastination`. As you can see, the fact that it is grammatical, does not ensure that it is meaningful. The sentence does convey an idea of some strange event which, unless it is employed in a very particular context, does not make any sense.

The pure syntax does not guarantee meaning. Yet, on the positive side, syntax is much easier to define and control than its intended meaning. At the even more basic level, and with respect to the written form, the basic building block of a language's syntax is the alphabet. We have the Latin alphabet `a,b,c...` and words are built from these letters. Yet, as with the meaningless grammatical sentences, not all possible combinations of the letters form valid words; `aabez` is perhaps a syntactic possibility but it is not a correct expression – there is no such word.

We will now start using such purely syntactic languanges. Their basis is determined by some, arbitrarily chosen alphabet. Then, one may design various syntactic rules determining which expressions built from the alphabet's symbols, form valid, well-formed words and sentences. In this chapter, we will merely introduce the fundamental definition of a language and observe how the formal notion of computability relates necessarily to some formal language. In the subsequent chapters, we will study some particular formal languages forming the basis of most common logical systems.

◇——————————————————————◇

Definition 3.1 An *alphabet* is a (typically finite) set, its members are called *symbols*. The set of all finite strings over an alphabet Σ is denoted Σ^*. A *language* L over an alphabet Σ is a subset L$\subseteq \Sigma^*$.

Example 3.2

The language with natural numbers as the only expressions is a subset of $\mathsf{N} \subseteq \Sigma^*$ where $\Sigma = \{0,1,2,3,4,5,6,7,8,9\}$.

Typically, languages are defined inductively. The language L of natural number arithmetic can be defined over the alphabet $\Sigma' = \Sigma \cup \{), (, +, -, *\}$

Basis :: If $n \in \mathsf{N}$ then $n \in \mathsf{L}$

Ind. :: If $m, n \in \mathsf{L}$ then $(m+n)$, $(m-n)$, $(m*n)$ $\in \mathsf{L}$. □

Alphabets of particular importance are the ones with only two symbols. Any finite alphabet Σ with n distinct symbols can be encoded using an alphabet with only two symbols, e.g., $\Delta = \{0,1\}$

Example 3.3

To code any 2-symbol alphabet Σ, we just choose any bijection $\Sigma \leftrightarrow \Delta$. To code $\Sigma = \{a,b,c\}$ we may choose the representation $\{a \mapsto 00, b \mapsto 01, c \mapsto 10\}$.

The string *aacb* will be represented as 00001001. Notice that to decode this string, we have to know that any symbol from Σ is represented by a string of exactly two symbols from Δ. $\Box$

In general, to code an alphabet with n symbols, one has to use strings from Δ of length $\lceil log_2 n \rceil$. Coding a 5-symbol alphabet, will require strings of length at least 3.

Binary representation of natural numbers is a more specific coding using not only the difference between symbols but also positions at which they occur.

Definition 3.4 A binary number b is an element of Σ^* where $\Sigma = \{0, 1\}$. A binary number $b = b_n b_{n-1} ... b_2 b_1 b_0$ represents the natural number $b_n * 2^n + b_{n-1} * 2^{n-1} + b_{n-2} * 2^{n-2} + ... + b_1 * 2^1 + b_0 * 2^0$.

For instance, 0001 and 01 both represent 1; 1101 represents $1 * 2^3 + 1 * 2^2 + 0 * 2 + 1 * 1 = 13$.

2: Turing Machines

◇—— a Background Story ——◇

Turing Machine (after English mathematician Alan Turing, 1912-1954) was the first general model designed for the purpose of separating problems which can be solved automatically from those which cannot. Although the model was purely mathematical, it was easy to imagine that a corresponding physical device could be constructed. In fact, it was and is today known as the computer.

Many other models of computability have been designed but, as it turns out, all such models define the same concept of computability, i.e., the same problems are mechanically computable irrespectively of the definition of computability. The fundamental results about Turing machines apply to all computations on even most powerful computers. The limits of Turing computability are also the limits of the modern computers.

The rest of the story below is taken from Turing's seminal paper "On computable numbers, with an application to the Entscheidungsproblem" from 1936:

"Computing is normally done by writing certain symbols on paper. We may suppose this paper is divided into squares like a child's arithmetic book. In elementary arithmetic the two-dimensional character

of the paper is sometimes used. But such a use is always avoidable, and I think that it will be agreed that the two-dimenstional character of paper is no essential of computation. I assume then that the computation is carried out one one-dimensional paper, i.e., on a tape divided into squares. I shall also suppose that the number of symbols which may be printed is finite. If we were to allow an infinitey of symbols, then there would be symbols differing to an arbitrary small extent. The effect of this restriction of the number of symbols is not very serious. It is always possible to use sequences of symbols in the place of single symbols. Thus an Arabic numeral such as 17 or 999999999999999 is normally treated as a single symbol. Similarly, in any European language words are treated as single symbols. (Chinese, however, attempts to have an enumerable infinity of symbols). The differences from our point of view between the single and compound symbols is that the compound symbols, if they are too lengthy, cannot be observed at one glance. This is in accordance with experience. We cannot tell at a glance whether 9999999999999999 and 999999999999999 are the same.

The behaviour of the computer at any moment is determined by the symbols which he is observing, and his "state of mind" at that moment. We may suppose that there is a bound B to the number of symbols of squares which the computer can observe at one moment. If he wishes to observe more, he must use successive observations. We will also suppose that the number of states of mind which need be taken into account is finite. The reasons for this are of the same character as those which restrict the number of symbols. If we admitted an infinity of states of mind, some of them will be "arbitrarily close" and will be confused. Again, the restriction is not one which seriously affects computation, since the use of more complicated states of mind can be avoided by writing more symbols on the tape.

Let us imagine the operations performed by the computer to be split up into "simple operations" which are so elementary that it is not easy to imagine them further divided. Every such operation consists of some change of the physical system consisting of the computer and his tape. We know the state of the system if we know the sequence of symbols on the tape, which of these are observed by the computer (possibly with a special order), and the state of mind of the computer. We may suppose that in a simple operation not more than onse symbol is altered. Any other change can be split up into simple changes of this kind. The situation in regard to the squares whose symbols may be altered in this way is the same as in regard to the observed squares. We may, therefore, without loss of generality, assume that the squares whose symbols are changed are always "observed" squares.

Besides these changes of symbols, the simple operations must include changes of distribution of observed squares. The new squares must be immediately recognisable by the computer. I think it is reasonable to suppose that they can only be squares whose distance from the closest of the immediately previously observed squares does not exceed a certain fixed amount. Let us say that each of the new observed squares is within L squares of an immediately previously observed square.

In connection with "immediate recognisability", it may be thought that there are other kinds of square which are immediately recognisable. In particular, squares marked by special symbols might be taken as imemdiately recognisable. Now if these squares are marked only by single symbols there can be only finite number of them, and we should not upset our theory by adjoining these marked squares to the observed squares. If, on the other hand, they are marked by a sequence of symbols, we cannot regard the process of recognition as a simple process. This is a fundamental point and should be illustrated. In most mathematical papers the equations and theorems are numbered. Normally the numbers do not go beyond (say) 1000. It is, therefore, possible to recognise a theorem at a glance by its number. But if the paper was very long, we might reach Theorem 157767733443477; then, further on in the paper, we might find "...hence (applying Theorem 157767733443477) we have...". In order to make sure which was the relevant theorem we should have to compare the two numbers figure by figure, possibly ticking the figures off in pencil to make sure of their not being counted twice. If in spite of this it is still thought that there are other "immediately recognisable" squares, it does not upset my contention so long as these squares can be found by some process of which my type of machine is capable. [...]

The simple operations must therefor include:

(a) Changes of the symbol on one of the observed squares.
(b) Changes of one of the squares observed to another square within L squares of one of the previously observed squares.

It may be that some of these changes necessarily involve a change of state of mind. The most general single operation must therefore be taken to be one of the following:

(A) A possible change (a) of symbol together with a possible change of state of mind.
(B) A possible change (b) of observed squares, together with a possible change of mind.

The operation actually performed is determined, as has been suggested, by the state of mind of the computer and the observed symbols. In particular, they determine the state of mind of the computer after the operation is carried out.

We may now construct a machine to do the work of this computer. To each state of mind of the computer corresponds an "m-configuration" of the machine. The machine scans B squares corresponding to the B squares observed by the computer. In any move the machine can change a symbol on a scanned square or can change any one of the scanned squares to another square distant not more than L squares from one of the other scanned squares. The move which is done, and the succeeding configuration, are determined by the scanned symbol and the m-configuration."

◊——◊

Definition 3.5 A *Turing machine*, M is a quadruple $\langle K, \Sigma, q_0, \tau \rangle$ where

- K is a finite set of *states* of M
- Σ is a finite alphabet of M
- $q_0 \in K$ is the *initial state*
- $\tau : K \times \Sigma \to K \times (\Sigma \cup \{\mathsf{L}, \mathsf{R}\})$ is a (partial) *transition function* of M.

For convenience, we will assume that Σ always contains the 'space' symbol $\#$. This definition deviates only slightly from the one sketched by Turing and does not deviate from it at all with respect to the computational power of the respective machines. The difference is that our machine reads only a single symbol (a single square, or position) at a time, $B = 1$, and that it moves at most one square at the time, $L = 1$.

Idea 3.6 [**Operation of TM**] Imagining M as a physical device with a "reading head" moving along an infinite "tape" divided into discrete positions, the transition function τ determines its operation as follows:

- M starts in the initial state q_0, with "its head" at some position on the input tape.

- When M is in a state q and "reads" a symbol a, and the pair $\langle q, a\rangle$ is in the domain of τ, then M "performs the action" $\langle q', a'\rangle = \tau(\langle q, a\rangle)$: "passes to the state" q' "doing" a' which may be one of the two things:
  - if $a' \in \Sigma$, M "prints" the symbol a' at the current position on the tape;
  - otherwise $a' = \mathsf{L}$ or $a' = \mathsf{R}$ – then M "moves its head" one step to the left or right, respectively.
- When M is in a state q and reads a symbol a, and the pair $\langle q, a\rangle$ is not in the domain of τ then M "stops its execution" – it *halts*.
- To "run M on the input string w" is to write w on an otherwise blank tape, place the reading head on the first symbol of w (if w is the empty string, place the reading head on any position) and then start the machine (from state q_0).
- $M(w)$ denotes the tape's content after M has run on the input w.

τ may be written as a set of quadruples $\{\langle q_1, a_1, q_1', a_1'\rangle, \langle q_2, a_2, q_2', a_2'\rangle, ...\}$ called *instructions*. We will often write a single instruction as $\langle q, a\rangle \mapsto \langle q', a'\rangle$. Notice that initially the tape contains the "input" for the computation. However, it is also used by M for writing the "output".

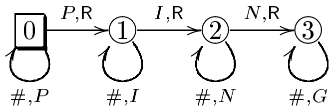
Remark.
Sometimes, one allows τ to be a relation which is not a function. This leads to *nondeterministic* Turing machines. However, such machines are not more powerful than the machines from our definition, and we will not study them.
Another variant, which does not increase the power and will not be discussed here either, allows TM to use several tapes. For instance, a machine with 2 tapes would have $\tau : K \times \Sigma \times \Sigma \to K \times (\Sigma \cup \{\mathsf{L}, \mathsf{R}\}) \times (\Sigma \cup \{\mathsf{L}, \mathsf{R}\})$.

Turing machines embody the idea of *mechanically computable functions* or *algorithms*. The following three examples illustrate different flavours of such computations. The one in Example 3.7 disregards its input (for simplicity we made the input blank) and merely produces a constant value – it *computes a constant function*. The one in Example 3.8 does not modify the input but *recognizes* whether it belongs to a specific language. Starting on the leftmost 1 it halts if and only if the number of consecutive 1's is even – in this case we say that it *accepts the input string*. If the number is odd the machine goes forever. The machine in Example 3.9 *computes a function* of the input x. Using unary representation of natural numbers, it returns $\lceil x/2 \rceil$ – the least natural number greater than or equal to $x/2$.

Example 3.7
The following machine goes PING! Starting on an empty (filled with blanks) tape, it writes "PING" and halts. The alphabet is $\{\#, P, I, N, G\}$,

we have 7 states $q_0 ... q_6$ with the initial state q_0, and the transition function is as follows (graphically, on the right, state q_x is marked by ⓧ and the initial state q_z by $\boxed{z}$):

$$\begin{array}{ll} \langle q_0, \# \rangle \mapsto \langle q_0, P \rangle & \langle q_0, P \rangle \mapsto \langle q_1, \mathsf{R} \rangle \\ \langle q_1, \# \rangle \mapsto \langle q_1, I \rangle & \langle q_1, I \rangle \mapsto \langle q_2, \mathsf{R} \rangle \\ \langle q_2, \# \rangle \mapsto \langle q_2, N \rangle & \langle q_2, N \rangle \mapsto \langle q_3, \mathsf{R} \rangle \\ \langle q_3, \# \rangle \mapsto \langle q_3, G \rangle & \end{array}$$

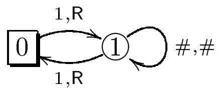


□

Example 3.8

The alphabet is $\{\#, 1\}$ and machine has 2 states. It starts on the left-most 1 in state q_0 – the transition function is given on the left, while the corresponding graphical representation on the right:

$$\begin{array}{r} \langle q_0, 1 \rangle \mapsto \langle q_1, \mathsf{R} \rangle \\ \langle q_1, 1 \rangle \mapsto \langle q_0, \mathsf{R} \rangle \\ \langle q_1, \# \rangle \mapsto \langle q_1, \# \rangle \end{array}$$

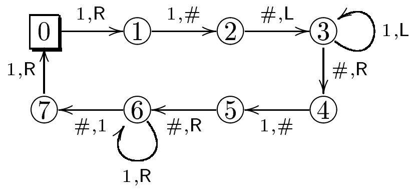


Write the computations of this machine on the inputs 1, 11 and 111. □

Example 3.9

The simplest idea to make a machine computing $\lceil x/2 \rceil$ – using the alphabet $\{\#, 1\}$ and unary representation of numbers – might be to go through the input removing every second 1 (replacing it with #) and then compact the result to obtain a contiguous sequence of 1's. We apply a different algorithm which keeps all 1's together all the time. Machine starts at the leftmost 1:

	#	1		Starting on the leftmost 1 in q_0
q_0		$\langle q_0, 1 \rangle \mapsto \langle q_1, \mathsf{R} \rangle$	–	jump over one 1; halt if no 1
q_1		$\langle q_1, 1 \rangle \mapsto \langle q_2, \# \rangle$	–	replace next 1 with #; halt if no 1
q_2	$\langle q_2, \# \rangle \mapsto \langle q_3, \mathsf{L} \rangle$		–	move left and
q_3	$\langle q_3, \# \rangle \mapsto \langle q_4, \mathsf{R} \rangle$	$\langle q_3, 1 \rangle \mapsto \langle q_3, \mathsf{L} \rangle$		return to the leftmost 1
q_4		$\langle q_4, 1 \rangle \mapsto \langle q_5, \# \rangle$	–	erase it
q_5	$\langle q_5, \# \rangle \mapsto \langle q_6, \mathsf{R} \rangle$		–	move right – return to # inserted
q_6	$\langle q_6, \# \rangle \mapsto \langle q_7, 1 \rangle$	$\langle q_6, 1 \rangle \mapsto \langle q_6, \mathsf{R} \rangle$		in q_1 replacing it with 1 erased in q_4
q_7		$\langle q_7, 1 \rangle \mapsto \langle q_0, \mathsf{R} \rangle$	–	move right and continue from q_0



Try to follow the computations on the input tapes #, 1, 11 and 111. □

2.1: Composing Turing machines .... [optional]
The following machine starts its computation anywhere within or just after a string of 1's and adds one 1 at the end of the string, M_{+1} : $\overset{1,R}{\circlearrowleft}\boxed{0} \xrightarrow{\#,1} 1$. Now taking, for instance the machine $M_{/2}$ from example 3.9, we should be able to put the two together into a machine M_f computing the function $f(x) = \lceil x/2 \rceil + 1$, by runing first $M_{/2}$ and then M_{+1}.

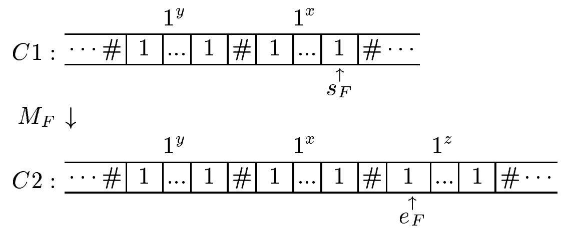
To do this in general, one has to ensure that the configuration in which the first machine halts (if it does) is of the form assumed by the second machine when it starts. A closer look at the machnie $M_{/2}$ from 3.9 enables us to conclude that it halts with the reading head just after the rightmost 1. This is an acceptable configuration for starting M_{+1} and so we can write our machine M_f as $\boxed{M_{/2}} \xrightarrow{\#,\#} \boxed{M_{+1}}$. This abbreviated notation says that: M_f starts by running $M_{/2}$ from its initial state and then, *whenever* $M_{/2}$ halts *and* reads a blank #, it passes to the initial state of M_{+1} writing #, and then runs M_{+1}.

We give a more elaborate example illustrating the idea of composing Turing machines. A while-loop in a programming language is a command of the form `while B do F`, where `B` is a boolean function and `F` is a command which we will assume computes some function. Assuming that we have machines M_B and M_F, we will construct a machine computing a function $G(y)$ expressed by the following while-loop:

```
G(y) = {  x:=1; z:=y;
          while not B(x,z) do {
            x:=x+1;
            z:=F(x,y);}
          return x; }
```

I.e., $G(y)$ is the least $x > 0$ such that $B(x, F(x, y))$ is true. If no such x exists $G(y)$ is undefined.

We consider the alphabet $\Sigma = \{\#, 1, Y, N\}$ and functions over positive natural numbers (without 0) $\mathbb{N}$, with unary representation as strings of 1's. Let our given functions be $F : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ and $B : \mathbb{N} \times \mathbb{N} \to \{Y, N\}$, and the corresponding Turing machines, M_F, resp. M_B. More precisely, s_F is the initial state of M_F which, starting in a configuration of the form $C1$, halts iff $z = F(x, y)$ is defined in the final state e_F in the configuration of the form $C2$:



If, for some pair x, y, $F(x, y)$ is undefined, $M_F(y, x)$ may go forever.

B, on the other hand, is total and M_B always halts in its final state e_B, when started from a configuration of the form $C2$ and intial state s_B, yiedling a confguration of the form $C3$:

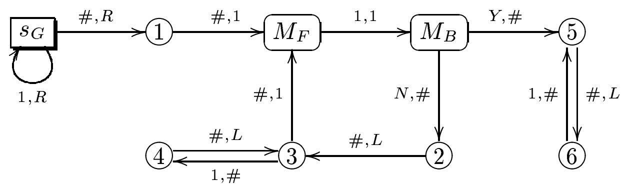
$$
\begin{array}{l}
C2: \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|c|}
\multicolumn{1}{c}{} & \multicolumn{3}{c}{1^y} & \multicolumn{1}{c}{} & \multicolumn{3}{c}{1^x} & \multicolumn{1}{c}{} & \multicolumn{3}{c}{1^z} & \multicolumn{1}{c}{} \\
\hline
\cdots\# & 1 & ... & 1 & \# & 1 & ... & 1 & \# & 1 & ... & 1 & \#\cdots \\
\hline
\multicolumn{9}{c}{} & \multicolumn{1}{c}{\overset{\uparrow}{s_B}} & \multicolumn{3}{c}{}
\end{array} \\
M_B \downarrow \\
C3: \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|c|c|}
\multicolumn{1}{c}{} & \multicolumn{3}{c}{1^y} & \multicolumn{1}{c}{} & \multicolumn{3}{c}{1^x} & \multicolumn{1}{c}{} & \multicolumn{3}{c}{1^z} & \multicolumn{2}{c}{} \\
\hline
\cdots\# & 1 & ... & 1 & \# & 1 & ... & 1 & \# & 1 & ... & 1 & u & \#\cdots \\
\hline
\multicolumn{12}{c}{} & \multicolumn{1}{c}{\overset{\uparrow}{e_B}} & \multicolumn{1}{c}{}
\end{array}
\end{array}
$$

where $u = Y$ iff $B(x, z) = Y$ (true) and $u = N$ iff $B(x, z) = N$ (false).

Using M_F and M_B, we design a TM M_G which starts in its initial state s_G in a configuration $C0$, and halts in its final state e_G iff $G(y) = x$ is defined, with the tape as shown in T:

$$
\begin{array}{l}
C0: \begin{array}{|c|c|c|c|c|}
\multicolumn{1}{c}{} & \multicolumn{3}{c}{1^y} & \multicolumn{1}{c}{} \\
\hline
\cdots\# & 1 & ... & 1 & \#\cdots \\
\hline
\multicolumn{3}{c}{} & \multicolumn{1}{c}{\overset{\uparrow}{s_G}} & \multicolumn{1}{c}{}
\end{array} \\
M_G \downarrow \\
T: \begin{array}{|c|c|c|c|c|c|c|c|c|}
\multicolumn{1}{c}{} & \multicolumn{3}{c}{1^y} & \multicolumn{1}{c}{} & \multicolumn{3}{c}{1^x} & \multicolumn{1}{c}{} \\
\hline
\cdots\# & 1 & ... & 1 & \# & 1 & ... & 1 & \#\cdots \\
\hline
\end{array}
\end{array}
$$

M_G will add a single 1 (= x) past the lefttmost # to the right of the input y, and run M_F and M_B on these two numbers. If M_B halts with Y, we only have to clean up $F(x, y)$. If M_B halts with N, M_G erases $F(x, y)$, extends x with a single 1 and continues:



In case of success, M_B exits along the Y and states 5-6 erase the sequence of 1's representing $F(x, y)$. M_G stops in state 6 to the right of x. If M_B got N, this N is erased and states 3-4 erase the current $F(x, y)$. The first blank # encountered in the state 3 is the blank right to the right of the last x. This # is replaced with 1 – increasing x to $x + 1$ – and M_F is started in a configuration of the form $C1$.

$M_G(y)$ will go forever if no x exists such that $B(x, F(x, y)) = Y$. However, it may also go forever if such x exists but $F(x', y)$ is undefined for some $0 < x' < x$! Then the function $G(y)$ computed by M_G is undefined. In the theory of recursive functions, such a schema is called μ-recursion ("μ" for minimal) – here it is the function $G : \mathbb{N} \to \mathbb{N}$

$$\begin{array}{rcl} G(y) & = & \text{the least } x \in \mathbb{N} \text{ such that } B(x, F(x,y)) = Y \\ & & \text{and } F(x',y) \text{ is defined for all } x' \leq x. \end{array}$$

The fact that if $G(y) = x$ (i.e., when it is defined) then also $F(x', y)$ must be defined for all $x' \leq x$, captures the idea of mechanic computability – M_G simply checks *all* consequtive values of x' until the correct one is found... [end optional]

2.2: Alternative representation of TMs .... [optional]
We give an alternative, equivalent, representation of an arbitrary TM defining it directly by a set of transitions between *situations* rather than of "more abstract" instructions.

The definition 3.5 of a TM embodies the abstract character of an algorithm which operates on any possible actual input. The following definition 3.10 gives a "more concrete" representation of a computation on some given input in that it takes into account the "global state" of the computation expressed by the contents of the tape to the left and to the right of the reading head.

Definition 3.10 A *situation* of a TM is a quadruple $\langle l, q, c, r \rangle$ where q is the current state, c the symbol currently under the reading head, l the tape to the left and r the tape to the right of the current symbol. For instance

$$\begin{array}{c} \cdots\ \#|a|b|b|\#\ \cdots \\ \uparrow \\ q_i \end{array}$$

corresponds to the situation $\langle ab, q_i, b, \epsilon \rangle$. Notice that l represents only the part of the tape to the left up to the beginning of the infinite sequence of only blanks (resp. r to the right).
A *computation* of a TM M is a sequence of transitions between situations

$$S_0 \mapsto_M S_1 \mapsto_M S_2 \mapsto_M \ldots \tag{3.11}$$

where S_0 is an initial situation and each $S \mapsto_M S'$ is an execution of a single instruction. The reflexive transitive closure of the relation $\mapsto_M$ is written $\mapsto_M^*$.

In example 3.8 we saw a machine accepting (halting on) each sequence of an even number of 1's. Its computation starting on the input 11 expressed as a sequence of transitions between the subsequent situations will be

$$\begin{array}{c} \cdots\ \#|1|1|\#\ \cdots \\ \uparrow \\ q_0 \end{array} \mapsto_M \begin{array}{c} \cdots\ \#|1|1|\#\ \cdots \\ \uparrow \\ q_1 \end{array} \mapsto_M \begin{array}{c} \cdots\ \#|1|1|\#\ \cdots \\ \uparrow \\ q_0 \end{array}$$

In order to capture the manipulation of the whole situations, we need some means of manipulating the strings (to the left and right of the reading head). Given a string s and a symbol $x \in \Sigma$, let xs denote application of a function prepending the symbol x in front of s ($x \dashv s$ from example 2.26). Furthermore, we consider the functions hd and tl returning, resp. the first symbol and the rest of a non-empty string. (E.g., $hd(abc) = a$ and $tl(abc) = bc$.) Since the infinite string of only blanks corresponds to empty input, we will identify such a string with the empty string, $\#^\omega = \epsilon$. Consequently, we let $\#\epsilon = \epsilon$. The functions hd and tl must be adjusted, i.e., $hd(\epsilon) = \#$ and $tl(\epsilon) = \epsilon$.

Basis :: $hd(\epsilon) = \#$ and $tl(\epsilon) = \epsilon$ (with $\#^\omega = \epsilon$)
Ind :: $hd(sx) = x$ and $tl(sx) = s$.

We imagine the reading head some place on the tape and two (infinte) strings staring to the left, resp., right of the head. (Thus, to ease readability, the prepending operation on the left string will be written sx rather than xs.)

Each instruction of a TM can be equivalently expressed as a set of transitions between situations. That is, given a TM M according to definition 3.5, we can construct an equivalent representation of M as a set of transitions between situtations. Each write-instruction $\langle q, a\rangle \mapsto \langle p, b\rangle$, for $a, b \in \Sigma$ corresponds to the transition:

$w: \ \langle l, q, a, r\rangle \vdash \langle l, p, b, r\rangle$

A move-right instruction $\langle q, x\rangle \mapsto \langle p, \mathsf{R}\rangle$ corresponds to
$\mathsf{R}: \ \langle l, q, x, r\rangle \vdash \langle lx, p, hd(r), tl(r)\rangle$

and, analogously, $\langle q, x\rangle \mapsto \langle p, \mathsf{L}\rangle$
$\mathsf{L}: \ \langle l, q, x, r\rangle \vdash \langle tl(l), p, hd(l), xr\rangle$

Notice that, for instance, for L, if $l = \epsilon$ and $x = \#$, the equations we have imposed earlier will yield $\langle \epsilon, q, \#, r\rangle \vdash \langle \epsilon, p, \#, \#r\rangle$. Thus a TM M can be represented as a quadruple $\langle K, \Sigma, q_0, \vdash_M\rangle$, where $\vdash_M$ is a relation (function, actually) on the set of situations $\vdash_M \subseteq \mathsf{Sit} \times \mathsf{Sit}$. (Here, Sit are represented using the functions on strings as above.) For instance, the machine M from example 3.8 will now look as follows:

(1) $\langle l, q_0, 1, r\rangle \vdash \langle l1, q_1, hd(r), tl(r)\rangle$

(2) $\langle l, q_1, 1, r\rangle \vdash \langle l1, q_0, hd(r), tl(r)\rangle$

(3) $\langle l, q_1, \#, r\rangle \vdash \langle l, q_1, \#, r\rangle$

A computation of a TM M according to this representation is a sequence of transitions

$$S_0 \vdash_M S_1 \vdash_M S_2 \vdash_M \ldots \tag{3.12}$$

where each $S \vdash_M S'$ corresponds to one of the specified transitions between situations. The reflexive transitive closure of this relation is denoted $\vdash_M^*$.

It is easily shown (exercise 3.8) that the two representations are equivalent, i.e., a machine obtained by such a transformation will have exactly the same computations (on the same inputs) as the original machine. . . . . . . [end optional]

3: Universal Turing Machine

Informally, we might say that one Turing machine M' *simulates* another one M if M' is able to perform all the computations which can be performed by M or, more precisely, if any input w for M can be represented as an input w' for M' and the result $M'(w')$ represents the result $M(w)$.

This may happen in various ways, the most trivial one being the case when M' is strictly more powerful than M. If M is a multiplication machine (returning $n * m$ for any two natural numbers), while M' can do both multiplication and addition, then augmenting the input w for M with the indication of multiplication, we can use M' to do the same thing as M would do. Another possibility might be some encoding of the instructions of M in such a way that M', using this encoding as a part of its input, can act as if it was M. This is what happens in a computer since a computer program is a description of an algorithm, while an algorithm is just a mechanical procedure for performing computations of some specific type – i.e., it is a Turing machine. A program in a high level language is a Turing machine M – compiling it into a machine code amounts to constructing a machine M' which can simulate M. Execution of $M(w)$ proceeds by representing the high level input w as an input w' acceptable for M', running $M'(w')$ and converting the result back to the high level representation.

We won't define formally the notions of representation and simulation, relying instead on their intuitive understanding and the example of a Universal Turing machine we will present. Such a machine is a Turing machine which can simulate any other Turing machine. It is a conceptual prototype and paradigm of the programmable computers as we know them.

Idea 3.13 [**A Universal TM**] To build a UTM which can simulate an arbitrary TM M

(1) Choose a coding of Turing machines so that they can be represented on an input tape for UTM.
(2) Represent the input of M on the input tape for UTM.
(3) Choose a way of representing the state of the simulated machine M (the current state and position of the head) on the tape of UTM.
(4) Design the set of instructions for the UTM.

To simplify the task, without losing generality, we will assume that the simulated machines work only on the default alphabet $\Sigma = \{*, \#\}$. At the same time, the UTM will use an extended alphabet with several symbols, namely Π, which is the union of the following sets:

- Σ – the alphabet of M
- $\{S, N, \mathsf{R}, \mathsf{L}\}$ – additional symbols to represent instructions of M
- $\{X, Y, 0, 1\}$ – symbols used to keep track of the current state and position
- $\{(, A, B\}$ – auxiliary symbols for bookkeeping

We will code machine M together with its original input as follows:

(	instructions of M	current state	input and head position

(3.14)

1. A possible coding of TMs.

(1) Get the set of instructions from the description of a TM $M = \langle K, \Sigma, q_1, \tau\rangle$.

(2) Each instruction $t \in \tau$ is a four-tuple

$$t: \quad \langle q_i, a\rangle \mapsto \langle q_j, b\rangle$$

where $q_i, q_j \in K$, a is $\#$ or $*$, and $b \in \Sigma \cup \{\mathsf{L}, \mathsf{R}\}$. We assume that states are numbered from 1 up to $n > 0$. Represent t as C_t :

$$C_t : \begin{array}{|c|c|c|c|c|c|c|c|}\hline S & ... & S & a & b & N & ... & N \\ \hline \multicolumn{1}{c}{1} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{i} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{1} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{j}\end{array}$$

i.e., first i S-symbols representing the initial state q_i, then the read symbol a, so the action – either the symbol to be written or R, L, and finally j N-symbols for the final state q_j.

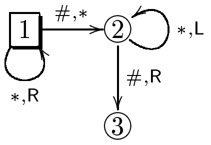
(3) String the representations of all the instructions, with no extra spaces, in increasing order of state numbers. If for a state i there are two instructions, $t_i^{\#}$ for input symbol $\#$ and t_i^* for input symbol $*$, put t_i^* before $t_i^{\#}$.

(4) Put the "end" symbol '(' to the left:

(	C_{t_1}	C_{t_2}	$\cdots$	C_{t_z}	current state $\cdots$

Example 3.15

Let $M = \langle\{q_1, q_2, q_3\}, \{*, \#\}, q_1, \tau\rangle$, where τ is given in the left part of the table:



$\langle q_1, *\rangle \mapsto \langle q_1, \mathsf{R}\rangle$	$S * \mathsf{R}N$
$\langle q_1, \#\rangle \mapsto \langle q_2, *\rangle$	$S\# * NN$
$\langle q_2, *\rangle \mapsto \langle q_2, \mathsf{L}\rangle$	$SS * \mathsf{L}NN$
$\langle q_2, \#\rangle \mapsto \langle q_3, \mathsf{R}\rangle$	$SS\#\mathsf{R}NNN$

The coding of the instructions is given in right part of the table. The whole machine will be coded as:

(	S	$*$	R	N	S	$\#$	$*$	N	N	S	S	$*$	L	N	N	S	S	$\#$	R	N	N	$N \cdots$

It is not necessary to perform the above conversion but – can you tell what M does? $\square$

2. Input representation. We included the alphabet of the original machines $\Sigma = \{*, \#\}$ in the alphabet of the UTM. There is no need to code this part of the simulated machines.

3. Current state. After the representation of the instruction set of M, we will reserve part of the tape for the representation of the current state. There are n states of M, so we reserve $n+1$ fields for unary representation of the number of the current state. The i-th state is represented by i X's followed by $(n+1-i)$ Y's: if M is in the state i, this part of the tape will be:

instructions	X	$\cdots$	X	Y	$\cdots$	Y	input
	1		i			$n{+}1$	

We use $n+1$ positions so that there is always at least one Y to the right of the sequence of X's representing the current state.

To "remember" the current position of the head, we will use the two extra symbols 0 and 1 corresponding, respectively, to # and $*$. The current symbol under the head will be always changed to 0, resp., 1. When the head is moved away, these symbols will be restored back to the original ones #, resp., $*$. For instance, if M's head on the input tape $**\#\#*\#*$ is in the 4-th place, the input part of the UTM tape will be $**\#0*\#*$.

4. Instructions for UTM. We will let UTM start execution with its head at the rightmost X in the bookkeeping section of the tape. After completing the simulation of one step of M's computation, the head will again be placed at the rightmost X. The simulation of each step of computation of M will involve several things:

(4.1) Locate the instruction to be used next.
(4.2) Execute this instruction, i.e., either print a new symbol or move the head on M's tape.
(4.3) Write down the new state in the bookkeeping section.
(4.4) Get ready for the next step: clean up and move the head to the rightmost X.

We indicate the working of UTM at these stages:

4.1. Find instruction. In a loop we erase one X at a time, replacing it by Y, and pass through all the instructions converting one S to A in each instruction. If there are too few S's in an instruction, we convert all the N's to B's in that instruction.

When all X's have been replaced by Y's, the instructions corresponding to the actual state have only A's instead of S. We desactivate the instructions which still contain S by going through all the instructions: if there is some S not converted to A, we replace all N's by B's in that instruction. Now, there remain at most 2 N-lists associated with the instruction(s) for the current state.

We go and read the current symbol on M's tape and replace N's by B's at the instruction (if any) which does not correspond to what we read.

The instruction to be executed has N's – others have only B's.

4.2. Execute instruction. UTM starts now looking for a sequence of N's. If none is found, then M – and UTM – stops. Otherwise, we check what to do looking at the symbol to the left of the leftmost N. If it is R or L, we go to the M's tape and move its head restoring the current symbol to its Σ form and replacing the new symbol by 1, resp. 0. If the instruction is to write a new symbol, we just write the appropriate thing.

4.3. Write new state. We find again the sequence of N's and write the same number of X's in the bookkeeping section indicating the next state.

4.4. Clean up. Finally, convert all A's and B's back to S and N's, and move the head to the rightmost X.

4: Undecidability

Turing machine is a possible formal expression of the idea of *mechanical computability* – we are willing to say that a function is computable iff there is a Turing machine which computes its values for all possible arguments. (Such functions are also called *recursive*.) Notice that if a function is not defined on some arguments (for instance, division by 0) this would require us to assign some special, perhaps new, values for such arguments. For the partial functions one uses a slightly different notion.

function F is		
computable	iff	there is a TM which halts with $F(x)$ for all inputs x
semi-computable	iff	there is a TM which halts with $F(x)$ whenever F is defined on x but does not halt when F is undefined on x

A problem P of YES-NO type (like "is x a member of set S?") gives rise to a special case of function F_P (a predicate) which returns one of the only two values. We get here a third notion.

problem P is		
decidable	iff	F_P is computable – the machine computing F_P always halts returning correct answer YES or NO
semi-decidable	iff	F_P is semi-computable – the machine computing F_P halts with the correct answer YES, but may not halt when the answer is NO
co-semi-decidable	iff	not-F_P is semi-computable – the machine computing F_P halts with the correct answer NO, but may not halt when the answer is YES

Thus a problem is decidable iff it is both semi- and co-semi-decidable.

Set membership is a special case of YES-NO problem but one uses a different terminology:

set S is	iff	the membership problem $x \in S$ is
recursive	iff	decidable
recursively enumerable	iff	semi-decidable
co-recursively enumerable	iff	co-semi-decidable

Again, a set is recursive iff it is both recursively and co-recursively enumerable.

One of the most fundamental results about Turing Machines concerns the undecidability of the Halting Problem. Following our strategy for encoding TMs and their inputs for simulation by a UTM, we assume that the encoding of the instruction set of a machine M is $E(M)$, while the encoding of input w for M is just w itself.

Problem 3.16 [The Halting problem] Is there a Turing machine M_H such that for any machine M and input w, $M_H(E(M), w)$ always halts and

$$M_H(E(M), w) = \begin{cases} Y(es) \text{ if } M(w) \text{ halts} \\ N(o) \text{ if } M(w) \text{ does not halt} \end{cases}$$

The problem is trivially semi-decidable: given an M and w, simply run $M(w)$ and see what happens. If the computation halts, we get the correct YES answer to our problem. If it does not halt, then we may wait forever.

Unfortunately, the following theorem ensures that, in general, there is not much else to do than wait and see what happens.

Theorem 3.17 **[Undecidability of Halting Problem]** There is no Turing machine which decides the halting problem.

Proof. Assume, on the contrary, that there is such a machine M_H.

(1) We can easily design a machine M_1 that is undefined (does not halt) on input Y and defined everywhere else, e.g., a machine with one state q_0 and instruction $\langle q_0, Y\rangle \mapsto \langle q_0, Y\rangle$.
(2) Now, construct machine M_1' which on the input $(E(M), w)$ gives $M_1(M_H(E(M), w))$. It has the property that $M_1'(E(M), w)$ halts iff $M(w)$ does not halt. In particular:
$$M_1'(E(M), E(M)) \text{ halts iff } M(E(M)) \text{ does not halt.}$$
(3) Let M^* be a machine which to an input w first computes (w, w) and then $M_1'(w, w)$. In particular, $M^*(E(M^*)) = M_1'(E(M^*), E(M^*))$. This one has the property that:
$$M^*(E(M^*)) \text{ halts iff } M_1'(E(M^*), E(M^*)) \text{ halts iff } M^*(E(M^*)) \text{ does not halt}$$

This is clearly a contradiction, from which the theorem follows. **QED** (3.17)

Thus the set $\{\langle M, w\rangle : M \text{ halts on input } w\}$ is semi-recursive but not recursive. In terms of programming, the undecidability of Halting Problem means that it is impossible to write a program which could 1) take as input an arbitrary program M and its possible input w and 2) determine whether M run on w will terminate or not.

The theorem gives rise to a series of corollaries identifying other undecidable problems. The usual strategy for such proofs is to show that if a given problem was decidable then we could use it to decide the (halting) problem already known to be undecidable.

Corollary 3.18 There is no Turing machine

(1) M_D which, for any machine M, always halts with $M_D(E(M)) = 0$ iff M is total (always halts) and with 1 iff M is undefined for some input;
(2) M_E which, for given two machines M_1, M_2, always halts with 1 iff the two halt on the same inputs and with 0 otherwise.

Proof. (1) Assume that we have an M_D. Given an M and some input w, we may easily construct a machine M_w which, for any input

x computes $M(w)$. In particular M_w is total iff $M(w)$ halts. Then we can use arbitrary x in $M_D(E(M_w), x)$ to decide halting problem. Hence there is no such M_D.
(2) Assume that we have an M_E. Take as M_1 a machine which does nothing but halts immediately on any input. Then we can use M_E and M_1 to construct an M_D, which does not exist by the previous point.
QED (3.18)

Exercises 3.

EXERCISE 3.1 Suppose that we want to encode the alphabet consisting of 26 (Latin) letters and 10 digits using strings – of fixed length – of symbols from the alphabet $\Delta = \{-, \bullet\}$. What is the minimal length of Δ-strings allowing us to do that? What is the maximal number of distinct symbols which can be represented using the Δ-strings of this length?

The Morse code examplifies such an encoding although it uses additional symbol – corresponding to $\#$ – to separate the representations, and it uses strings of different lengths. For instance, Morse represents A as $\bullet-$ and B as $-\bullet\bullet\bullet$, while 0 as $-----$. (The more frequently a letter is used, the shorter its representation in Morse.) Thus the sequence $\bullet - \# - \bullet\bullet\bullet$ is distinct from $\bullet - - \bullet\bullet\bullet$.

EXERCISE 3.2 The questions at the end of Examples 3.8 and 3.9 (run the respective machines on the suggested inputs).

EXERCISE 3.3 Let $\Sigma = \{1, \#\}$ and a sequence of 1's represent a natural number. Design a TM which starting at the leftmost 1 of the input x computes $x + 1$ by appending 1 at the end of x, and returns the head to the leftmost 1.

EXERCISE 3.4 Consider the alphabet $\Sigma = \{a, b\}$ and the language from example 2.19.1, i.e., $L = \{a^n b^n : n \in \mathbb{N}\}$.

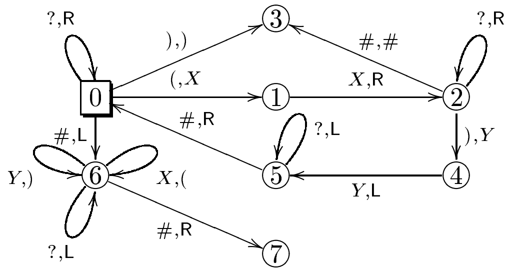
(1) Build a TM M_1 which given a string s over Σ (possibly with additional blank symbol $\#$) halts iff $s \in L$ and goes forever iff $s \notin L$. If you find it necessary, you may allow M_1 to modify the input string.
(2) Modify M_1 to an M_2 which does a similar thing but always halts in the same state indicating the answer. For instance, the answer 'YES' may be indicated by M_2 just halting, and 'NO' by M_2 writing some specific string (e.g., 'NO') and halting.

EXERCISE 3.5 The correct ()-expressions are defined inductively (relatively to a given set S of other expressions):

Basis :: Each $s \in S$ and empty word are correct ()-expressions
Ind. :: If s and t are correct ()-expressions then so are: (s) and st.

(1) Show the following equivalence, i.e., two implications, by the induction on the length of ()-expressions and on their structure, respectively: s is correct iff 1) the numbers of left '(' and right ')' parantheses in s are equal, say n, and 2) for each $1 \leq i \leq n$ the i-th '(' comes before the i-th ')'. (the leftmost '(' comes before the leftmost ')', the second leftmost '(' before the second leftmost ')', etc.)

(2) The machine below reads a ()-expression starting on its leftmost symbol and halting in state 7 iff the input was correct and in state 3 otherwise. Its alphabet Σ consists of two disjoint sets $\Sigma_1 \cap \Sigma_2 = \varnothing$, where Σ_1 is some set of symbols (for writing S-expressions) and $\Sigma_2 = \{X, Y, (,), \#\}$. In the diagram we use an abbreviation '?' to indicate 'any other symbol from Σ not mentioned explicitly among the transitions from this state'. For instance, when in state 2 and reading # the machine goes to state 3 and writes #; reading) it writes Y and goes to state 4 – while reading any other symbol ?, it moves head to the right remaining in state 2.



Run the machine on a couple of your own tapes with ()-expressions (correct and incorrect!). Justify, using the claim from (1), that this machine does the right thing, i.e., *decides* the correctness of ()-expressions.

<u>Exercise 3.6</u> Let $\Sigma = \{a, b, c\}$ and $\Delta = \{0, 1\}$ (Example 3.3). Specify an encoding of Σ in Δ^* and build two Turing machines:

(1) M_c which given a string over Σ converts it to a string over Δ
(2) M_d which given a string over Δ converts it to a string over Σ

The two should act so that their composition gives identity, i.e., for all $s \in \Sigma^* : M_d(M_c(s)) = s$ and, for all $d \in \Delta^* : M_c(M_d(d)) = d$. Choose the initial and final position of the head for both machines so that, executing the one after another will actually produce the same initial string.
Run each of the machines on some example tapes. Run then the two

machines subsequently to check whether the final tape is the same as the initial one.

EXERCISE 3.7 What is the cardinality of the set $TM = \{M \mid M$ is a Turing machine$\}$, assuming a uniform representation of Turing machines, i.e., each machine occurs exactly once in TM?
(1) What is the cardinality of the $\wp(1^*)$?
(2) Now, show that there exists an undecidable subset of 1^*.

EXERCISE 3.8 Use induction on the length of computations to show that, applying the schema from subsection 2.2 of transforming an instruction representation of an arbitrary TM M over the alphabet $\Sigma = \{\#, 1\}$, yields the same machine M. I.e. for any input (initial situation) S_0 the two computations given by (3.11) of definition 3.10 and (3.12) from subsection 2.2 are identical: $S_0 \mapsto_M S_1 \mapsto_M S_2 \mapsto_M \ldots \;=\; S_0 \vdash_M S_1 \vdash_M S_2 \vdash_M \ldots$

The following (optional) exercises concern construction of a UTM.

EXERCISE 3.9 Following the strategy from *1: A possible coding of TMs*, and the Example 3.15, code the machine which you designed in exercise 3.3.

EXERCISE 3.10 Complete the construction of UTM.

(1) Design four TMs to be used in a UTM as described in the four stages of simulation in *4: Instructions for UTM.*
(2) Indicate for each (sub)machine the assumptions about its initial and final situation.
(3) Put the four pieces together and run your UTM on the coding from the previous exercise with some actual inputs.

EXERCISE 3.11 The representation of the tape for the simulated TM M, given in (3.14), seems to allow it to be infinite only in one direction, extending indefinitely to the right, but terminating on the left at the preceding block of X's and Y's coding the current state.
Explain why this is not any real restriction, i.e., why everything computed by a TM with tape infinite in both directions, can also be computed by a TM with tape which is infinite only in one direction.

Chapter 4
Syntax and Proof Systems

- Axiomatic Systems in general
- Syntax of PL
- Proof Systems
- Provable equivalence, Compactness
- Decidability of PL

1: Axiomatic Systems

◊—— a Background Story ——◊

One of the fundamental goals of all scientific inquiry is to achieve precision and clarity of a body of knowledge. This "precision and clarity" means, among other things:

- all assumptions of a given theory are stated explicitly;
- the language of the theory is designed carefully by choosing some basic, primitive notions and defining others in terms of these ones;
- the theory contains some basic principles – all other claims of the theory follow from its basic principles by applications of definitions and some explicit laws.

Axiomatization in a formal system is the ultimate expression of these postulates. Axioms play the role of basic principles – explicitly stated fundamental assumptions, which may be disputable but, once assumed imply the other claims, called theorems. Theorems follow from the axioms not by some unclear arguments but by formal deductions according to well defined rules.

The most famous example of an axiomatisation (and the one which, in more than one way gave the origin to the modern axiomatic systems) was Euclidean geometry. Euclid systematised geometry by showing how many geometrical statements could be logically derived from a small set of axioms and principles. The axioms he postulated were supposed to be intuitively obvious:

A1. Given two points, there is an interval that joins them.
A2. An interval can be prolonged indefinitely.
A3. A circle can be constructed when its center, and a point on it, are given.
A4. All right angles are equal.

There was also the famous fifth axiom – we will return to it shortly. Another part of the system were "common notions" which may be perhaps more adequately called inference rules about equality:

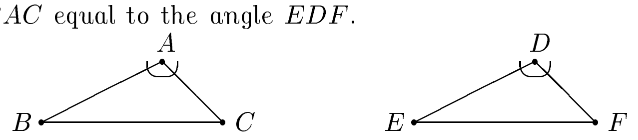
CN1. Things equal to the same thing are equal.
CN2. If equals are added to equals, the wholes are equal.
CN3. If equals are subtracted from equals, the reminders are equal.
CN4. Things that coincide with one another are equal.
CN5. The whole is greater than a part.

Presenting a theory, in this case geometry, as an axiomatic system has tremendous advantages. For the first, it is economical – instead of long lists of facts and claims, we can store only axioms and deduction rules, since the rest is derivable from them. In a sense, axioms and rules "code" the knowledge of the whole field. More importantly, it systematises knowledge by displaying the fundamental assumptions and basic facts which form a logical basis of the field. In a sense, Euclid uncovered "the essence of geometry" by identifying axioms and rules which are sufficient and necessary for deriving all geometrical theorems. Finally, having such a compact presentation of a complicated field, makes it possible to relate not only to particular theorems but also to *the whole field as such.* This possibility is reflected in us speaking about Euclidean geometry vs. non-Euclidean ones. The differences between them concern precisely changes of some basic principles – inclusion or removal of the fifth postulate.

As an example of proof in Euclid's system, we show how using the above axioms and rules he deduced the following proposition (*"Elements"*, Book 1, Proposition 4):

Proposition 4.1 If two triangles have two sides equal to two sides respectively, and have the angles contained by the equal straight lines equal, then they also have the base equal to the base, the triangle equals the triangle, and the remaining angles equal the remaining angles respectively, namely those opposite the equal sides.

Proof. Let ABC and DEF be two triangles having the two sides AB and AC equal to the two sides DE and DF respectively, namely AB equal to DE and AC equal to DF, and the angle BAC equal to the angle EDF.



I say that the base BC also equals the base EF, the triangle ABC equals the triangle DEF, and the remaining angles equal the remaining angles respectively, namely those opposite the equal sides, that is, the angle ABC equals the angle DEF, and the angle ACB equals the angle DFE.

If the triangle ABC is superposed on the triangle DEF, and if the point A is placed on the point D and the straight line AB on DE, then the point B also coincides with E, because AB equals DE.

Again, AB coinciding with DE, the straight line AC also coincides with DF, because the angle BAC equals the angle EDF. Hence the point C also coincides with the point F, because AC again equals DF.

But B also coincides with E, hence the base BC coincides with the base EF and – by CN4. – equals it. Thus the whole triangle ABC coincides with the whole triangle DEF and – by CN4. – equals it. **QED** (4.1)

The proof is allowed to use only the given assumptions, the axioms and the deduction rules. Yet, the Euclidean proofs are not exactly what we mean by a formal proof in an axiomatic system. Why? Because Euclid presupposed a particular *model*, namely, the abstract set of points, lines and figures in an infinite, homogenous space. This presupposition need not be wrong (although, according to modern physics, it is), but it has important bearing on the notion of proof. For instance, it is intuitively obvious what Euclid means by "superposing one triangle on another". Yet, this operation hides some further assumptions, for instance, that length does not change during such a process. This implicit assumption comes most clearly forth in considering the language of Euclid's geometry. Here are just few definitions from *"Elements"*:

D1. A point is that which has no part.
D2. A line is breadthless length.
D3. The ends of a line are points.
D4. A straight line is a line which lies evenly with the points on itself.
D23. Parallel straight lines are straight lines which, being in the same plane and being produced indefinitely in both directions, do not meet one another in either direction.

These are certainly smart formulations but one can wonder if, say, D1 really defines anything or, perhaps, merely states a property of something intended by the name "point". Or else, does D2 define anything *if one does not pressupose* some intuition of what length is? To make a genuinely formal system, one would have to identify some basic notions as truly primitive – that is, with *no intended interpretation.* For these notions one may postulate some properties. For instance, one might say that we have the primitive notions of P, L and IL (for point, line and indefinitely prolonged line). P has no parts; L has two ends, both being P's; any two P's determine an L (whose ends they are – this reminds of A1); any L determines uniquely an IL (cf. A2.), and so on. Then, one may identify derived notions which are defined in terms of the primitive ones. Thus, for instance, the notion of parallel lines can be defined from the primitives as it was done in D23.

The difference may seem negligible but is of the utmost importance. By insisting on the *uniterpreted* character of the primitive notions, it opens an entirely new perspective. On the one hand, we have our primitive, uniterpreted notions. These can be manipulated according to the axioms and rules we have postulated. On the other hand, there are *various* possibilities of *interpretating* these primitive notions. All such interpretations will have to satisfy the axioms and conform to the rules, but otherwise they may be vastly different. This was the insight which led, first, to non-Euclidean geometry and, then, to the formal systems. We will now illustrate this first stage of development.

The strongest, and relatively simple formulation of the famous fifth axiom, the "Parallel Postulate", is as follows:

A5. Given a line L and a point p not on line L, exactly one line L' can be drawn through p parallel to L (i.e., not intersecting L no matter how far extended).

This axiom seems to be much less intuitive than the other ones and mathematicians had spent centuries trying to derive it from the other ones. Failing to do that, they started to ask the question "But, does this postulate have to be true? What if it isn't?"

Well, it may seem that it is true – but how can we check? It may be hard to prolong any line indefinitely. Thus we encouter the other aspect of formal systems, which we will study in the following chapters, namely, what is the *meaning* or *semantics* of such a system. Designing an axiomatic system, one has to specify precisely what are its primitive terms and how these terms may interact in derivation of the theorems. On the other hand, one specifies what these terms are supposed to denote. In fact, terms of a formal system may denote anything which conforms to the rules specified for their interaction. Euclidean geometry was designed with a particular model in mind – the abstract set of points, lines and figures that can be constructed with compass and straightedge in an infinite space. But now, allowing for the primitve character of the basic notions, we can consider other interpretations. We can consider as our space a finite circle C, interpret a P as any point within C, an L as any closed interval within C and an IL as an open-ended chord of the circle, i.e., a straight line within the circle which approaches indefinitely closely, but never touches the circumference. (Thus one can "prolong a line indefinitely" without ever meeting the circumference.) Such an interpretation does not satisfy the fifth postulate.

We start with a line L and a point p not on L. We can then choose two other points x and y and, by A1, obtain two lines xp and yp which can be prolonged indefinitely according to A2. As we see, neither of these indefinitely prolonged lines intersects L. Thus, both are parallel to L according to the very same, old definition D23.

Failing to satify the fifth postulate, this interpretation is not a model of Euclidean geometry. But it is a model of the first non-Euclidean geometry – the Bolyai-Lobachevsky geometry, which keeps all the definitions, postulates and rules except the fifth postulate. Later, many other non-Euclidean geometries have been developed – perhaps the most famous one, by Hermann Minkowski as a fourdimensional space-time universe of the relativity theory.

And now we can observe another advantage of using axiomatic systems. Since non-Euclidean geometry preserves all Euclid's postulates except the fifth one, all the theorems and results which were derived *without* the use of the fifth postulate remain valid. For instance, the proposition 4.1 need no new proof in the new geometries.

This illustrates one of the reasons why axiomatic systems deserve a separate study. Revealing which sets of postulates are needed to establish which consquences, it allows their reuse. Studying some particular phenomena, one can then start by asking which postulates are satisfied by them. An answer to this question yields then immediately all the theorems which have been proven from these postulates.

It is of crucial importance and should be constantly kept in mind that axiomatic systems, their primitive terms and proofs, are purely syntactic, that is, do *not presuppose any interpretation.* Of course, their eventual usefulness depends on whether we can find interesting interpretations for their terms and rules but this is another story. In this chapter, we study some fundamental axiomatic systems without considering such interpretations, which will be addressed later on.

Recall that an inductive definition of a set consists of a BASIS, an INDUCTION part, and an implicit CLOSURE condition. When the set defined is a language, i.e., a set of strings, we often talk about an *axiomatic system.* In this case, the elements of the basis are called *axioms,*, the induction part is given by a set of *proof rules*, and *theorems* are the members of so defined set. The symbol $\vdash$ denotes the set of theorems, i.e., $A \in \vdash$ iff A is a theorem but the statement $A \in \vdash$ is written $\vdash A$. Usually $\vdash$ is identified as a subset of some other language $\mathsf{L} \subseteq \Sigma^*$, thus $\vdash \subseteq \mathsf{L} \subseteq \Sigma^*$.

Definition 4.2 An axiomatic system $\vdash$, over an $\mathsf{L} \subseteq \Sigma^*$, has the form:

AXIOMS :: A set $Ax \subseteq \vdash \subseteq \mathsf{L}$, and

PROOF RULES :: of the form: "if $A_1 \in \vdash, \ldots, A_n \in \vdash$ then $C \in \vdash$", i.e., elements $R \in \mathsf{L}^n \times \mathsf{L}$, written $R : \dfrac{\vdash A_1; \ldots; \vdash A_n}{\vdash C}$.

A_i are premisses and C conclusion of the rule R.

The rules are always designed so that C is in L if $A_1, \ldots, A_n$ are, thus $\vdash$ is guaranteed to be a subset of L. A formula A is a *theorem* of the system iff there is a proof of A in the system.

Definition 4.3 A *proof* in an axiomatic system is a finite sequence $A_1, \ldots, A_n$ of strings from L, such that for each A_i

- either $A_i \in Ax$ or else
- there are $A_{i_1}, \ldots, A_{i_k}$ in the sequence with all $i_1, \ldots, i_k < i$, and an application of a proof rule $R : \dfrac{\vdash A_{i_1}; \ldots; \vdash A_{i_k}}{\vdash A_i}$.

A proof *of* A is a proof in which A is the final string.

Remark.
Notice that for a given language L there may be several axiomatic systems which all define the same subset of L, albeit, by means of very different rules.
There are also variations which we will consider, where the predicate $\vdash$ is defined on various sets built over L, for instance, $\wp(\mathsf{L}) \times \mathsf{L}$.

2: Syntax of PL

The basic logical system, originating with Boole's algebra, is Propositional Logic (PL). The name reflects the fact that the expressions of the language are "intended as" propositions. This interpretation will be part of the semantics of PL to be discussed in the following chapters. Here we introduce syntax and the associated axiomatic proof system of PL.

Definition 4.4 The language of well-formed formulae of PL is defined as follows:

(1) An alphabet for an PL language consists of a set of propositional variables $\Sigma = \{a, b, c...\}$, together with the (formula building) connectives: $\neg$ and $\rightarrow$, and auxiliary symbols $(,)$.
(2) The *well-formed formulae*, $\mathsf{WFF}^{\Sigma}_{\mathsf{PL}}$, are defined inductively:

$$\begin{aligned} \textsc{Basis} &:: \Sigma \subseteq \mathsf{WFF}^{\Sigma}_{\mathsf{PL}}; \\ \textsc{Ind} &:: 1) \text{ if } A \in \mathsf{WFF}^{\Sigma}_{\mathsf{PL}} \text{ then } \neg A \in \mathsf{WFF}^{\Sigma}_{\mathsf{PL}} \\ & \quad\;\; 2) \text{ if } A, B \in \mathsf{WFF}^{\Sigma}_{\mathsf{PL}} \text{ then } (A \rightarrow B) \in \mathsf{WFF}^{\Sigma}_{\mathsf{PL}}. \end{aligned}$$

(3) The propositional variables are called *atomic formulae*, the formulae of the form A or $\neg A$, where A is atomic are called *literals*.

Remark 4.5 **[Some conventions]**
1) Compare this definition to Exercise 2.2.

2) The outermost pair of parantheses is often suppressed, hence $A \rightarrow (B \rightarrow C)$ stands for the formula $(A \rightarrow (B \rightarrow C))$ while $(A \rightarrow B) \rightarrow C$ stands for the

formula $((A \to B) \to C)$.

3) Formulae are strings over the symbols in $\Sigma \cup \{), (, \to, \neg\}$, i.e., $\mathsf{WFF}^{\Sigma}_{\mathsf{PL}} \subseteq (\Sigma \cup \{), (, \to, \neg\})^*$. We use lower case letters for the propositional variables of an alphabet Σ, while upper case letters stand for arbitrary formulae. The sets of $\mathsf{WFF}^{\Sigma}_{\mathsf{PL}}$ over $\Sigma = \{a, b\}$ and over $\Sigma_1 = \{c, d\}$ are disjoint (though in one-to-one correspondence). Thus, the definition yields a different set of formulae for different Σ's. Writing $\mathsf{WFF}_{\mathsf{PL}}$ we mean well-formed PL formulae over an arbitrary alphabet and most of our discussion is concerned with this general case irrespectively of a particular alphabet Σ.

4) It is always implicitly assumed that $\Sigma \neq \varnothing$.

5) For the reasons which we will explain later, occasionally, we may use the abbreviations $\bot$ for $\neg(B \to B)$ and $\top$ for $B \to B$, for arbitrary B.

In the following we will always – unless explicitly stated otherwise – assume that the formulae involved are well-formed.

3: Hilbert's axiomatic system

Hilbert's system $\mathcal{H}$ for PL is defined with respect to a unary relation (predicate) $\vdash_{\mathcal{H}} \subseteq \mathsf{WFF}_{\mathsf{PL}}$ which we write as $\vdash_{\mathcal{H}} B$ rather than as $B \in \vdash_{\mathcal{H}}$. It reads as "*$B$ is provable in $\mathcal{H}$*".

Definition 4.6 The predicate $\vdash_{\mathcal{H}}$ of *Hilbert's system* for PL is defined inductively by:

AXIOMS :: A1: $\vdash_{\mathcal{H}} A \to (B \to A)$;
A2: $\vdash_{\mathcal{H}} (A \to (B \to C)) \;\to\; ((A \to B) \to (A \to C))$;
A3: $\vdash_{\mathcal{H}} (\neg B \to \neg A) \;\to\; (A \to B)$;

PROOF RULE :: called *Modus Ponens:* $\dfrac{\vdash_{\mathcal{H}} A \;\; ; \;\; \vdash_{\mathcal{H}} A \to B}{\vdash_{\mathcal{H}} B}$.

Remark [Axioms vs. axiom schemata]
A1–A3 are in fact axiom *schemata*; the actual axioms comprise all formulae of the indicated form with the letters A, B, C instantiated to arbitrary formulae. For each particular alphabet Σ, there will be a different (infinite) collection of actual axioms. Similar instantiations are performed in the proof rule. For instance, for $\Sigma = \{a, b, c, d\}$, all the following formulae are instances of axiom schemata:

$A1 : b \to (a \to b),\ (b \to d) \to (\neg a \to (b \to d)),\ a \to (a \to a),$

$A3 : (\neg\neg d \to \neg b) \to (b \to \neg d).$

The following formulae are *not* (instances of the) axioms:

$a \to (b \to b),\ (\neg b \to a) \to (\neg a \to b)$.
Hence, an axiom schema, like $A1$, is actually a predicate giving, for any Σ, the set of Σ-instances $A1^{\Sigma} = \{x \to (y \to x) : x, y \in \mathsf{WFF}^{\Sigma}_{\mathsf{PL}}\} \subset \mathsf{WFF}^{\Sigma}_{\mathsf{PL}}$.

Also any proof rule R with n premises (in an axiomatic system over a language L) is typically given as a *schema* – a relation $R \subseteq \mathsf{L}^n \times \mathsf{L}$. A proof rule as in Definition 4.2 is just one element of this relation, which is called an "application of the rule". For a given Σ, Hilbert's Modus Ponens schema yields an infinite set of its applications $MP^{\Sigma} = \{\langle x, x \to y, y\rangle : x, y \in \mathsf{WFF}^{\Sigma}_{\mathsf{PL}}\} \subset \mathsf{WFF}^{\Sigma}_{\mathsf{PL}} \times \mathsf{WFF}^{\Sigma}_{\mathsf{PL}} \times \mathsf{WFF}^{\Sigma}_{\mathsf{PL}}$. The following are examples of such applications for $\{a, b, c\} \subseteq \Sigma$:

$$\frac{\vdash_{\mathcal{H}} a \ \ ; \ \ \vdash_{\mathcal{H}} a \to b}{\vdash_{\mathcal{H}} b} \qquad \frac{\vdash_{\mathcal{H}} a \to \neg b \ \ ; \ \ \vdash_{\mathcal{H}} (a \to \neg b) \to (b \to c)}{\vdash_{\mathcal{H}} b \to c} \quad \ldots$$

In $\mathcal{H}$, the sets Ai^{Σ} and MP^{Σ} are *recursive*, provided that Σ is (which it always is by assumption). Recursivity of MP^{Σ} means that we can always *decide* whether a given triple of formulae is an application of the rule. Recursivity of the set of axioms means that we can always *decide* whether a given formula is an axiom or not. Axiomatic systems which do not satisfy these conditions are of lesser interest and we will not consider them. □

That both Ai^{Σ} and MP^{Σ} of $\mathcal{H}$ are recursive sets does not imply that so is $\vdash_{\mathcal{H}}$. This only means that *given* a sequence of formulae, we can *decide* whether it is a proof or not. To decide if a given formula belongs to $\vdash_{\mathcal{H}}$ would require a procedure for deciding if such a proof exists – probably, a procedure for *constructing* a proof. We will see several examples illustrating that, even if such a procedure for $\vdash_{\mathcal{H}}$ exists, it is by no means simple.

Lemma 4.7 For an arbitrary $B \in \mathsf{WFF}_{\mathsf{PL}} : \vdash_{\mathcal{H}} B \to B$

Proof.

$1 : \vdash_{\mathcal{H}} (B \to ((B \to B) \to B)) \ \to \ ((B \to (B \to B)) \to (B \to B))$	$A2$
$2 : \vdash_{\mathcal{H}} B \to ((B \to B) \to B)$	$A1$
$3 : \vdash_{\mathcal{H}} (B \to (B \to B)) \to (B \to B)$	$MP(2,1)$
$4 : \vdash_{\mathcal{H}} B \to (B \to B)$	$A1$
$5 : \vdash_{\mathcal{H}} B \to B$	$MP(4,3)$

QED (4.7)

The phrase "for an arbitrary $B \in \mathsf{WFF}_{\mathsf{PL}}$" indicates that any formula of the above form (with any well-formed formula over any actual alphabet Σ substituted for B) will be derivable, e.g. $\vdash_{\mathcal{H}} a \to a$, $\vdash_{\mathcal{H}} (a \to \neg b) \to (a \to \neg b)$, etc. All the results concerning PL will be stated in this way.

But we cannot substitute different formulae for the two occurences of B. If we try to apply the above proof to deduce $\vdash_{\mathcal{H}} A \to B$ it will fail – identify the place(s) where it would require invalid transitions.

In addition to provability of simple formulae, also derivations can be "stored" for future use. The above lemma means that we can always, for arbitrary formula B, use $\vdash_{\mathcal{H}} B \to B$ as a step in a proof. More generally, we can "store" derivations in the form of admissible rules.

Definition 4.8 Let $\mathcal{C}$ be an axiomatic system. A rule $\dfrac{\vdash_{\mathcal{C}} A_1 \ ; \ \dots \ ; \vdash_{\mathcal{C}} A_n}{\vdash_{\mathcal{C}} C}$ is *admissible* in $\mathcal{C}$ if whenever there are proofs in $\mathcal{C}$ of all the premisses, i.e., $\vdash_{\mathcal{C}} A_i$ for all $1 \leq i \leq n$, then there is a proof in $\mathcal{C}$ of the conclusion $\vdash_{\mathcal{C}} C$.

Lemma 4.9 The following rules are admissible in $\mathcal{H}$:

1. $\dfrac{\vdash_{\mathcal{H}} A \to B \ \ ; \ \ \vdash_{\mathcal{H}} B \to C}{\vdash_{\mathcal{H}} A \to C}$ 2. $\dfrac{\vdash_{\mathcal{H}} B}{\vdash_{\mathcal{H}} A \to B}$

Proof.

1.
$$\begin{array}{lll}
1: & \vdash_{\mathcal{H}} (A \to (B \to C)) \ \to \ ((A \to B) \to (A \to C)) & A2 \\
2: & \vdash_{\mathcal{H}} (B \to C) \ \to \ (A \to (B \to C)) & A1 \\
3: & \vdash_{\mathcal{H}} B \to C & assumption \\
4: & \vdash_{\mathcal{H}} A \to (B \to C) & MP(3,2) \\
5: & \vdash_{\mathcal{H}} (A \to B) \to (A \to C) & MP(4,1) \\
6: & \vdash_{\mathcal{H}} A \to B & assumption \\
7: & \vdash_{\mathcal{H}} A \to C & MP(6,5)
\end{array}$$

2.
$$\begin{array}{lll}
1: & \vdash_{\mathcal{H}} B & assumption \\
2: & \vdash_{\mathcal{H}} B \to (A \to B) & A1 \\
3: & \vdash_{\mathcal{H}} A \to B & MP(1,2)
\end{array}$$

QED (4.9)

Lemma 4.10 1. $\vdash_{\mathcal{H}} \neg\neg B \to B$ 2. $\vdash_{\mathcal{H}} B \to \neg\neg B$

Proof.

1.
$$\begin{array}{lll}
1: & \vdash_{\mathcal{H}} \neg\neg B \to (\neg\neg\neg\neg B \to \neg\neg B) & A1 \\
2: & \vdash_{\mathcal{H}} (\neg\neg\neg\neg B \to \neg\neg B) \to (\neg B \to \neg\neg\neg B) & A3 \\
3: & \vdash_{\mathcal{H}} \neg\neg B \to (\neg B \to \neg\neg\neg B) & L.4.9.1\ (1,2) \\
4: & \vdash_{\mathcal{H}} (\neg B \to \neg\neg\neg B) \to (\neg\neg B \to B) & A3 \\
5: & \vdash_{\mathcal{H}} \neg\neg B \to (\neg\neg B \to B) & L.4.9.1\ (3,4) \\
6: & \vdash_{\mathcal{H}} (\neg\neg B \to (\neg\neg B \to B)) \to & \\
 & \qquad ((\neg\neg B \to \neg\neg B) \to (\neg\neg B \to B)) & A2 \\
7: & \vdash_{\mathcal{H}} (\neg\neg B \to \neg\neg B) \to (\neg\neg B \to B) & MP(5,6) \\
8: & \vdash_{\mathcal{H}} \neg\neg B \to B & MP(L.4.7, 7)
\end{array}$$

2.
$$\begin{array}{lll}
1: & \vdash_{\mathcal{H}} \neg\neg\neg B \to \neg B & point\ 1. \\
2: & \vdash_{\mathcal{H}} (\neg\neg\neg B \to \neg B) \to (B \to \neg\neg B) & A3 \\
3: & \vdash_{\mathcal{H}} B \to \neg\neg B & MP(1,2)
\end{array}$$

QED (4.10)

4: The system $\mathcal{N}$

In the system $\mathcal{N}$, instead of the unary predicate $\vdash_{\mathcal{H}}$ we use a binary relation $\vdash_{\mathcal{N}} \subseteq \wp(\mathsf{WFF}_{\mathsf{PL}}) \times \mathsf{WFF}_{\mathsf{PL}}$, written as $\Gamma \vdash_{\mathcal{N}} B$. It reads as "*$B$ is provable in $\mathcal{N}$ from the assumptions* Γ".

Definition 4.11 The axioms and rules of $\mathcal{N}$ are as in Hilbert's system with the additional axiom schema A0:

Axioms :: A0: $\Gamma \vdash_{\mathcal{N}} B$, whenever $B \in \Gamma$;
A1: $\Gamma \vdash_{\mathcal{N}} A \to (B \to A)$;
A2: $\Gamma \vdash_{\mathcal{N}} (A \to (B \to C)) \ \to \ ((A \to B) \to (A \to C))$;
A3: $\Gamma \vdash_{\mathcal{N}} (\neg B \to \neg A) \ \to \ (A \to B)$;

Proof Rule :: Modus Ponens: $\dfrac{\Gamma \vdash_{\mathcal{N}} A \ \ ; \ \ \Gamma \vdash_{\mathcal{N}} A \to B}{\Gamma \vdash_{\mathcal{N}} B}$.

Remark.
As for $\mathcal{H}$, the "axioms" are actually *schemata.* The real set of axioms is the infinite set of actual formulae obtained from these schemata by substituting actual formulae for the upper case variables. Similarly for the proof rule.

The next lemma corresponds exactly to lemma 4.9. In fact, the proof of that lemma (and most others) can be taken over line for line from $\mathcal{H}$, with hardly any modification (just replace $\vdash_{\mathcal{H}}$ by $\Gamma \vdash_{\mathcal{N}}$) to serve as a proof of this lemma.

Lemma 4.12 The following rules are admissible in $\mathcal{N}$:

1. $\dfrac{\Gamma \vdash_{\mathcal{N}} A \to B \ \ ; \ \ \Gamma \vdash_{\mathcal{N}} B \to C}{\Gamma \vdash_{\mathcal{N}} A \to C}$
2. $\dfrac{\Gamma \vdash_{\mathcal{N}} B}{\Gamma \vdash_{\mathcal{N}} A \to B}$

The name $\mathcal{N}$ reflects the intended association with the so called "natural deduction" reasoning. This system is not exactly what is usually so called and we have adopted $\mathcal{N}$ because it corresponds so closely to $\mathcal{H}$. But while $\mathcal{H}$ derives only single formulae, tautologies, $\mathcal{N}$ provides also means for *reasoning from the assumptions* Γ. This is the central feature which it shares with natural deduction systems: they both satisfy the following Deduction Theorem. (The expression "Γ, A" is short for "$\Gamma \cup \{A\}$.")

Theorem 4.13 **[Deduction Theorem]** If $\Gamma, A \vdash_{\mathcal{N}} B$, then $\Gamma \vdash_{\mathcal{N}} A \to B$.

Proof. By induction on the length l of a proof of $\Gamma, A \vdash_{\mathcal{N}} B$. Basis, $l = 1$, means that the proof consists merely of an instance of an axiom and it has two cases depending on which axiom was involved:

A1-A3 :: If B is one of these axioms, then we also have $\Gamma \vdash_{\mathcal{N}} B$ and lemma 4.12.2 gives the conclusion.

A0 :: If B results from this axiom, we have two subcases:

(1) If $B = A$, then Lemma 4.7 gives that $\Gamma \vdash_{\mathcal{N}} B \to B$.

(2) If $B \neq A$, then $B \in \Gamma$, and so $\Gamma \vdash_{\mathcal{N}} B$. Lemma 4.12.2 gives $\Gamma \vdash_{\mathcal{N}} A \to B$.

MP :: B follows by MP: $\dfrac{\Gamma, A \vdash_{\mathcal{N}} C \quad ; \quad \Gamma, A \vdash_{\mathcal{N}} C \to B}{\Gamma, A \vdash_{\mathcal{N}} B}$

By the induction hypothesis, we have the first two lines of the following proof:

$1 : \Gamma \vdash_{\mathcal{N}} A \to C$
$2 : \Gamma \vdash_{\mathcal{N}} A \to (C \to B)$
$3 : \Gamma \vdash_{\mathcal{N}} (A \to (C \to B)) \;\to\; ((A \to C) \to (A \to B))$ $A2$
$4 : \Gamma \vdash_{\mathcal{N}} (A \to C) \;\to\; (A \to B)$ $MP(2,3)$
$5 : \Gamma \vdash_{\mathcal{N}} A \to B$ $MP(1,4)$

QED (4.13)

Example 4.14
Using Deduction Theorem significantly shortens the proofs. The tedious proof of Lemma 4.7 can be now recast as:

$1 : B \vdash_{\mathcal{N}} B$ $A0$
$2 : \vdash_{\mathcal{N}} B \to B$ DT

□

Lemma 4.15 $\vdash_{\mathcal{N}} (A \to B) \to (\neg B \to \neg A)$

Proof.
$1 : A \to B \vdash_{\mathcal{N}} (\neg\neg A \to \neg\neg B) \to (\neg B \to \neg A)$ $A3$
$2 : A \to B \vdash_{\mathcal{N}} \neg\neg A \to A$ $L.4.10.1$
$3 : A \to B \vdash_{\mathcal{N}} A \to B$ $A0$
$4 : A \to B \vdash_{\mathcal{N}} \neg\neg A \to B$ $L.4.12.1(2,3)$
$5 : A \to B \vdash_{\mathcal{N}} B \to \neg\neg B$ $L.4.10.2$
$6 : A \to B \vdash_{\mathcal{N}} \neg\neg A \to \neg\neg B$ $L.4.12.1(4,5)$
$7 : A \to B \vdash_{\mathcal{N}} \neg B \to \neg A$ $MP(6,1)$
$8 : \vdash_{\mathcal{N}} (A \to B) \to (\neg B \to \neg A)$ DT **QED** (4.15)

Deduction Theorem is a kind of dual to MP: each gives one implication of the following

Corollary 4.16 $\Gamma, A \vdash_{\mathcal{N}} B$ iff $\Gamma \vdash_{\mathcal{N}} A \to B$.

Proof. $\Rightarrow$) is Deduction Theorem 4.13. $\Leftarrow$) By Exercise 4.4, the assumption can be strengthened to $\Gamma, A \vdash_{\mathcal{N}} A \to B$. But then, also $\Gamma, A \vdash_{\mathcal{N}} A$, and by MP $\Gamma, A \vdash_{\mathcal{N}} B$. **QED** (4.16)

We can now easily show the following:

Corollary 4.17 The following rule is admissible in $\mathcal{N}$: $\dfrac{\Gamma \vdash_{\mathcal{N}} A \to (B \to C)}{\Gamma \vdash_{\mathcal{N}} B \to (A \to C)}$

Proof. Follows trivially from the above 4.16: $\Gamma \vdash_{\mathcal{N}} A \to (B \to C)$ iff $\Gamma, A \vdash_{\mathcal{N}} B \to C$ iff $\Gamma, A, B \vdash_{\mathcal{N}} C$. As Γ, A, B abbreviates the set $\Gamma \cup \{A, B\}$, this is also equivalent to $\Gamma, B \vdash_{\mathcal{N}} A \to C$, and then to $\Gamma \vdash_{\mathcal{N}} B \to (A \to C)$. **QED** (4.17)

5: $\mathcal{H}$ vs. $\mathcal{N}$

In $\mathcal{H}$ we prove only single formulae, while in $\mathcal{N}$ we work "from the assumptions" proving their consequences. Since the axiom schemata and rules of $\mathcal{H}$ are special cases of their counterparts in $\mathcal{N}$, it is obvious that for any formula B, if $\vdash_{\mathcal{H}} B$ then $\varnothing \vdash_{\mathcal{N}} B$. In fact this can be strengthened to an equivalence. (We follow the convention of writing $\vdash_{\mathcal{N}} B$ for $\varnothing \vdash_{\mathcal{N}} B$.)

Lemma 4.18 For any formula B we have: $\vdash_{\mathcal{H}} B$ iff $\vdash_{\mathcal{N}} B$.

Proof. One direction is noted above. In fact, any proof of $\vdash_{\mathcal{H}} B$ itself qualifies as a proof of $\vdash_{\mathcal{N}} B$. The other direction is almost as obvious, since there is no way to make any real use of A0 in a proof of $\vdash_{\mathcal{N}} B$. More precisely, take any proof of $\vdash_{\mathcal{N}} B$ and delete all lines (if any) of the form $\Gamma \vdash_{\mathcal{N}} A$ for $\Gamma \neq \varnothing$. The result is still a proof of $\vdash_{\mathcal{N}} B$, and now also of $\vdash_{\mathcal{H}} B$.

More formally, the lemma can be proved by induction on the length of a proof of $\vdash_{\mathcal{N}} B$: Since $\Gamma = \varnothing$ the last step of the proof could have used either an axiom A1, A2, A3 or MP. The same step can be then done in $\mathcal{H}$ – for MP, the proofs of $\vdash_{\mathcal{N}} A$ and $\vdash_{\mathcal{N}} A \to B$ for the appropriate A are shorter and hence by the IH have counterparts in $\mathcal{H}$. **QED** (4.18)

The next lemma is a further generalization of this result.

Lemma 4.19 $\vdash_{\mathcal{H}} G_1 \to (G_2 \to ...(G_n \to B)...)$ iff $\{G_1, G_2, \ldots, G_n\} \vdash_{\mathcal{N}} B$.

Proof. We prove the lemma by induction on n:

Basis :: The special case corresponding to $n = 0$ is just the previous lemma.

Ind. :: Suppose the IH:
$\vdash_{\mathcal{H}} G_1 \to (G_2 \to ...(G_n \to B)..)$ iff $\{G_1, G_2...G_n\} \vdash_{\mathcal{N}} B$
for any B. Then, taking $(G_{n+1} \to B)$ for B, we obtain
$\vdash_{\mathcal{H}} G_1 \to (G_2 \to ...(G_n \to (G_{n+1} \to B))..)$

$$\begin{array}{rcl} \text{(by IH)} & \text{iff} & \{G_1, G_2...G_n\} \vdash_{\mathcal{N}} (G_{n+1} \to B) \\ \text{(by Corollary 4.16)} & \text{iff} & \{G_1, G_2, \ldots, G_n, G_{n+1}\} \vdash_{\mathcal{N}} B. \end{array}$$

QED (4.19)

Lemma 4.18 states the equivalence of $\mathcal{N}$ and $\mathcal{H}$ with respect to the simple formulae of $\mathcal{H}$. This lemma states a more general equivalence of these two systems: for any finite $\mathcal{N}$-expression $B \in \vdash_{\mathcal{N}}$ there is a corresponding $\mathcal{H}$-formula $B' \in \vdash_{\mathcal{H}}$ and vice versa.

Observe, however, that this equivalence is restricted to finite Γ in $\mathcal{N}$-expressions. The significant difference between the two systems consists in that $\mathcal{N}$ allows to consider also consequences of infinite sets of assumptions, for which there are no corresponding formulae in $\mathcal{H}$, since every formula must be finite.

6: Provable Equivalence of formulae

Equational reasoning is based on the simple principle of substitution of equals for equals. E.g., having the arithmetical expression $2 + (7 + 3)$ and knowing that $7+3 = 10$, we also obtain $2+(7+3) = 2+10$. The rule applied in such cases may be written as $\dfrac{a = b}{F[a] = F[b]}$ where $F[\_]$ is an expression "with a hole" (a variable or a placeholder) into which we may substitute other expressions. We now illustrate a logical counterpart of this idea.

Lemma 4.7 showed that any formula of the form $(B \to B)$ is derivable in $\mathcal{H}$ and, by lemma 4.18, in $\mathcal{N}$. It allows us to use, for instance, 1) $\vdash_{\mathcal{N}} a \to a$, 2) $\vdash_{\mathcal{N}} (a \to b) \to (a \to b)$, 3) $\ldots$ as a step in any proof. Putting it a bit differently, the lemma says that 1) is provable iff 2) is provable iff ... Recall the abbreviation $\top$ for an *arbitrary* formula of this form introduced in Remark 4.5. It also introduced the abbreviation $\bot$ for an *arbitrary* formula of the form $\neg(B \to B)$. These abbreviations indicate that all the formulae of the respective form are equivalent in the following sense.

Definition 4.20 Formulae A and B are *provably equivalent* in an axiomatic system $\mathcal{C}$ for PL, if both $\vdash_{\mathcal{C}} A \to B$ and $\vdash_{\mathcal{C}} B \to A$. If this is the case, we write $\vdash_{\mathcal{C}} A \leftrightarrow B$.[2]

Lemma 4.10 provides an example, namely

$$\vdash_{\mathcal{H}} B \leftrightarrow \neg\neg B \tag{4.21}$$

Another example follows from axiom A3 and lemma 4.15:

$$\vdash_{\mathcal{N}} (A \to B) \leftrightarrow (\neg B \to \neg A) \tag{4.22}$$

In Exercise 4.2, you are asked to show that all formulae $\top$ are provably equivalent, i.e.,

$$\vdash_{\mathcal{N}} (A \to A) \leftrightarrow (B \to B). \tag{4.23}$$

To show the analogous equivalence of all $\bot$ formulae,

$$\vdash_{\mathcal{N}} \neg(A \to A) \leftrightarrow \neg(B \to B), \tag{4.24}$$

we have to proceed differently since we do not have $\vdash_{\mathcal{N}} \neg(B \to B)$.[3] We can use the above fact and lemma 4.15:

$1: \vdash_{\mathcal{N}} (A \to A) \to (B \to B)$
$2: \vdash_{\mathcal{N}} ((A \to A) \to (B \to B)) \to (\neg(B \to B) \to \neg(A \to A))$ $L.4.15$
$3: \vdash_{\mathcal{N}} \neg(B \to B) \to \neg(A \to A)$ $MP(1,2)$

and the opposite implication is again an instance of this one.

Provable equivalence $A \leftrightarrow B$ means – and it is its main importance – that the formulae are interchangeable. Whenever we have a proof of a formula $F[A]$ containing A (as a subformula, possibly with several occurences), we can replace A by B – the result will be provable too. This fact is a powerful tool in simplifying proofs and is expressed in the following theorem. (The analogous version holds for $\mathcal{H}$.)

Theorem 4.25 The following rule is admissible in $\mathcal{N}$: $\dfrac{\vdash_{\mathcal{N}} A \leftrightarrow B}{\vdash_{\mathcal{N}} F[A] \leftrightarrow F[B]}$, for any formula $F[A]$.

Proof. By induction on the complexity of $F[\_]$ viewed as a formula "with a hole" (where there may be several occurences of the "hole", i.e., $F[\_]$ may have the form $\neg[\_] \to G$ or else $[\_] \to (\neg G \to ([\_] \to H))$, etc.).

[2] In the view of lemma 4.7 and 4.9.1, and their generalizations to $\mathcal{N}$, the relation $Im \subseteq \mathsf{WFF}^{\Sigma}_{\mathsf{PL}} \times \mathsf{WFF}^{\Sigma}_{\mathsf{PL}}$ given by $Im(A,B) \Leftrightarrow \vdash_{\mathcal{N}} A \to B$ is reflexive and transitive. This definition amounts to adding the requirement of symmetricity making $\leftrightarrow$ the greatest equivalence contained in Im.

[3] In fact, this is not true, as we will see later on.

$[\_]$ :: i.e., $F[A] = A$ and $F[B] = B$ – the conclusion is then the same as the premise.

$\neg G[\_]$ :: IH allows us to assume the claim for $G[\_]$: $\dfrac{\vdash_{\mathcal{N}} A \leftrightarrow B}{\vdash_{\mathcal{N}} G[A] \leftrightarrow G[B]}$.

$1 : \vdash_{\mathcal{N}} A \to B$ — *assumption*
$2 : \vdash_{\mathcal{N}} G[A] \to G[B]$ — IH
$3 : \vdash_{\mathcal{N}} (G[A] \to G[B]) \to (\neg G[B] \to \neg G[A])$ — $L.4.15$
$4 : \vdash_{\mathcal{N}} \neg G[B] \to \neg G[A]$ — $MP(2,3)$

The same for $\vdash_{\mathcal{N}} \neg G[A] \to \neg G[B]$, starting with the assumption $\vdash_{\mathcal{N}} B \to A$.

$G[\_] \to H[\_]$ :: Assuming $\vdash_{\mathcal{N}} A \leftrightarrow B$, IH gives us the following ssumptions : $\vdash_{\mathcal{N}} G[A] \to G[B]$, $\vdash_{\mathcal{N}} G[B] \to G[A]$, $\vdash_{\mathcal{N}} H[A] \to H[B]$ and $\vdash_{\mathcal{N}} H[B] \to H[A]$. We show only the implication $\vdash_{\mathcal{N}} F[A] \to F[B]$:

$1 : \vdash_{\mathcal{N}} H[A] \to H[B]$ — IH
$2 : G[A] \to H[A] \vdash_{\mathcal{N}} H[A] \to H[B]$ — $exc.4.4$
$3 : G[A] \to H[A] \vdash_{\mathcal{N}} G[A] \to H[A]$ — $A0$
$4 : G[A] \to H[A] \vdash_{\mathcal{N}} G[A] \to H[B]$ — $L.4.12.1(3,2)$
$5 : \vdash_{\mathcal{N}} G[B] \to G[A]$ — IH
$6 : G[A] \to H[A] \vdash_{\mathcal{N}} G[B] \to G[A]$ — $exc.4.4$
$7 : G[A] \to H[A] \vdash_{\mathcal{N}} G[B] \to H[B]$ — $L.4.12.1(6,4)$
$8 : \vdash_{\mathcal{N}} (G[A] \to H[A]) \to (G[B] \to H[B])$ — $DT(7)$

Entirely symmetric proof yields the other implication $\vdash_{\mathcal{N}} F[B] \to F[A]$.

QED (4.25)

The theorem, together with the preceding observations about equivalence of all $\top$ and all $\bot$ formulae justify the use of these abbreviations: in a proof, any formula of the form $\bot$, resp. $\top$, can be replaced by any other formula of the same form. As a simple consequence of the theorem, we obtain:

Corollary 4.26 For any formula $F[A]$, the following rule is admissible: $\dfrac{\vdash_{\mathcal{N}} F[A] \ ; \ \vdash_{\mathcal{N}} A \leftrightarrow B}{\vdash_{\mathcal{N}} F[B]}$

Proof. If $\vdash_{\mathcal{N}} A \leftrightarrow B$, theorem 4.25 gives us $\vdash_{\mathcal{N}} F[A] \leftrightarrow F[B]$ which, in particular, implies $\vdash_{\mathcal{N}} F[A] \to F[B]$. MP applied to this and the premise $\vdash_{\mathcal{N}} F[A]$, gives $\vdash_{\mathcal{N}} F[B]$. **QED** (4.26)

7: Consistency

Lemma 4.7, and the discussion of provable equivalence above, show that for any Γ (also for $\Gamma = \varnothing$) we have $\Gamma \vdash_{\mathcal{N}} \top$, where $\top$ is an arbitrary instance of $B \to B$. The following notion indicates that the similar fact, namely $\Gamma \vdash_{\mathcal{N}} \bot$ need not always hold.

Definition 4.27 A set of formulae Γ is *consistent* iff $\Gamma \nvdash_{\mathcal{N}} \bot$.

An equivalent formulation says that Γ is consistent iff there is a formula A such that $\Gamma \nvdash_{\mathcal{N}} A$. In fact, if $\Gamma \vdash_{\mathcal{N}} A$ for all A then, in particular, $\Gamma \vdash_{\mathcal{N}} \bot$. Equivalence follows then by the next lemma.

Lemma 4.28 If $\Gamma \vdash_{\mathcal{N}} \bot$, then $\Gamma \vdash_{\mathcal{N}} A$ for all A.

Proof. (Observe how corollary 4.26 simplifies the proof.)

$1 : \Gamma \vdash_{\mathcal{N}} \neg(B \to B)$	*assumption*
$2 : \Gamma \vdash_{\mathcal{N}} B \to B$	$L.4.7$
$3 : \Gamma \vdash_{\mathcal{N}} \neg A \to (B \to B)$	$2 + L.4.12.2$
$4 : \Gamma \vdash_{\mathcal{N}} \neg(B \to B) \to \neg\neg A$	$C.4.26$ (4.22)
$5 : \Gamma \vdash_{\mathcal{N}} \neg\neg A$	$MP(1,4)$
$6 : \Gamma \vdash_{\mathcal{N}} A$	$C.4.26$ (4.21)

QED (4.28)

This lemma is the (syntactic) reason for why inconsistent sets of "assumptions" Γ are uninteresting. Given such a set, we do not need the machinery of the proof system in order to check whether something is a theorem or not – we merely have to check if the formula is well-formed. Similarly, an axiomatic system, like $\mathcal{H}$, is inconsistent if its rules and axioms allow us to derive $\vdash_{\mathcal{H}} \bot$.

Notice that the definition requires that $\bot$ is *not* derivable. In other words, to decide if Γ is consistent it does not suffice to run enough proofs and see what can be derived from Γ. One must show that, no matter what, one will never be able to derive $\bot$. This, in general, may be an infinite task requiring searching through all the proofs. If $\bot$ is derivable, we will eventually construct a proof of it, but if it is not, we will never reach any conclusion. That is, in general, consistency of a given system may be semi-decidable. (Fortunately, consistency of $\mathcal{H}$ as well as of $\mathcal{N}$ for an arbitrary Γ is decidable (as a consequence of the fact that "being a theorem" is decidable for these systems) and we will comment on this in subsection 8.1.) In some cases, the following theorem may be used to ease the process of deciding that a given Γ is (in)consistent.

Theorem 4.29 [**Compactness**] Γ is consistent iff each finite subset $\Delta \subseteq \Gamma$ is consistent.

Proof.
$\Rightarrow$ If $\Gamma \nvdash_{\mathcal{N}} \bot$ then, obviously, there is no such proof from any subset of Γ.
$\Leftarrow$ Contrapositively, assume that Γ is inconsistent. The proof of $\bot$ must be finite and, in particular, uses only a finite number of assumptions $\Delta \subseteq \Gamma$. This means that the proof $\Gamma \vdash_{\mathcal{N}} \bot$ can be carried from a finite subset Δ of Γ, i.e., $\Delta \vdash_{\mathcal{N}} \bot$. **QED** (4.29)

8: The axiomatic system of Gentzen

By now you should be convinced that it is rather cumbersome to design proofs in $\mathcal{H}$ or $\mathcal{N}$. From the mere form of the axioms and rules of these systems it is by no means clear that they define recursive sets of formulae. (As usual, it is easy to see (a bit more tedious to prove) that these sets are semi-recursive.)

We give yet another axiomatic system for PL in which proofs can be constructed mechanically. The relation $\vdash_{\mathcal{G}} \subseteq \wp(\mathsf{WFF}_{\mathsf{PL}}) \times \wp(\mathsf{WFF}_{\mathsf{PL}})$, contains expressions, called *sequents*, of the form $\Gamma \vdash_{\mathcal{G}} \Delta$, where $\Gamma, \Delta \subseteq \mathsf{WFF}_{\mathsf{PL}}$ are finite sets of formulae. It is defined inductively as follows:

Axioms :: $\Gamma \vdash_{\mathcal{G}} \Delta$, whenever $\Gamma \cap \Delta \neq \varnothing$
Rules ::

$$\neg\vdash : \frac{\Gamma \vdash_{\mathcal{G}} \Delta, A}{\Gamma, \neg A \vdash_{\mathcal{G}} \Delta} \qquad \vdash\neg : \frac{\Gamma, A \vdash_{\mathcal{G}} \Delta}{\Gamma \vdash_{\mathcal{G}} \Delta, \neg A}$$

$$\to\vdash : \frac{\Gamma \vdash_{\mathcal{G}} \Delta, A \quad ; \quad \Gamma, B \vdash_{\mathcal{G}} \Delta}{\Gamma, A \to B \vdash_{\mathcal{G}} \Delta} \qquad \vdash\to : \frac{\Gamma, A \vdash_{\mathcal{G}} \Delta, B}{\Gamma \vdash_{\mathcal{G}} \Delta, A \to B}$$

The power of the system is the same whether we allow Γ's and Δ's in the axioms to contain arbitrary formulae or only atomic ones. We comment now on the "mechanical" character of $\mathcal{G}$ and the way one can use it.

8.1: Decidability of the axiomatic systems for PL

Gentzen's system defines a set $\vdash_{\mathcal{G}} \subseteq \wp(\mathsf{WFF}_{\mathsf{PL}}) \times \wp(\mathsf{WFF}_{\mathsf{PL}})$. Unlike for $\mathcal{H}$ or $\mathcal{N}$, it is (almost) obvious that this set is recursive – we do not give a formal proof but indicate its main steps.

Theorem 4.30 Relation $\vdash_{\mathcal{G}}$ is decidable.

Proof. Given an arbitrary sequent $\Gamma \vdash_{\mathcal{G}} \Delta \;=\; A_1, ..., A_n \vdash_{\mathcal{G}} B_1, ..., B_m$, we can start processing its formulae in an arbitrary order, for instance, from left to right, by applying relevant rules *bottom-up!* For instance, $B \to A, \neg A \vdash_{\mathcal{G}} \neg B$ is shown by building the proof starting at the bottom line:

$$\begin{array}{lcc}
5: & & \underline{\quad axiom \quad} \qquad\qquad\qquad \\
4: & & \underline{B \vdash_{\mathcal{G}} A, B} \quad \underline{\quad axiom \quad} \\
3: & \vdash\neg & \underline{\vdash_{\mathcal{G}} \neg B, A, B \;;\; A \vdash_{\mathcal{G}} A, \neg B} \\
2: & \to\vdash & \underline{B \to A \vdash_{\mathcal{G}} \neg B, A} \\
1: & \neg\vdash & \neg A, B \to A \vdash_{\mathcal{G}} \neg B
\end{array}$$

In general, the proof in $\mathcal{G}$ proceeds as follows:

- If A_i is atomic, we continue with A_{1+i}, and then with B's.
- If a formula is not atomic, it is either $\neg C$ or $C \to D$. In either case there is *only one* rule which can be applied (remember, we go bottom-up). Premise(s) of this rule are uniquely determined by the conclusion (formula we are processing at the moment) and its application will remove the main connective, i.e., *reduce the number of* $\neg$*, resp.* $\to$!
- Thus, eventually, we will arrive at a sequent $\Gamma' \vdash_{\mathcal{G}} \Delta'$ which contains only atomic formulae. We then only have to check whether $\Gamma' \cap \Delta' = \varnothing$, which is obviously a decidable problem since both sets are finite.

Notice that the rule $\to\vdash$ "splits" the proof into two branches, but each of them contains fewer connectives. We have to process both branches but, again, for each we will eventually arrive at sequents with only atomic formulae. The initial sequent is derivable in $\mathcal{G}$ iff all such branches terminate with axioms. And it is not derivable iff at least one terminates with a non-axiom (i.e., $\Gamma' \vdash_{\mathcal{G}} \Delta'$ where $\Gamma' \cap \Delta' = \varnothing$). Since all branches are guaranteed to terminate $\vdash_{\mathcal{G}}$ is decidable. **QED** (4.30)

Now, notice that the expressions used in $\mathcal{N}$ are special cases of sequents, namely, the ones with exactly one formula on the right of $\vdash_{\mathcal{N}}$. If we restrict our attention in $\mathcal{G}$ to such sequents, the above theorem still tells us that the respective restriction of $\vdash_{\mathcal{G}}$ is decidable. We now indicate the main steps involved in showing that this restricted relation is the same as $\vdash_{\mathcal{N}}$. As a consequence, we obtain that $\vdash_{\mathcal{N}}$ is decidable, too. That is, we want to show that

$$\Gamma \vdash_{\mathcal{N}} B \;\text{ iff }\; \Gamma \vdash_{\mathcal{G}} B.$$

1) In Exercise 4.3, you are asked to prove a part of the implication "if $\vdash_{\mathcal{N}} B$ then $\vdash_{\mathcal{G}} B$", by showing that all axioms of $\mathcal{N}$ are derivable in $\mathcal{G}$. It is not too difficult to show that also the MP rule is admissible in $\mathcal{G}$. It is there called the (cut)-rule whose simplest form is:

$$\frac{\Gamma \vdash_{\mathcal{G}} A \quad ; \quad \Gamma, A \vdash_{\mathcal{G}} B}{\Gamma \vdash_{\mathcal{G}} B} \ (cut)$$

and MP is easily derivable from it. (If $\Gamma \vdash_{\mathcal{G}} A \to B$, then it must have been derived using the rule $\vdash\!\to$, i.e., we must have had earlier ("above") in the proof of the right premise $\Gamma, A \vdash_{\mathcal{G}} B$. Thus we could have applied (cut) at this earlier stage and obtain $\Gamma \vdash_{\mathcal{G}} B$, without bothering to derive $\Gamma \vdash_{\mathcal{G}} A \to B$ at all.)

2) To complete the proof we would have to show also the opposite implication "if $\vdash_{\mathcal{G}} B$ then $\vdash_{\mathcal{N}} B$", namely that $\mathcal{G}$ does not prove more formulae than $\mathcal{N}$ does. (If it did, the problem would be still open, since we would have a decision procedure for $\vdash_{\mathcal{G}}$ but not for $\vdash_{\mathcal{N}} \subset \vdash_{\mathcal{G}}$. I.e., for some formula $B \not\in \vdash_{\mathcal{N}}$ we might still get the positive answer, which would merely mean that $B \in \vdash_{\mathcal{G}}$.) This part of the proof is more involved since Gentzen's rules for $\neg$ do not produce $\mathcal{N}$-expressions, i.e., a proof in $\mathcal{G}$ may go through intermediary steps involving expressions not derivable (not existing, or "illegal") in $\mathcal{N}$.

3) Finally, if $\mathcal{N}$ is decidable, then lemma 4.18 implies that also $\mathcal{H}$ is decidable – according to this lemma, to decide if $\vdash_{\mathcal{H}} B$, it suffices to decide if $\vdash_{\mathcal{N}} B$.

8.2: Gentzen's rules for abbreviated connectives

The rules of Gentzen's form a very well-structured system. For each connective, $\to$, $\neg$ there are two rules – one treating its occurrence on the left, and one on the right of $\vdash_{\mathcal{G}}$. As we will soon see, it makes often things easier if one is allowed to work with some abbreviations for frequently occurring sets of symbols. For instance, assume that in the course of some proof, we run again and again in the sequence of the form $\neg A \to B$. Processing it requires application of at least two rules. One may be therefore tempted to define a new connective $A \vee B \stackrel{\text{def}}{=} \neg A \to B$, and a new rule for its treatement. In fact, in Gentzen's system we should obtain two rules for the occurrence of this new symbol on the left, resp. on the right of $\vdash_{\mathcal{G}}$. Now looking back at the original rules from the begining of this section, we can see how such a connective should be treated:

$$\dfrac{\dfrac{\dfrac{\Gamma \vdash_{\mathcal{G}} A, B, \Delta}{\Gamma, \neg A \vdash_{\mathcal{G}} B, \Delta}}{\Gamma \vdash_{\mathcal{G}} \neg A \to B, \Delta}}{\Gamma \vdash_{\mathcal{G}} A \vee B, \Delta} \qquad \dfrac{\dfrac{\dfrac{\Gamma, A \vdash_{\mathcal{G}} \Delta}{\Gamma \vdash_{\mathcal{G}} \neg A, \Delta} \quad \Gamma, B \vdash_{\mathcal{G}} \Delta}{\Gamma, \neg A \to B \vdash_{\mathcal{G}} \Delta}}{\Gamma, A \vee B \vdash_{\mathcal{G}} \Delta}$$

Abbreviating these two derivations yields the following two rules:

$$\vdash\vee \quad \frac{\Gamma \vdash_{\mathcal{G}} A, B, \Delta}{\Gamma \vdash_{\mathcal{G}} A \vee B, \Delta} \qquad \vee\vdash \quad \frac{\Gamma, A \vdash_{\mathcal{G}} \Delta \quad ; \quad \Gamma, B \vdash_{\mathcal{G}} \Delta}{\Gamma, A \vee B \vdash_{\mathcal{G}} \Delta} \tag{4.31}$$

In a similar fashion, we may construct the rules for another, very common abbreviation, $A \wedge B \stackrel{\text{def}}{=} \neg(A \to \neg B)$:

$$\vdash\wedge \quad \frac{\Gamma \vdash_{\mathcal{G}} A, \Delta \quad ; \quad \Gamma \vdash_{\mathcal{G}} B, \Delta}{\Gamma \vdash_{\mathcal{G}} A \wedge B, \Delta} \qquad \wedge\vdash \quad \frac{\Gamma, A, B \vdash_{\mathcal{G}} \Delta}{\Gamma, A \wedge B \vdash_{\mathcal{G}} \Delta} \tag{4.32}$$

It is hard to imagine how to perform a similar construction in the systems $\mathcal{H}$ or $\mathcal{N}$. We will meet the above abbreviations in the following chapters.

9: Some proof techniques

In the next chapter we will see that formulae of PL may be interpreted as propositions – statements possessing truth-value true or false. The connective $\neg$ may be then interpreted as negation of the argument proposition, while $\to$ as (a kind of) implication. With this intuition, we may recognize some of the provable facts (either formulae or admissible rules) as giving rise to particular strategies of proof which can be – and are – utilized at all levels, in fact, throughout the whole of mathematics, as well as in much of informal reasoning. Most facts from and about PL can be viewed in this way, and we give only a few most common examples.

• As a trivial example, the provable equivalence $\vdash_{\mathcal{N}} B \leftrightarrow \neg\neg B$ from (4.21), means that in order to show double negation $\neg\neg B$, it suffices to show B. One will hardly try to say "I am *not un*married." – "I am married." is both more convenient and natural.

• Let G, D stand, respectively, for the statements '$\Gamma \vdash_{\mathcal{N}} \bot$' and '$\Delta \vdash_{\mathcal{N}} \bot$ for some $\Delta \subseteq \Gamma$' from the proof of theorem 4.29. In the second point, we showed $\neg D \to \neg G$ contrapositively, i.e., by showing $G \to D$. That this is a legal and sufficient way of proving the first statement can be, at the present point, justified by appealing to (4.22) – $(A \to B) \leftrightarrow (\neg B \to \neg A)$ says precisely that proving one is the same as (equivalent to) proving the other.

• Another proof technique is expressed in corollary 4.16: $A \vdash_{\mathcal{N}} B$ iff $\vdash_{\mathcal{N}} A \to B$. Treating formulae on the left of $\vdash_{\mathcal{N}}$ as assumptions, this tells us that in order to prove that A implies B, $A \to B$, we may prove $A \vdash_{\mathcal{N}} B$, i.e., *assume* that A is true and show that then also B must be true.

• In Exercise 4.1 you are asked to show admissibility of the following rule: $\dfrac{A \vdash_{\mathcal{N}} \bot}{\vdash_{\mathcal{N}} \neg A}$. Interpreting $\bot$ as something which can never be true, a contradiction or an absurdity, this rule expresses actually *reductio ad absurdum* which we have seen in the chapter on history of logic (Zeno's argument about Achilles and tortoise): if A can be used to derive an absurdity, then A can not be true i.e., (applying the law of excluded middle) its negation must be.

Exercises 4.

EXERCISE 4.1 Prove the following statements in $\mathcal{N}$:

(1) $\vdash_{\mathcal{N}} \neg A \to (A \to B)$

(Hint: Complete the following proof:

$$\begin{array}{ll} 1: \vdash_{\mathcal{N}} \neg A \to (\neg B \to \neg A) & A1 \\ 2: \neg A \vdash_{\mathcal{N}} \neg B \to \neg A & C.4.16 \\ 3: & A3 \\ 4: & MP(2,3) \\ 5: & DT(4) \end{array}$$
)

(2) $\neg B, A \vdash_{\mathcal{N}} \neg(A \to B)$

(Hint: Start as follows:

$$\begin{array}{ll} 1: A, A \to B \vdash_{\mathcal{N}} A & A0 \\ 2: A, A \to B \vdash_{\mathcal{N}} A \to B & A0 \\ 3: A, A \to B \vdash_{\mathcal{N}} B & MP(1,2) \\ 4: A \vdash_{\mathcal{N}} (A \to B) \to B & DT(3) \\ \vdots & \end{array}$$

Apply then Lemma 4.15; you will also need Corollary 4.16.)

(3) $\vdash_{\mathcal{N}} A \to (\neg B \to \neg(A \to B))$

(4) $\vdash_{\mathcal{N}} (A \to \bot) \to \neg A$

(5) Show now admissibility in $\mathcal{N}$ of the rules

$$\text{(a)}\ \frac{\vdash_{\mathcal{N}} A \to \bot}{\vdash_{\mathcal{N}} \neg A} \qquad\qquad \text{(b)}\ \frac{A \vdash_{\mathcal{N}} \bot}{\vdash_{\mathcal{N}} \neg A}$$

(Hint: for (a) use 4 and MP, and for (b) use (a) and Deduction Theorem)

(6) Prove the first formula in $\mathcal{H}$, i.e., $(1') :\ \vdash_{\mathcal{H}} \neg A \to (A \to B)$.

EXERCISE 4.2 Show the claim (4.23), i.e., $\vdash_{\mathcal{N}} (A \to A) \leftrightarrow (B \to B)$.
(Hint: use lemma 4.7 and then lemma 4.12.2.)

EXERCISE 4.3 Consider the Gentzen's system $\mathcal{G}$ from section 8.

(1) Show that all axioms of the $\mathcal{N}$ system are derivable in $\mathcal{G}$.
(Hint: Instead of pondering over the axioms to start with, apply the bottom-up strategy from Section 8.1.)

(2) Using the same bottom-up strategy, prove the formulae 1., 2. and 3. from exercise 4.1 in $\mathcal{G}$.

EXERCISE 4.4 Lemma 4.12 generalized lemma 4.9 to the expressions involving assumptions $\Gamma \vdash_{\mathcal{N}} \ldots$ We can, however, reformulate the rules in a different way, namely, by placing the antecedents of $\rightarrow$ to the left of $\vdash_{\mathcal{N}}$. Show the admissibility in $\mathcal{N}$ of the rules:

$$1.\ \frac{\Gamma \vdash_{\mathcal{N}} B}{\Gamma, A \vdash_{\mathcal{N}} B} \qquad\qquad 2.\ \frac{\Gamma, A \vdash_{\mathcal{N}} B \ \ ; \ \ \Gamma, B \vdash_{\mathcal{N}} C}{\Gamma, A \vdash_{\mathcal{N}} C}$$

(1. must be shown directly by induction on the length of the proof of $\Gamma \vdash_{\mathcal{N}} B$, without using corollary 4.16 – why? For 2. you can then use 4.16.)

EXERCISE 4.5 Show that the following definition of consistency is equivalent to 4.27:

Γ is consistent iff there is no formula A such that both $\Gamma \vdash A$ and $\Gamma \vdash \neg A$.

Hint: You should show that for arbitrary Γ one has that:

$\Gamma \not\vdash_{\mathcal{N}} \bot$ iff for no $A : \Gamma \vdash_{\mathcal{N}} A$ and $\Gamma \vdash_{\mathcal{N}} \neg A$, which is the same as showing that:
$\Gamma \vdash_{\mathcal{N}} \bot \ \Leftrightarrow$ for some $A : \Gamma \vdash_{\mathcal{N}} A$ and $\Gamma \vdash_{\mathcal{N}} \neg A$.

The implication $\Rightarrow$) follows easily from the assumption $\Gamma \vdash_{\mathcal{N}} \neg(B \rightarrow B)$ and lemma 4.7. For the opposite one start as follows (use corollary 4.26 on 3: and then MP):

$1 : \Gamma \vdash_{\mathcal{N}} A$ — *ass.*
$2 : \Gamma \vdash_{\mathcal{N}} \neg A$ — *ass.*
$3 : \Gamma \vdash_{\mathcal{N}} \neg\neg(A \rightarrow A) \rightarrow \neg A$ — $L.4.12.2$ (2)
$\vdots$

Chapter 5
Semantics of PL

- Semantics of PL
- Semantic properties of formulae
- Abbreviations
- Propositions, Sets and Boolean Algebras

In this chapter we are leaving the proofs and axioms from the previous chapter aside. For the time being, *none* of the concepts and discussion below should be referred to any earlier results on axiomatic systems. (Connection to the proof systems will be studied in the following chapters.) We are now studying exclusively *the language* of PL – definition 4.4 – and the standard way of assigning *meaning* to its expressions.

1: Semantics of PL

◊—— a Background Story ——◊

There is a huge field of Proof Theory which studies axiomatic systems *per se*, i.e., without reference to their possible meanings. This was the kind of study we were carrying out in the preceding chapter. As we emphasised at the begining of that chapter, an axiomatic system may be given very different interpretations and we will in this chapter see a few possibilities for interpreting the system of Statement Logic. Yet, axiomatic systems are typically introduced for the purpose of studying particular areas of interest or particular phenomena therein. They provide *syntactic* means for such a study: a language which is used to refer to various objects of some domain and a proof calculus which, hopefully, captures some of the essentail relationships between various aspects of the domain.

As you should have gathered from the presentation of the history of logic, its original intention was to capture the patterns of correct reasoning which we otherwise carry out in natural language. Statement Logic, in particular, was conceived as a logic of statements or propositions: propositional variables may be interpreted as arbitrary statements, while the connectives as the means of constructing new statements from others. Thus, for instance, lets make the following reasoning:

	`If it is raining, we will go to cinema.`
and	`If we go to cinema, we will see a Kurosawa film.`
hence	`If it is raining, we will see a Kurosawa film.`

If we agree to represent the implication `if ... then ...` by the syntactic symbol $\rightarrow$, this reasoning is represented by interpreting A as `It will rain`, B as `We will go to cinema`, C as `We will see a Kurosawa film` and by the deduction $\dfrac{A \rightarrow B \quad ; \quad B \rightarrow C}{A \rightarrow C}$. As we have seen in lemma 4.9, this is a valid rule in the system $\vdash_{\mathcal{H}}$. Thus, we might say that the system $\vdash_{\mathcal{H}}$ (as well as $\vdash_{\mathcal{N}}$) captures this aspect of our natural reasoning.

However, one has to be very – indeed, extremely – careful with this kinds of analogies. They are never complete and any formal system runs, sooner or later, into problems when confronted with the richness and sophistication of natural language. Consider the following argument:

	`If I am in Paris then I am in France.`
and	`If I am in Rome then I am in Italy.`
hence	`If I am in Paris then I am in Italy or else`
	`if I am in Rome then I am in France.`

It does not look plausible, does it? Now, let us translate it into statement logic: P for being in Paris, F for being in France, R in Rome and I in Italy. Using Gentzen's rules with the standard reading of $\wedge$ as '`and`' and $\vee$ as '`or`', we obtain:

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{R \rightarrow I, P, R \vdash_{\mathcal{G}} I, F, P \quad ; \quad F, R \rightarrow I, P, R \vdash_{\mathcal{G}} I, F}{P \rightarrow F, R \rightarrow I, P, R \vdash_{\mathcal{G}} I, F}}{P \rightarrow F, R \rightarrow I, P \vdash_{\mathcal{G}} I, R \rightarrow F}}{P \rightarrow F, R \rightarrow I \vdash_{\mathcal{G}} P \rightarrow I, R \rightarrow F}}{P \rightarrow F, R \rightarrow I \vdash_{\mathcal{G}} P \rightarrow I \vee R \rightarrow F}}{P \rightarrow F \wedge R \rightarrow I \vdash_{\mathcal{G}} P \rightarrow I \vee R \rightarrow F}}{\vdash_{\mathcal{G}} (P \rightarrow F \wedge R \rightarrow I) \rightarrow (P \rightarrow I \vee R \rightarrow F)}
$$

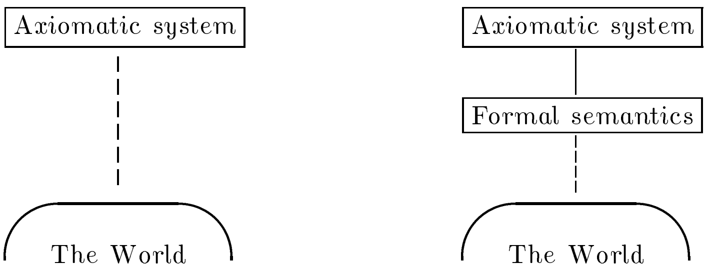
Our argument – the implication from ($P \rightarrow F$ and $R \rightarrow I$) to ($P \rightarrow I$ or $R \rightarrow F$) turns out to be provable in $\vdash_{\mathcal{G}}$. (It is so in the other systems as well.) Logicians happen to have an answer to this particular problem (we will return to it in exercise 6.1). But there are other strange things which cannot be easily answered. Typically, any formal system attempting to capture some area of discourse, will capture only *some part* of it. Attempting to apply it beyond this area, leads inevitably to counterintuitive phenomena.

Statement logic attempts to capture some simple patterns of reasoning at the level of propositions. A proposition can be thought of as a declarative sentence which may be assigned a unique truth value. The sentence "It is raining" is either true or false. Thus, the intended and possible *meanings* of propositions are truth values: true or false. Now, the meaning of the proposition `If it rains, we will go to a cinema`, $A \to B$, can be construed as: *if* 'it is true that it will rain' *then* 'it is true that we will go to a cinema'. The implication $A \to B$ says that if A is true then B must be true as well.

Now, since this implication is itself a proposition, it will have to be given a truth value as its meaning. And this truth value will depend on the truth value of its constituents: the propositions A and B. If A is true (it is raining) but B is false (we are not going to a cincema), the whole implication $A \to B$ is false.

And now comes the question: what if A is false? Did the implication $A \to B$ assert anything about this situation? No, it did not. If A is false (it is not raining), we may go to a cinema or we may stay at home – I haven't said anything about that case. Yet, the proposition *has to* have a meaning for all possible values of its parts. In this case – when the antecedent A is false – the whole implication $A \to B$ is declared true irrespectively of the truth value of B. You should notice that here something special is happening which does not necessarily correspond so closely to our intuition. And indeed, it is something very strange! `If I am a woman, then you are Dalai Lama.` Since I am not a woman, the implication happens to be true! But, as you know, this does not mean that you are Dalai Lama. This example, too, can be explained by the same argument as the above one (to be indicated in exercise 6.1). However, the following implication is true, too, and there is no formal way of excusing it being so or explaining it away: `If it is not true that when I am a man then I am a man, then you are Dalai Lama`, $\neg(M \to M) \to D$. It is correct, it is true and ... it seems to be entirely meaningless.

In short, formal correctness and accuracy does not always correspond to something meaningful in natural language, even if such a correspondance was the original motivation. A possible discrepancy indicated above concerned, primarily, the discrepancy between our intuition about the meaning of sentences and their representation in a syntactic system. But the same problem occurs at yet another level – the same or analogous discrepancies occur between our intuitive understanding of the world and the formal *semantic* model of the world. Thinking about axiomatic systems as tools for modelling the world, we might be tempted to look at the relation as illustrated on the left side of the following figure: an axiomatic system modelling the world. In truth, however, the relation is more complicated as illustrated on the right of the figure.



An axiomatic system *never* addresses the world directly. What it addresses is a possible semantic model which tries to give a formal representation of the world. As we have mentioned several times, a given axiomatic system may be given various interpretations – all such interpretations will be possible formal semantic models of the system. To what extent these semantic models capture our intuition about the world is a different question. It is the question about "correctness" or "incorrectness" of modelling – an axiomatic system in itself is neither, because it can be endowed with different interpretations. The problems indicated above were really the problems with the semantic model of natural language which was implicitly introduced by assuming that statements are to be interpreted as truth values.

We will now endavour to study the semantics – meaning – of the syntactic expessions from PL. We will see some alternative semantics starting with the standard one based on the so called "truth functions" (which we will call "boolean functions"). To avoid confusion and surprises, one should always keep in mind that we are not talking about the world but are defining a *formal* model of PL which, at best, can *provide an imperfect link between the syntax of* PL *and the world. The formality of the model, as always, will introduce some discrepancies as those described above and many things may turn out not exactly as we would expect them to be in the real world.*

◇———◇

Let $\mathbf{B}$ be a set with two elements. *Any* such set would do but, for convenience, we will typically let $\mathbf{B} = \{\mathbf{1}, \mathbf{0}\}$. Whenever one tries to capture the meaning of propositions as their truth value, and uses Statement Logic with this intention, one interprets $\mathbf{B}$ as the set $\{\mathsf{true}, \mathsf{false}\}$. Since this gives too strong associations and leads often to incorrect intuitions without improving anything, we will avoid the words true and false. Instead we will talk about "boolean values" ($\mathbf{1}$ and $\mathbf{0}$) and "boolean functions". If the word "truth" appears, it may be safely replaced with "boolean".

For any $n \geq 0$ we can imagine various functions mapping $\mathbf{B}^n \to \mathbf{B}$. For instance, for $n = 2$, a function $f : \mathbf{B} \times \mathbf{B} \to \mathbf{B}$ can be defined by $f(\mathbf{1}, \mathbf{1}) \stackrel{\text{def}}{=} \mathbf{1}$, $f(\mathbf{1}, \mathbf{0}) \stackrel{\text{def}}{=} \mathbf{0}$, $f(\mathbf{0}, \mathbf{1}) \stackrel{\text{def}}{=} \mathbf{1}$ and $f(\mathbf{0}, \mathbf{0}) \stackrel{\text{def}}{=} \mathbf{1}$. It can be written more concisely as the table:

x	y	$f(x,y)$
$\mathbf{1}$	$\mathbf{1}$	$\mathbf{1}$
$\mathbf{1}$	$\mathbf{0}$	$\mathbf{0}$
$\mathbf{0}$	$\mathbf{1}$	$\mathbf{1}$
$\mathbf{0}$	$\mathbf{0}$	$\mathbf{1}$

The first n-columns contain all the possible combinations of the arguments (giving 2^n distinct rows), and the last column specifies the value of the function for this combination of the arguments. For each of the 2^n rows a function takes one of the two possible values, so for any n there are exactly 2^{2^n} different functions $\mathbf{B}^n \to \mathbf{B}$. For $n = 0$, there are only two (constant) functions, for $n = 1$ there will be four distinct functions (which ones?) and so on.

Surprisingly (or not), the language of PL is designed exactly for describing such functions!

Definition 5.1 An PL *structure* consists of:

(1) A domain with two elements, called *boolean values*, $\mathbf{B} = \{\mathbf{1}, \mathbf{0}\}$

(2) Interpretation of the connectives, $\underline{\neg} : \mathbf{B} \to \mathbf{B}$ and $\underline{\to} : \mathbf{B}^2 \to \mathbf{B}$, given by

the boolean tables:[4]

x	$\underline{\neg}\ x$
1	**0**
0	**1**

x	y	$x \underline{\to} y$
1	**1**	**1**
1	**0**	**0**
0	**1**	**1**
0	**0**	**1**

Given an alphabet Σ, an PL structure for Σ is an PL structure with

(3) an assignment of boolean values to all propositional variables, i.e., a function $V : \Sigma \to \{\mathbf{1}, \mathbf{0}\}$. (Such a V is also called a *valuation* of Σ.)

Thus connectives are interpreted as functions on the set $\{\mathbf{1}, \mathbf{0}\}$. To distinguish the two, we use the simple symbols $\neg$ and $\to$ when talking about syntax, and the underlined ones $\underline{\neg}$ and $\underline{\to}$ when we are talking about the semantic interpretation as boolean functions. $\neg$ is interpreted as the function $\underline{\neg} : \{\mathbf{1}, \mathbf{0}\} \to \{\mathbf{1}, \mathbf{0}\}$, defined by $\underline{\neg}(\mathbf{1}) \stackrel{\text{def}}{=} \mathbf{0}$ and $\underline{\neg}(\mathbf{0}) \stackrel{\text{def}}{=} \mathbf{1}$. $\to$ is binary and represents one of the functions from $\{\mathbf{1}, \mathbf{0}\}^2$ into $\{\mathbf{1}, \mathbf{0}\}$.

Example 5.2

Let $\Sigma = \{a, b\}$. $V = \{a \mapsto \mathbf{1}, b \mapsto \mathbf{1}\}$ is a Σ-structure (i.e., a structure interpreting all symbols from Σ) assigning $\mathbf{1}$ (true) to both variables. $V = \{a \mapsto \mathbf{1}, b \mapsto \mathbf{0}\}$ is another Σ-structure.

Let $\Sigma = \{$'John smokes', 'Mary sings'$\}$. Here 'John smokes' is a propositional variable (with a rather lengthy name). $V = \{$'John smokes' $\mapsto \mathbf{1}$, 'Mary sings' $\mapsto \mathbf{0}\}$ is a Σ-structure in which both "John smokes" and "Mary does not sing". □

The domain of interpretation has two boolean values $\mathbf{1}$ and $\mathbf{0}$, and so we can imagine various functions, in addition to those interpreting the connectives. As remarked above, for arbitrary $n \geq 0$ there are 2^{2^n} distinc functions mapping $\{\mathbf{1}, \mathbf{0}\}^n$ into $\{\mathbf{1}, \mathbf{0}\}$.

Example 5.3

Here is an example of a (somewhat involved) boolean function F :

[4] The standard name for such tables is "truth tables" but since we are trying not to misuse the word "truth", we stay consistent by replacing it here, too, with "boolean".

$\{\mathbf{1},\mathbf{0}\}^3 \to \{\mathbf{1},\mathbf{0}\}$

x	y	z	$F(x,y,z)$
1	**1**	**1**	**1**
1	**1**	**0**	**1**
1	**0**	**1**	**1**
1	**0**	**0**	**0**
0	**1**	**1**	**1**
0	**1**	**0**	**1**
0	**0**	**1**	**1**
0	**0**	**0**	**1**

□

Notice that in Definition 5.1 only the valuation differs from structure to structure. The interpretation of the connectives is always the same – for any Σ, it is fixed once and for all as the specific boolean functions. Hence, given a valuation V, there is a canonical way of extending it to the interpretation of all formulae – a valuation of propositional variables induces a valuation of all well-formed formulae. We sometimes write $\widehat{V}$ for this extended valuation. This is given in the following definition which, intuitively, corresponds to the fact that if we know that 'John smokes' and 'Mary does not sing', then we also know that 'John smokes *and* Mary does not sing', or else that it is not true that 'John does not smoke'.

Definition 5.4 Any valuation $V : \Sigma \to \{\mathbf{1},\mathbf{0}\}$ induces a *unique* valuation $\widehat{V} : \mathsf{WFF}^{\Sigma}_{\mathsf{PL}} \to \{\mathbf{1},\mathbf{0}\}$ as follows:

1. for $A \in \Sigma$: $\widehat{V}(A) = V(A)$
2. for $A = \neg B$: $\widehat{V}(A) = \underline{\neg}(\widehat{V}(B))$
3. for $A = (B \to C)$: $\widehat{V}(A) = \widehat{V}(B) \underline{\to} \widehat{V}(C)$

For the purposes of this section it is convenient to assume that some total ordering has been selected for the propositional variables, so that for instance a "comes before" b, which again "comes before" c.

Example 5.5

Given the alphabet $\Sigma = \{a, b, c\}$, we use the fixed interpretation of the connectives to determine the boolean value of, for instance, $a \to (\neg b \to c)$

as follows:

a	b	c	$\underline{\neg}\ b$	$\underline{\neg} b \underline{\to} c$	$a \underline{\to} (\underline{\neg} b \underline{\to} c)$
1	**1**	**1**	**0**	**1**	**1**
1	**1**	**0**	**0**	**1**	**1**
1	**0**	**1**	**1**	**1**	**1**
1	**0**	**0**	**1**	**0**	**0**
0	**1**	**1**	**0**	**1**	**1**
0	**1**	**0**	**0**	**1**	**1**
0	**0**	**1**	**1**	**1**	**1**
0	**0**	**0**	**1**	**0**	**1**

□

Ignoring the intermediary columns, this table displays exactly the same dependence of the entries in the last column on the entries in the first three ones as the function F from Example 5.3. We say that the formula $a \to (\neg b \to c)$ *determines* the function F. The general definition is given below.

Definition 5.6 For any formula B, let $\{b_1, \ldots, b_n\}$ be the propositional variables in B, listed in increasing order. Each assignment $V : \{b_1, \ldots, b_n\} \to \{\mathbf{1}, \mathbf{0}\}$ determines a unique boolean value $\widehat{V}(B)$. Hence, each formula B determines a function $\underline{B} : \{\mathbf{1}, \mathbf{0}\}^n \to \{\mathbf{1}, \mathbf{0}\}$, given by the equation

$$\underline{B}(x_1, \ldots, x_n) = \{b_1 \mapsto \widehat{x_1, \ldots, b_n} \mapsto x_n\}(B).$$

Example 5.7
Suppose a and b are in Σ, and a comes before b in the ordering. Then $(a \to b)$ determines the function $\underline{\to}$, while $(b \to a)$ determines the function $\underline{\leftarrow}$ with the boolean table shown below.

x	y	$x \underline{\to} y$	$x \underline{\leftarrow} y$
1	**1**	**1**	**1**
1	**0**	**0**	**1**
0	**1**	**1**	**0**
0	**0**	**1**	**1**

□

Observe that although for a given n there are exactly 2^{2^n} boolean functions, there are infinitely many formulae over n propositional variables. Thus, different formulae will often determine the same boolean function. Deciding which formulae determine the same functions is an important problem which we will soon encounter.

2: Semantic properties of formulae

Formula determines a boolean function and we now list some semantic properties of formulae, i.e., properties which are actually the properties of such induced functions.

Definition 5.8 Let $A, B \in \mathsf{WFF}_{\mathsf{PL}}$, and V be a valuation.

A is	iff	condition holds	notation:
satisfied in V	iff	$\widehat{V}(A) = \mathbf{1}$	$V \models A$
not satisfied in V	iff	$\widehat{V}(A) = \mathbf{0}$	$V \not\models A$
valid/tautology	iff	for all $V : V \models A$	$\models A$
falsifiable	iff	there is a $V : V \not\models A$	$\not\models A$
satisfiable	iff	there is a $V : V \models A$	
unsatisfiable/contradiction	iff	for all $V : V \not\models A$	
(tauto)logical consequence of B	iff	$B \to A$ is valid	$B \Rightarrow A$
(tauto)logically equivalent to B	iff	$A \Rightarrow B$ and $B \Rightarrow A$	$A \Leftrightarrow B$

If A is satisfied in V, we say that V *satisfies* A. Otherwise V *falsifies* A. (Sometimes, one also says that A is valid *in* V, when A is satisfied in V. But notice that validity of A *in* V does not mean or imply that A is valid (in general), only that it is satisfiable.) Valid formulae – those satisfied in all structures – are also called *tautologies* and the unsatisfiable ones *contradictions*. Those which are both falsifiable and satisfiable, i.e., which are neither tautologies nor contradictions, are called *contingent*. A valuation which satisfies a formula A is called a *model of* A.

Sets of formulae are sometimes called *theories*. Many of the properties defined for formulae are defined for theories as well. Thus a valuation is said to satisfy a theory iff it satisfies every formula in the theory. Such a valuation is also said to be a model of the theory. The class of all models of a given theory Γ is denoted $Mod(\Gamma)$. Like a single formula, a set of formulae Γ is satisfiable iff it has a model, i.e., iff $Mod(\Gamma) \neq \varnothing$.

Example 5.9
$a \to b$ is not a tautology – assign $V(a) = \mathbf{1}$ and $V(b) = \mathbf{0}$. Hence $a \Rightarrow b$ does not hold. However, it is satisfiable, since it is true, for instance, under the valuation $\{a \mapsto \mathbf{1}, b \mapsto \mathbf{1}\}$. The formula is contingent.

$B \to B$ evaluates to $\mathbf{1}$ for any valuation (and any $B \in \mathsf{WFF}_{\mathsf{PL}}$), and so

$B \Rightarrow B$. As a last example, we have that $B \Leftrightarrow \neg\neg B$.

B	B	$B \to B$
0	**0**	**1**
1	**1**	**1**

B	$\neg B$	$\neg\neg B$	$B \to \neg\neg B$	*and*	$\neg\neg B \to B$
1	**0**	**1**	**1**	*and*	**1**
0	**1**	**0**	**1**	*and*	**1**

□

The operators $\Rightarrow$ and $\Leftrightarrow$ are *meta*-connectives stating that a corresponding relation ($\to$ and $\leftrightarrow$, respectively) between the two formulae holds *for all* boolean assignments. These operators are therefore used only at the outermost level, like for $A \Rightarrow B$ – we avoid something like $A \Leftrightarrow (A \Rightarrow B)$ or $A \to (A \Leftrightarrow B)$.

Fact 5.10 We have the obvious relations between the sets of Sat(isfiable), Fal(sifiable), Taut(ological), Contr(adictory) and All formulae:

- $Contr \subset Fal$
- $Taut \subset Sat$
- $Fal \cap Sat \neq \varnothing$
- $All = Taut \cup Contr \cup (Fal \cap Sat)$

3: Abbreviations

Intuitively, $\neg$ is supposed to express negation and we read $\neg B$ as "not B". $\to$ corresponds to implication: $A \to B$ is similar to "if A then B". These formal symbols and their semantics are not exact counterparts of the natural language expressions but they do try to *mimic* the latter as far as possible. In natural language there are several other connectives but, as we will see in chapter 5, the two we have introduced for PL are all that is needed. We will, however, try to make our formulae shorter – and more readable – by using the following abbreviations:

Definition 5.11 We define the following abbreviations:

- $A \vee B \stackrel{\text{def}}{=} \neg A \to B$, read as "$A$ or B"
- $A \wedge B \stackrel{\text{def}}{=} \neg(A \to \neg B)$, read as "$A$ and B"
- $A \leftrightarrow B \stackrel{\text{def}}{=} (A \to B) \wedge (B \to A)$, read as "$A$ if and only if B"
  (!Not to be confused with the provable equivalence from definition 4.20!)

Example 5.12

Some intuitive justification for the reading of these abbreviations comes from the boolean tables for the functions they denote. For instance, the table for $\underline{\wedge}$ will be constructed according to its definition:

x	y	$\underline{\neg}\ y$	$x \underline{\rightarrow} \underline{\neg} y$	$x \underline{\wedge} y = \underline{\neg}\ (x \underline{\rightarrow} \underline{\neg} y)$
1	**1**	**0**	**0**	**1**
1	**0**	**1**	**1**	**0**
0	**1**	**0**	**1**	**0**
0	**0**	**1**	**1**	**0**

Thus $A \wedge B$ evaluates to **1** (true) iff both components are true. (In exercise 5.1 you are asked to do the analogous thing for $\vee$.) $\square$

4: Sets and Propositions

We have defined semantics of PL by interpreting the connectives as functions over **B**. Some consequences in form of the laws which follow from this definition are listed in subsection 4.1. In subsection 4.2 we observe close relationship to the laws obeyed by the set operations and then define an altenative semantics of the language of PL based on set-interpretation. Finally, in subsection 4.3, we gather these similarities in the common concept of boolean algebra.

4.1: Laws

The definitions of semantics of the connectives together with the introduced abbreviations entitle us to conclude validity of some laws for PL.

1. Idempotency

$$A \vee A \Leftrightarrow A$$
$$A \wedge A \Leftrightarrow A$$

2. Associativity

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

3. Commutativity

$$A \vee B \Leftrightarrow B \vee A$$
$$A \wedge B \Leftrightarrow B \wedge A$$

4. Distributivity

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

5. de Morgan

$$\neg(A \vee B) \Leftrightarrow \neg A \wedge \neg B$$
$$\neg(A \wedge B) \Leftrightarrow \neg A \vee \neg B$$

6. Conditional

$$A \rightarrow B \Leftrightarrow \neg A \vee B$$
$$A \rightarrow B \Leftrightarrow \neg B \rightarrow \neg A$$

For instance, idempotency of $\wedge$ is verified directly from the definition of $\wedge$, as follows:

A	$\underline{\neg} A$	$A \underline{\rightarrow} \underline{\neg} A$	$\underline{\neg}(A \underline{\rightarrow} \underline{\neg} A) \stackrel{\text{def}}{=} A \underline{\wedge} A$
1	**0**	**0**	**1**
0	**1**	**1**	**0**

The other laws can (and should) be verified in a similar manner. $A \Leftrightarrow B$ means that for all valuations (of the propositional variables occurring in A and B) the truth values of both formulae are the same. This means almost that they determine the same function, with one restriction which is discussed in exercise 5.10.

For any A, B, C the two formulae $(A \wedge B) \wedge C$ and $A \wedge (B \wedge C)$ are distinct. However, as they are tautologically equivalent it is not always a very urgent matter to distinguish between them. In general, there are a great many ways to insert the missing parentheses in an expression like $A_1 \wedge A_2 \wedge \ldots \wedge A_n$, but since they all yield equivalent formulae we usually do not care where these parentheses go. Hence for a sequence $A_1, A_2, \ldots, A_n$ of formulae we may just talk about their *conjunction* and mean any formula obtained by supplying missing parentheses to the expression $A_1 \wedge A_2 \wedge \ldots \wedge A_n$. Analogously, the *disjunction* of $A_1, A_2, \ldots, A_n$ is any formula obtained by supplying missing parentheses to the expression $A_1 \vee A_2 \vee \ldots \vee A_n$.

Moreover, the laws of commutativity and idempotency tell us that order and repetition don't matter either. Hence we may talk about the conjunction of the formulae in some finite set, and mean any conjunction formed by the elements in some order or other. Similarly for disjunction.

The elements $A_1, \ldots, A_n$ of a conjunction $A_1 \wedge \ldots \wedge A_n$ are called the *conjuncts*. The term *disjunct* is used analogously.

4.2: Sets and PL

Compare the set laws 1.– 5. from page 1 with the tautological equivalences from the previous subsection. It is easy to see that they have "corresponding form" and can be obtained from each other by the following translations.

set − expression	−	statement
set variable a, b...	−	propositional variable a, b...
	−	$\neg$
$\cap$	−	$\wedge$
$\cup$	−	$\vee$
$=$	−	$\Leftrightarrow$
One also translates :		
U	−	$\top$
$\varnothing$	−	$\bot$

Remark 5.13 **[Formula- vs. set-operations]**
Although there is some sense of connection between the subset $\subseteq$ and implication

$\to$, the two have very different function. The latter allows us to construct new propositions. The former, $\subseteq$, is not however a set building operation: $A \subseteq B$ does not denote a (new) set but states a relation between two sets. The consistency principles are not translated because they are not so much laws as definitions introducing a new relation $\subseteq$ which holds only under the specified conditions. In order to find a set operation corresponding to $\to$, we should reformulate the syntactic definiton 5.11 and verify that $A \to B \Leftrightarrow \neg A \vee B$. The corresponding set-building operation $\triangleright$, would be then defined by $A \triangleright B \stackrel{\text{def}}{=} \overline{A} \cup B$.

We do not have a propositional counterpart of the set minus $\setminus$ operation and so the second complement law $A \setminus B = A \cap \overline{B}$ has no propositional form. However, this law says that in the propositional case we can merely use the expression $A \wedge \neg B$ corresponding to $A \cap B'$. We may translate the remaining set laws, e.g., $A \cap \overline{A} = \varnothing$ as $A \wedge \neg A \Leftrightarrow \bot$, etc. Using definition of $\wedge$, we then get $\bot \Leftrightarrow A \wedge \neg A \stackrel{3}{\Leftrightarrow} \neg A \wedge A \Leftrightarrow \neg(\neg A \to \neg A)$, which is an instance of the formula $\neg(B \to B)$.

Let us see if we can discover the reason for this exact match of laws. For the time being let us ignore the superficial differences of syntax, and settle for the logical symbols on the right. Expressions built up from Σ with the use of these, we call *boolean expressions*, BE^Σ. As an alternative to a valuation $V : \Sigma \to \{\mathbf{0}, \mathbf{1}\}$ we may consider a *set-valuation* $SV : \Sigma \to \wp(U)$, where U is any non-empty set. Thus, instead of the boolean-value semantics in the set $\mathbf{B}$, we are defining a set-valued semantics in an arbitrary set U. Such SV can be extended to $\widehat{SV} : \mathsf{BE}^\Sigma \to \wp(U)$ according to the rules

$$
\begin{array}{rcll}
\widehat{SV}(a) & = & SV(a) & \text{for all } a \in \Sigma \\
\widehat{SV}(\top) & = & U & \\
\widehat{SV}(\bot) & = & \varnothing & \\
\widehat{SV}(\neg A) & = & U \setminus \widehat{SV}(A) & \\
\widehat{SV}(A \wedge B) & = & \widehat{SV}(A) \cap \widehat{SV}(B) & \\
\widehat{SV}(A \vee B) & = & \widehat{SV}(A) \cup \widehat{SV}(B) &
\end{array}
$$

Lemma 5.14 Let $x \in U$ be arbitrary, $V : \Sigma \to \{\mathbf{1}, \mathbf{0}\}$ and $SV : \Sigma \to \wp(U)$ be such that for all $a \in \Sigma$ we have $x \in SV(a)$ iff $V(a) = \mathbf{1}$. Then for all $A \in \mathsf{BE}^\Sigma$ we have $x \in \widehat{SV}(A)$ iff $\widehat{V}(A) = \mathbf{1}$.

Proof. By induction on the complexity of A. Everything follows from the boolean-tables of $\top, \bot, \neg, \wedge, \vee$ and the observations below.

$$
\begin{array}{rcl}
x \in U & & \text{always} \\
x \in \varnothing & & \text{never} \\
x \in \overline{P} & \text{iff} & x \notin P \\
x \in P \cap Q & \text{iff} & x \in P \text{ and } x \in Q \\
x \in P \cup Q & \text{iff} & x \in P \text{ or } x \in Q
\end{array}
$$

QED (5.14)

Example 5.15
Let $\Sigma = \{a, b, c\}$, $U = \{4, 5, 6, 7\}$ and choose $x \in U$ to be 4. The upper part of the table shows an example of a valuation and set-valuation satisfying the conditions of the lemma, and the lower part the values of some formulae (boolean expressions) under these valuations.

$$
\begin{array}{cccl}
\{\mathbf{1},\mathbf{0}\} & \stackrel{V}{\leftarrow} & \Sigma & \stackrel{SV}{\rightarrow} \wp(\{4,5,6,7\}) \\
\hline
\mathbf{1} & \leftarrow & a & \rightarrow \{\mathbf{4},5\} \\
\mathbf{1} & \leftarrow & b & \rightarrow \{\mathbf{4},6\} \\
\mathbf{0} & \leftarrow & c & \rightarrow \{5,7\} \\
\hline
\{\mathbf{1},\mathbf{0}\} & \stackrel{\widehat{V}}{\leftarrow} & \mathsf{BE}^{\mathsf{PL}} & \stackrel{\widehat{SV}}{\rightarrow} \wp(\{4,5,6,7\}) \\
\hline
\mathbf{1} & \leftarrow & a \wedge b & \rightarrow \{\mathbf{4}\} \\
\mathbf{0} & \leftarrow & \neg a & \rightarrow \{6,7\} \\
\mathbf{1} & \leftarrow & a \vee c & \rightarrow \{\mathbf{4},5,7\} \\
\mathbf{0} & \leftarrow & \neg(a \vee c) & \rightarrow \{6\}
\end{array}
$$

The four formulae illustrate the general fact that, for any $A \in \mathsf{BE}^{\mathsf{PL}}$ we have $\widehat{V}(A) = \mathbf{1} \Leftrightarrow 4 \in \widehat{SV}(A)$. □

The set identities on page 1 say that the BE's on each side of an identity are interpreted identically by any set-valuation. Hence the corollary below expresses the correspondence between the set-identities and tautological equivalences.

Corollary 5.16 Let $A, B \in \mathsf{BE}^{\Sigma}$. Then
($\widehat{SV}(A) = \widehat{SV}(B)$ for all set-valuations SV, into all sets U) iff $A \Leftrightarrow B$.

Proof. The idea is to show that for every set-valuation that interprets A and B differently, there is some valuation that interprets them differently, and conversely.
$\Leftarrow$) First suppose $\widehat{SV}(A) \neq \widehat{SV}(B)$. Then there is some $x \in U$ that is contained in one but not the other. Let V_x be the valuation such that for all $a \in \Sigma$,

$$V_x(a) = \mathbf{1} \text{ iff } x \in SV(a).$$

Then $\widehat{V_x}(A) \neq \widehat{V_x}(B)$ follows from lemma 5.14.
$\Rightarrow$) Now suppose $\widehat{V}(A) \neq \widehat{V}(B)$. Let SV be the set-valuation into $\wp(\{\mathbf{1}\}) = \{\varnothing, \{\mathbf{1}\}\}$ such that for all $a \in \Sigma$,

$$\mathbf{1} \in SV(a) \text{ iff } V(a) = \mathbf{1}.$$

Again lemma 5.14 applies, and $\widehat{SV}(A) \neq \widehat{SV}(B)$ follows. **QED** (5.16)

This corollary provides an explanation for the validity of essentially the same laws for statement logic and for sets. These laws were universal, i.e., they stated equality of some set expressions *for all possible sets* and, on the other hand, logical equivalence of corresponding logical formulae *for all possible valuations*. We can now rewrite any valid equality $A = B$ between set expressions as $\overline{A} \Leftrightarrow \overline{B}$, where primed symbols indicate the corresponding logical formulae; and vice versa. Corollary says that one is valid if and only if the other one is.

Let us reflect briefly over this result which is quite significant. For the first, observe that the semantics with which we started, namely, the one interpreting connectives and formulae over the set $\mathbf{B}$, turns out to be a special case of the set-based semantics. We said that $\mathbf{B}$ may be an arbitrary two-element set. Now, take $U = \{\bullet\}$; then $\wp(U) = \{\varnothing, \{\bullet\}\}$ has two elements. Using $\bullet$ as the "designated" element x (x from lemma 5.14), the set-based semantics over this set will coincide with the propositional semantics which identifies $\varnothing$ with $\mathbf{0}$ and $\{\bullet\}$ with $\mathbf{1}$. Reinterpreting corollary with this in mind, i.e., substituting $\wp(\{\bullet\})$ for $\mathbf{B}$, tells us that $A = B$ is valid (in all possible $\wp(U)$ for all possible assignments) iff it is valid in $\wp(\{\bullet\})$! In other words, to check if some set equality holds under all possible interpretations of the involved set variables, it is enough to check if it holds under all possible interpretations of these variables in the structure $\wp(\{\bullet\})$. (One says that this structure is a *cannonical representative* of all such set-based interpretations of propositional logic.) We have thus reduced a problem which might seem to involve infinitely many possibilities (all possible sets standing for each variable), to a simple task of checking the solutions with substituting only $\{\bullet\}$ or $\varnothing$ for the inolved variables.

4.3: BOOLEAN ALGEBRAS ....[optional]

The discussion in subsection 4.2 shows the concrete connection between the set interpretation and the standard interpretation of the language of PL. The fact that both set operations and (functions interpreting the) propositional connectives obey essentially the same laws can be, however, stated more abstractly – they are both examples of yet other, general structures called *"boolean algebras"*.

Definition 5.17 The language of boolean algebra is given by 1) the set of *boolean expressions*, BE^Σ, relatively to a given alphabet Σ of variables:

BASIS :: $0, 1 \in \mathsf{BE}^\Sigma$ and $\Sigma \subset \mathsf{BE}^\Sigma$
IND. :: If $t \in \mathsf{BE}^\Sigma$ then $-t \in \mathsf{BE}^\Sigma$
:: If $s, t \in \mathsf{BE}^\Sigma$ then $(s + t) \in \mathsf{BE}^\Sigma$ and $(s * t) \in \mathsf{BE}^\Sigma$

and by 2) the formulae which are equations $s \equiv t$ where $s, t \in \mathsf{BE}^{\Sigma}$.

A *boolean algebra* is any set X with interpretation

- of $0, 1$ as constants $\underline{0}, \underline{1} \in X$ ("bottom" and "top");
- of $-$ as a unary operation $\underline{-} : X \to X$ ("complement"), and
- of $+, *$ as binary operations $\underline{+}, \underline{*} : X^2 \to X$ ("join" and "meet"),
- of $\equiv$ as identity, $=$,

satisfying the following axioms:

1. Neutral elements
$$x + 0 \equiv x \qquad x * 1 \equiv x$$
2. Associativity
$$(x + y) + z \equiv x + (y + z) \qquad (x * y) * z \equiv x * (y * z)$$
3. Commutativity
$$x + y \equiv y + x \qquad x * y \equiv y * x$$
4. Distributivity
$$x + (y * z) \equiv (x + y) * (x + z) \qquad x * (y + z) \equiv (x * y) + (x * z)$$
5. Complement
$$x * (-x) \equiv 0 \qquad x + (-x) \equiv 1$$

Be wary of confusing the meaning of the symbols "0,1,+, $-$, $*$" above with the usual meaning of arithmetic zero, one, plus, etc. – they have nothing in common, except for the superficial syntax!!!

Roughly speaking, the word "algebra", stands here for the fact that the only formulae are equalities and reasoning happens by using properties of equality: reflexivity – $x \equiv x$, symmetry – $\frac{x \equiv y}{y \equiv x}$, transitivity – $\frac{x \equiv y \ ; \ y \equiv z}{x \equiv z}$, and by "substituting equals for equals", according to the rule:

$$\frac{g[x] \equiv z \quad ; \quad x \equiv y}{g[y] \equiv z} \tag{5.18}$$

i.e. given two equalities $x \equiv y$ and $g[x] \equiv z$ we can derive $g[y] \equiv z$. (Compare this to the provable equivalence from theorem 4.25, in particular, the rule from 4.26.) Other laws, which we listed before, are derivable in this manner from the above axioms. For instance,

• **Idempotency** of $*$, i.e.

$$x \equiv x * x \tag{5.19}$$

is shown as follows

$: x \overset{1}{\equiv} x * 1 \overset{5}{\equiv} x * (x + (-x)) \overset{4}{\equiv} (x * x) + (x * (-x)) \overset{5}{\equiv} (x * x) + 0 \overset{1}{\equiv} x * x$

(Similarly, $x \equiv x + x$.)

• Another fact is a form of **absorption**:

$$0 * x \equiv 0 \qquad\qquad x + 1 \equiv 1 \tag{5.20}$$

$: 0 * x \overset{5}{\equiv} (x * (-x)) * x \overset{3}{\equiv} ((-x) * x) * x \overset{2}{\equiv} (-x) * (x * x) \overset{(5.19)}{\equiv} (-x) * x \overset{5}{\equiv} 0$

$: x+1 \stackrel{5}{\equiv} x+(x+(-x)) \stackrel{3,2}{\equiv} (x+x)+(-x) \stackrel{(5.19)}{\equiv} x+(-x) \stackrel{5}{\equiv} 1$

• **Complement of any x is determined uniquely** by the two properties from 5., namely, any y satisfying both these properties is necessarily x's complement:

$$\text{if a) } x+y \equiv 1 \quad \text{and} \quad \text{b) } y*x \equiv 0 \quad \text{then} \quad y \equiv -x \tag{5.21}$$

$: y \stackrel{1}{\equiv} y*1 \stackrel{5}{\equiv} y*(x+(-x)) \stackrel{4}{\equiv} (y*x)+(y*(-x)) \stackrel{b)}{\equiv} 0+(y*(-x)) \stackrel{5}{\equiv} (x*(-x))+(y*(-x)) \stackrel{3,4}{\equiv} (x+y)*(-x) \stackrel{a)}{\equiv} 1*(-x) \stackrel{3,1}{\equiv} -x$

• **Involution**,

$$-(-x) \equiv x \tag{5.22}$$

follows from (5.21). By 5. we have $x*(-x) \equiv 0$ and $x+(-x) \equiv 1$ which, by (5.21) imply that $x \equiv -(-x)$.

The new notation used in the definition 5.17 was meant to emphasize the fact that boolean algebras are more general structures. It should have been obvious, however, that the intended interpretation of these new symbols was as follows:

sets	$\leftarrow$	*boolean algebra*	$\rightarrow$	PL
$\wp(U) \ni$	$\leftarrow$	x	$\rightarrow$	$\in \{\mathbf{1}, \mathbf{0}\}$
$x \cup y$	$\leftarrow$	$x+y$	$\rightarrow$	$x \underline{\vee} y$
$x \cap y$	$\leftarrow$	$x*y$	$\rightarrow$	$x \underline{\wedge} y$
$\overline{x}$	$\leftarrow$	$-x$	$\rightarrow$	$\underline{\neg} x$
$\varnothing$	$\leftarrow$	0	$\rightarrow$	$\mathbf{0}$
U	$\leftarrow$	1	$\rightarrow$	$\mathbf{1}$
$=$	$\leftarrow$	$\equiv$	$\rightarrow$	$\Leftrightarrow$

The fact that any set $\wp(U)$ obeys the set laws from page 5, and that the set $\mathbf{B} = \{\mathbf{1}, \mathbf{0}\}$ obeys the PL-laws from 4.1 amounts to the statement that these structures are, in fact, boolean algebras under the above interpretation of boolean operations. (Not all the axioms of boolean algebras were included, so one has to verify, for instance, the laws for neutral elements and complement, but this is an easy task.) Thus, all the above formulae (5.19)–(5.22) will be valid for these structures under the interpretation from the table above, i.e.,

$\wp(U)$−law	$\leftarrow$	boolean algebra law	$\rightarrow$	PL−law	
$A \cap A = A$	$\leftarrow$	$x*x = x$	$\rightarrow$	$A \wedge A \Leftrightarrow A$	(5.19)
$\varnothing \cap A = \varnothing$	$\leftarrow$	$0*x = 0$	$\rightarrow$	$\bot \wedge A \Leftrightarrow \bot$	(5.20)
$U \cup A = U$	$\leftarrow$	$1+x = 1$	$\rightarrow$	$\top \vee A \Leftrightarrow \top$	(5.20)
$\overline{(\overline{A})} = A$	$\leftarrow$	$-(-x) = x$	$\rightarrow$	$\neg(\neg A) \Leftrightarrow A$	(5.22)

The last fact for PL was, for instance, verified at the end of example 5.9.

Thus the two possible semantics for our $\mathsf{WFF}_{\mathsf{PL}}$, namely, the set $\{\mathbf{1}, \mathbf{0}\}$ with the logical interpretation of the (boolean) connectives as $\underline{\neg}$, $\underline{\wedge}$, etc. on the one hand, and an arbitrary $\wp(U)$ with the interpretation of (boolean) connectives as , $\cap$, etc. are both boolean algebras.

Now, we said that boolean algebras come with the reasoning system – equational logic – which allows us to prove equations $A \equiv B$, where $A, B \in \mathsf{BE}$. On the other hand, the axiomatic systems for PL, e.g., the Hilbert's system $\vdash_{\mathcal{H}}$, proved only simple boolean expressions: $\vdash_{\mathcal{H}} A$. Are these two reasoning systems related in some way? They are, indeed, but we will not study precise relationship in detail. At this point we only state the following fact: if $\vdash_{\mathcal{H}} A$ then also the equation $A \equiv 1$ is provable in equational logic, where $A \equiv 1$ is obtained by replacing all subformulas $x \to y$ by the respective expressions $-x + y$ (recall $x \to y \Leftrightarrow \neg x \vee y$). For instance, the equation corresponding to the first axiom of $\vdash_{\mathcal{H}} A \to (B \to A)$ is obtained by translating $\to$ to the equivalent boolean expression: $(-A + -B + A) = 1$. You may easily verify provability of this equation from axioms 1.-5., as well as that it holds under set interpretation – for any sets $A, B \subseteq U : \overline{A} \cup \overline{B} \cup A = U$. . . . . [end optional]

Exercises 5.

EXERCISE 5.1 Recall example 5.12 and set up the boolean table for the formula $a \vee b$ where $\vee$ is trying to represent "or". Consider if it is possible – and if so, then how – to represent the following statements using your definition:

(1) $x < y$ or $x = y$.
(2) John is ill or Paul is ill.
(3) Either we go to cinema or we stay at home.

EXERCISE 5.2 Write the boolean tables for the following formulae and decide to which among the four classes from Fact 5.10 (Definition 5.8) they belong:

(1) $a \to (b \to a)$
(2) $(a \to (b \to c)) \ \to \ ((a \to b) \to (a \to c))$
(3) $(\neg b \to \neg a) \ \to \ (a \to b)$

EXERCISE 5.3 Rewrite the laws 1. Neutral elements and 5. Complement of boolean algebra to their propositional form and verify their validity using boolean tables. (Use $\top$ for 1 and $\bot$ for 0.)

EXERCISE 5.4 Verify whether $(a \to b) \to b$ is a tautology. Is the following proof correct? If not, what is wrong with it?

$$\begin{array}{ll}
1: A, A \to B \vdash_{\mathcal{N}} A \to B & A0 \\
2: A, A \to B \vdash_{\mathcal{N}} A & A0 \\
3: A \to B \vdash_{\mathcal{N}} B & MP(2,1) \\
4: \vdash_{\mathcal{N}} (A \to B) \to B & DT
\end{array}$$

EXERCISE 5.5 Verify the following facts:

(1) $A_1 \to (A_2 \to (A_3 \to B)) \;\Leftrightarrow\; (A_1 \wedge A_2 \wedge A_3) \to B$.
(2) $(A \wedge (A \to B)) \Rightarrow B$

Use boolean tables to show that the following formulae are contradictions:
(3) $\neg(B \to B)$
(4) $\neg(B \vee C) \wedge C$

Determine now what sets are denoted by these two expressions – for the set-interpretation of $\to$ recall remark 5.13.

EXERCISE 5.6 Show which of the following pairs are equivalent:

$$\begin{array}{ll}
1. & A \to (B \to C) \;?\; (A \to B) \to C \\
2. & A \to (B \to C) \;?\; B \to (A \to C) \\
3. & A \wedge \neg B \;?\; \neg(A \to B)
\end{array}$$

EXERCISE 5.7 Prove a result analogous to corollary 5.16 for $\subseteq$ and $\Rightarrow$ instead of $=$ and $\Leftrightarrow$.

EXERCISE 5.8 Use point 3. from exercise 5.6 to verify that $(C \wedge D) \to (A \wedge \neg B)$ and $(C \wedge D) \to \neg(A \to B)$ are equivalent. How does this earlier exercise simplify the work here?

EXERCISE 5.9 (Compositionality and substitutivity)
Let $F[A]$ be an arbitrary formula with a subformula A, and V be a valuation of all propositional variables occurring in $F[A]$. It induces the valuation $\widehat{V}(F[A])$ of the whole formula.
Let B be another formula and assume that for all valuations V, $\widehat{V}(A) = \widehat{V}(B)$. Use induction on the complexity of $F[\_]$ to show that then $\widehat{V}(F[A]) = \widehat{V}(F[B])$, for all valuations V.
(Hint: The structure of the proof will be similar to that of theorem 4.25. Observe, however, that here you are proving a completely different fact concerning not the provability relation but the semantic interpretation of the formulae – not their provable but tautological equivalence.)

EXERCISE 5.10 Tautological equivalence $A \Leftrightarrow B$ amounts *almost* to the fact that A and B have the same interpretation. We have to make the meaning of this "almost" more precise.

(1) Show that neither of the two relations $A \Leftrightarrow B$ and $\underline{A} = \underline{B}$ imply the other, i.e., give examples of A and B such that (a) $A \Leftrightarrow B$ but $\underline{A} \neq \underline{B}$

and (b) $\underline{A} = \underline{B}$ but not $A \Leftrightarrow B$.

(Hint: Use extra/different propositional variables not affecting the truth of the formula.)

(2) Explain why the two relations are the same whenever A and B contain the same variables.

(3) Finally explain that if $\underline{A} = \underline{B}$ then there exists some formula C obtained from B by "renaming" the propositional variables, such that $A \Leftrightarrow C$.

<u>EXERCISE 5.11</u> Let Φ be an arbitrary, possibly infinite, set of formulae. The following conventions generalize the notion of (satisfaction of) binary conjunction/disjunction to such arbitrary sets. Given a valuation V, we say that Φ's:

- conjunction is true under V, $\widehat{V}(\bigwedge \Phi) = \mathbf{1}$, iff for all formulae A : if $A \in \Phi$ then $\widehat{V}(A) = \mathbf{1}$.
- disjunction is true udner V, $\widehat{V}(\bigvee \Phi) = \mathbf{1}$, iff there exists an A such that $A \in \Phi$ and $\widehat{V}(A) = \mathbf{1}$.

Let now Φ be a set containing zero or one formulae. What would be the most natural interpretations of the expressions "the conjunction of formulae in Φ" and "the disjunction of formulae in Φ" ?

Chapter 6
Soundness and Completeness

- Adequate Sets of Connectives
- Normal Forms
  - DNF
  - CNF
- Soundness of $\mathcal{N}$ and $\mathcal{H}$
- Completeness of $\mathcal{N}$ and $\mathcal{H}$

This chapter focuses on the relations between the syntax and axiomatic systems for PL and their semantic counterpart. Before we discuss the central concepts of soundness and completeness, we will first ask about the 'expressive power' of the language we have introduced. Expressive power of a language for statement logic can be identified with the possibilities it provides for defining various boolean functions. In section 1 we show that *all* boolean functions can be defined by the formulae in our language. Section 2 explores a useful consequence of this fact showing that each formula can be written equivalently in a special normal form. The rest of the chapter then studies soundness and completeness of our axiomatic systems.

1: Adequate Sets of Connectives

This and next section study the relation we have established in definition 5.6 between formulae of PL and boolean functions (on our two-element set $\mathbf{B}$). According to this definition, any PL formula defines a boolean function. The question now is the opposite: Can any boolean function be defined by some formula of PL?

Introducing abbreviations $\wedge$, $\vee$ and others in chapter 5, section 3, we remarked that they are not necessary but merely convenient. The meaning of their being "not necessary" is that any function which can be defined by a formula containing these connectives, can also be defined by a formula which does not contain them. E.g., a function defined using $\vee$ can be also defined using $\neg$ and $\rightarrow$.

Concerning our main question we need a stronger notion, namely, the notion of a complete, or *adequate* set of connectives which is sufficient to define all boolean functions.

Definition 6.1 A set AS of connectives is *adequate* if for every $n > 0$, every boolean-function
$f : \{\mathbf{1}, \mathbf{0}\}^n \rightarrow \{\mathbf{1}, \mathbf{0}\}$ is determined by some formula containing only the connectives from the set AS.

Certainly, not every set is adequate. If we take, for example, the set with only negation $\{\neg\}$, it is easy to see that it cannot be adequate. It is a unary operation, so that it will never give rise to, for instance, a function with two arguments. But it won't even be able to define all unary functions. It can be used to define only two functions $\mathbf{B} \rightarrow \mathbf{B}$ – inverse (i.e., $\underline{\neg}$ itself) and identity ($\underline{\neg\neg}$):

x	$\underline{\neg}(x)$	$\underline{\neg\neg}(x)$
$\mathbf{1}$	$\mathbf{0}$	$\mathbf{1}$
$\mathbf{0}$	$\mathbf{1}$	$\mathbf{0}$

(The proof-theoretic counterpart of the fact that $\underline{\neg\neg}$ is identity was lemma 4.10 showing provable equivalence of B and $\neg\neg B$.) Any further applications of $\underline{\neg}$ will yield one of these two functions. The constant functions ($f(x) = \mathbf{1}$ or $f(x) = \mathbf{0}$) can not be defined using exclusively this single connective.

The following theorem identifies the first adequate set.

Theorem 6.2 $\{\neg, \wedge, \vee\}$ is an adequate set.

Proof. Let $f : \{\mathbf{1}, \mathbf{0}\}^n \rightarrow \{\mathbf{1}, \mathbf{0}\}$ be an arbitrary boolean function of n arguments (for some $n > 0$) with a given boolean table.
If f always produces $\mathbf{0}$ then the contradiction $(a_1 \wedge \neg a_1) \vee \ldots \vee (a_n \wedge \neg a_n)$ determines f.
For the case when f produces $\mathbf{1}$ for at least one set of arguments, we write the proof to the left illustrating it with an example to the right.

Proof	Example
Let $a_1, a_2, \ldots, a_n$ be distinct propositional variables listed in increasing order. The boolean table for f has 2^n rows. Let $\underline{a}_c^r$ denote the entry in the c-th column and r-th row.	$\begin{array}{c\|c\|c} a_1 & a_2 & f(a_1,a_2) \\ \hline \mathbf{1} & \mathbf{1} & \mathbf{0} \\ \mathbf{1} & \mathbf{0} & \mathbf{1} \\ \mathbf{0} & \mathbf{1} & \mathbf{1} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{array}$
For each $1 \leq r \leq 2^n$, $1 \leq c \leq n$ let $L_c^r = \begin{cases} a_c \text{ if } \underline{a}_c^r = \mathbf{1} \\ \neg a_c \text{ if } \underline{a}_c^r = \mathbf{0} \end{cases}$	$L_1^1 = a_1,\ L_2^1 = a_2$ $L_1^2 = a_1,\ L_2^2 = \neg a_2$ $L_1^3 = \neg a_1,\ L_2^3 = a_2$ $L_1^4 = \neg a_1,\ L_2^4 = \neg a_2$
For each row $1 \leq r \leq 2^n$ form the conjunction: $C^r = L_1^r \wedge L_2^r \wedge \ldots \wedge L_n^r$. Then for all rows r and $p \neq r$: $\underline{C}^r(\underline{a}_1^r, \ldots, \underline{a}_n^r) = \mathbf{1}$ and $\underline{C}^r(\underline{a}_1^p, \ldots, \underline{a}_n^p) = \mathbf{0}$.	$C^1 = a_1 \wedge a_2$ $C^2 = a_1 \wedge \neg a_2$ $C^3 = \neg a_1 \wedge a_2$ $C^4 = \neg a_1 \wedge \neg a_2$
Let D be the disjunction of those C^r for which $f(\underline{a}_1^r, \ldots, \underline{a}_n^r) = \mathbf{1}$.	$D = C^2 \vee C^3$ $= (a_1 \wedge \neg a_2) \vee (\neg a_1 \wedge a_2)$

The claim is: the function determined by D is f, i.e., $\underline{D} = f$. If $f(\underline{a}_1^r, \ldots, \underline{a}_n^r) = \mathbf{1}$ then D contains the corresponding disjunct C^r which, since $\underline{C}^r(\underline{a}_1^r, \ldots, \underline{a}_n^r) = \mathbf{1}$, makes $\underline{D}(\underline{a}_1^r, \ldots, \underline{a}_n^r) = \mathbf{1}$. If $f(\underline{a}_1^r, \ldots, \underline{a}_n^r) = \mathbf{0}$, then D does not contain the corresponding disjunct C^r. But for all $p \neq r$ we have $\underline{C}^p(\underline{a}_1^r, \ldots, \underline{a}_n^r) = \mathbf{0}$, so none of the disjuncts in D will be $\mathbf{1}$ for these arguments, and hence $\underline{D}(\underline{a}_1^r, \ldots, \underline{a}_n^r) = \mathbf{0}$.

QED (6.2)

Corollary 6.3 The following sets of connectives are adequate:

1. $\{\neg, \vee\}$ 2. $\{\neg, \wedge\}$ 3. $\{\neg, \rightarrow\}$

Proof.

1. By de Morgan's law $A \wedge B \Leftrightarrow \neg(\neg A \vee \neg B)$. Thus we can express each conjunction by negations and disjunction. Using distributive and associative laws, this allows us to rewrite the formula obtained in the proof of Theorem 6.2 to an equivalent one without conjunction
2. The same argument as above.

3. According to definition 5.11, $A \vee B \stackrel{\text{def}}{=} \neg A \to B$. This, however, was a merely syntactic definition of a new symbol '$\vee$'. Here we have to show that the boolean functions $\underline{\neg}$ and $\underline{\to}$ can be used to define the boolean function $\underline{\vee}$. But this was done in Exercise 5.2.(1) where the semantics (boolean table) for $\underline{\vee}$ was given according to the definition 5.11, i.e., where $A \vee B \Leftrightarrow \neg A \to B$, required here, was shown. So the claim follows from point 1. **QED** (6.3)

Remark.
Our definition of "adequate" does not require that any formula determines the functions from $\{\mathbf{1},\mathbf{0}\}^0$ into $\{\mathbf{1},\mathbf{0}\}$. $\{\mathbf{1},\mathbf{0}\}^0$ is the singleton set $\{\epsilon\}$ and there are two functions from it into $\{\mathbf{1},\mathbf{0}\}$, namely $\{\epsilon \mapsto \mathbf{1}\}$ and $\{\epsilon \mapsto \mathbf{0}\}$. These functions are not determined by any formula in the connectives $\wedge, \vee, \to, \neg$. The best approximations are tautologies and contradictions like $(a \to a)$ and $\neg(a \to a)$, which we in fact took to be the special formulae $\top$ and $\bot$. However, these determine the functions $\{\mathbf{1} \mapsto \mathbf{1},\ \mathbf{0} \mapsto \mathbf{1}\}$ and $\{\mathbf{1} \mapsto \mathbf{0},\ \mathbf{0} \mapsto \mathbf{0}\}$, which in a strict set-theoretic sense are distinct from the functions above. To obtain a set of connectives that is adequate in a stricter sense, one would have to introduce $\top$ or $\bot$ as a special formula (in fact a 0-argument connective) that does not contain any propositional variables.

2: DNF, CNF

The fact that, for instance, $\{\neg, \to\}$ is an adequate set, vastly reduces the need for elaborate syntax when studying propositional logic. We can (as we indeed have done) restrict the syntax of $\mathsf{WFF}^{\mathsf{PL}}$ to the necessary minimum. This simplifies many proofs concerned with the syntax and the axiomatic systems since such proofs involve, typically, induction on the definition (of $\mathsf{WFF}^{\mathsf{PL}}$, of $\vdash$, etc.) Adequacy of the set means that any entity (any function defined by a formula) has some specific, "normal" form using only the connectives from the adequate set.

Now we will show that even more "normalization" can be achieved. Not only every (boolean) function can be defined by *some* formula using only the connectives from one adequate set – every such a function can be defined by such a formula which, in addition, has a very specific form.

Definition 6.4 A formula B is in

(1) *disjunctive normal form*, DNF, iff $B = C_1 \vee ... \vee C_n$, where each C_i is a conjunction of literals.
(2) *conjunctive normal form*, CNF, iff $B = D_1 \wedge ... \wedge D_n$, where each D_i is a disjunction of literals.

Example 6.5
Let $\Sigma = \{a, b, c\}$.

- $(a \wedge b) \vee (\neg a \wedge \neg b)$ and $(a \wedge b \wedge \neg c) \vee (\neg a \wedge c)$ are both in DNF
- $a \vee b$ and $a \wedge b$ are both in DNF and CNF
- $(a \vee (b \wedge c)) \wedge (\neg b \vee a)$ is neither in DNF nor in CNF
- $(a \vee b) \wedge c \wedge (\neg a \vee \neg b \vee \neg c)$ is in CNF but not in DNF
- $(a \wedge b) \vee (\neg a \vee \neg b)$ is in DNF but not in CNF.

The last formula can be transformed into CNF applying the laws like those from 5.4.1 on p. 104. The distributivity and associativity laws yield:

$$(a \wedge b) \vee (\neg a \vee \neg b) \Leftrightarrow (a \vee \neg a \vee \neg b) \wedge (b \vee \neg a \vee \neg b)$$

and the formula on the right hand side is in CNF. □

Recall the form of the formula constructed in the proof of Theorem 6.2 – it was in DNF! Thus, this proof tells us not only that the set $\{\neg, \wedge, \vee\}$ is adequate but also

Corollary 6.6 Each formula is logically equivalent to a formula in DNF.

Proof. For any B there is a D in DNF such that $\underline{B} = \underline{D}$. By "renaming" the propositional variables of D (see exercise 5.10) one obtains a new formula B_D in DNF, such that $B \Leftrightarrow B_D$. **QED** (6.6)

We now use this corollary to show the next.

Corollary 6.7 Each formula is logically equivalent to a formula in CNF.

Proof. Assuming, by Corollary 6.3, that the only connectives in B are $\neg$ and $\wedge$, we proceed by induction on B's complexity:

a :: A propositional variable is a conjunction over one literal, and hence is in CNF.

$\neg A$ :: By Corollary 6.6, A is equivalent to a formula A_D in DNF. Exercise 6.10 allows us to conclude that B is equivalent to B_C in CNF.

$C \wedge A$ :: By IH, both C and A have CNF : C_C, A_C. Then $C_C \wedge A_C$ is easily transformed into CNF (using associative laws), i.e., we obtain an equivalent B_C in CNF. **QED** (6.7)

The concepts of disjunctive and conjunctive normal forms may be ambiguous, as the definitions do not determine uniquely the form. If 3 variables a, b, c are involved, the CNF of $\neg(a \wedge \neg b)$ can be naturally seen as $\neg a \vee b$.

But one could also require all variables to be present, in which case the CNF would be $(\neg a \vee b \vee c) \wedge (\neg a \vee b \vee \neg c)$. Applying distributivity to the following formula in DNF: $(b \wedge a) \vee (c \wedge a) \vee (b \wedge \neg a) \vee (c \wedge \neg a)$, we obtain CNF $(b \vee c) \wedge (a \vee \neg a)$. The last, tautological conjunct can be dropped, so that also $b \vee c$ can be considered as the CNF of the original formula.

2.1: CNF, Clauses and Satisfiability .... [optional]

An algorithm for constructing a DNF follows from the proof of Theorem 6.2. CNF can be constructed by performing the dual moves. Given a boolean table for a function, pick the rows with value **0** and for each such row form the conjunction of literals as in the proof of Theorem 6.2. Take the conjunction of negations of all these rows – CNF results now from moving negations inwards using de Morgan. For the function from the proof of Theorem 6.2, we obtain $\neg(a_1 \wedge a_2) \wedge \neg(\neg a_1 \wedge \neg a_2)$ which, after application of de Morgan, yields $(\neg a_1 \vee \neg a_2) \wedge (a_1 \vee a_2)$.

A disjunction of literals is called a clause and thus CNF is a conjunction of clauses. It plays a crucial role in many applications and the problem of deciding if a given CNF formula is satisfiable, SAT, is the paradigmatic NP-complete problem. (The problem of deciding if a given DNF formula is satisfiable is trivial – it suffices to check if it contains any conjunction without any pair of complementary literals.)

For a given set V of $n = |V|$ variables, a V-clause or (abstracting from the actual names of the variables and considering only their number) n-clause is one with n literals. There are, of course, 2^n distinct n-clauses, and the set containing them all is denoted $C(V)$. The following fact may seem at first surprising, claiming that a subset of such n-clauses is unsatisfiable if and only if it is the whole $C(V)$.

Fact 6.8 Let $T \subseteq C(V) : mod(T) = \varnothing \Leftrightarrow T = C(V)$.

Proof. Proceeding by induction on the number $|V| = n$ of variables, the claim is trivially verified for $n = 1$ or $n = 2$. For any $n + 1 > 2$, in any subset of $2^n - 1 = 2^{n-1} + 2^{n-1} - 1$ clauses, at least one of the literals, say x, occurs 2^{n-1} times. These clauses are satisfied by making $x = \mathbf{1}$. From the remaining $2^{n-1} - 1$ clauses, we remove $\overline{x}$, and obtain $2^{n-1} - 1$ clauses over $n - 1$ variables, for which IH applies. Since every subset of $2^n - 1$ clauses is satisfiable, so is every smaller subset, for every n. **QED** (6.8)

This follows, in fact, from a more general observation. Given a subset $T \subseteq C(V)$ of V-clauses, its models are determined exactly by the clauses in $C(V) \setminus T$. For an n-clause C, let $val(C)$ denote the valuation assigning **0** to all positive literals and **1** to all negative literals in C.

Fact 6.9 Let $T \subseteq C(V) : mod(T) = \{val(C) \mid C \in C(V) \setminus T\}$.

This holds because each $C \in C(V) \setminus T$ differs from each $M \in T$ at least at one literal, say $l_{CM} \in M$ and $\overline{l}_{MC} \in C$ (where $l - \overline{l}$ denotes arbitrarily complementary pair of literals). Taking the complements of all literals in C will then make true

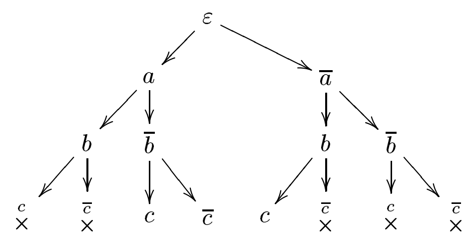
$val(C) \models M \in T$, at least by the respective literal $\bar{l}_{CM} \in M$, on which the two differ. Since C contains such a literal for each clause from T, $val(C) \models T$.

Example 6.10
For $V = \{a, b, c\}$, let the theory T contain the 5 V-clauses listed in the first column.

	$C(V) \setminus T$								
	$a \vee \neg b \vee \neg c$			$\neg a \vee b \vee \neg c$			$\neg a \vee b \vee c$		
$val(\_)$:	a	b	c	a	b	c	a	b	c
T	**0**	**1**	**1**	**1**	**0**	**1**	**1**	**0**	**0**
1. $a \vee b \vee c$		**1**	**1**	**1**		**1**	**1**		
2. $a \vee b \vee \neg c$		**1**		**1**			**1**		**1**
3. $a \vee \neg b \vee c$			**1**	**1**		**1**	**1**	**1**	
4. $\neg a \vee \neg b \vee c$	**1**		**1**		**1**	**1**		**1**	
5. $\neg a \vee \neg b \vee \neg c$	**1**				**1**			**1**	**1**

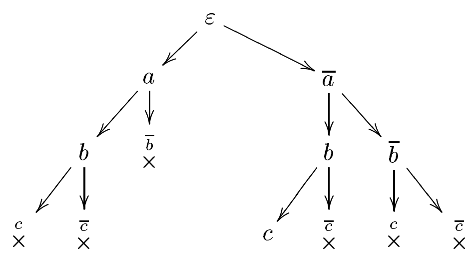
The three clauses outside T give valuations in the second row. For each of them, the literals made **1** are marked for each clause of T in its respective row. $\square$

Fact 6.9 finds important application in the decision procedures for satisfiability and in counting the number of models of a theory. It is commonly applied as the so called "semantic tree", representing the models of a theory. Order the atoms V of the alphabet in an arbitrary total ordering $v_1, v_2, ..., v_n$, and build a complete binary tree by staring with the empty root (level 0) and, at each level $i > 0$, adding two children, v_i and $\overline{v}_i$, to each node at the previous level $i - 1$. A complete branch in such a tree (i.e., each of the 2^n leafs) represents a possible assignment of the values **1** to each node v_i and **0** to each node $\overline{v}_i$ on the branch. According to Fact 6.9, each clause from $C(V)$ excludes one such branch, formed by negations of all literals in the clause. One says that the respective branch becomes "closed" and, on the drawing below, these closed branches, for the five clauses from T, are marked by $\times$:



The literals on the branches terminating with the unmarked leafs give the three models of T, as in Example 6.10.

In practice, the algorithms do not build the complete tree which quickly becomes prohibitively large as n grows. Instead, it is developed gradually, observing if there remain any open branches. Usually, a theory is given by clauses of various length, much shorter than the total number n of variables. Such a shorter clause excludes then *all* branches containing all its negated literals. E.g., if the theory is extended with the clause $\neg a \vee b$, all branches containing a and $\overline{b}$ become closed, as shown below. Once closed, a branch is never extended during the further construction of the tree.



The resulting theory still has one model, represented by the open branch $\overline{a}, b, c$. Adding any clause closing this branch, i.e., $a \vee \neg b \vee \neg c$ or $\neg c$ or $\neg b$ or $\neg c \vee a$, etc., makes the theory inconsistent. . . . . [end optional]

3: Soundness

◇———— a Background Story ————◇

The library offers its customers the possibility of ordering books on internet. From the main page one may ask the system to find the book one wishes to borrow. (We assume that appropriate search engine will always find the book one is looking for or else give a message that it could not be identified. In the sequel we are considering only the case when the book you asked for was found.)

The book (found by the system) may happen to be immediately available for loan. In this case, you may just reserve it and our story ends here. But the most frequent case is that the book is on loan or else must be borrowed from another library. In such a case, the system gives you the possibility to order *it: you mark the book and the system will send you a message as soon as the book becomes available. (You need no message as long as the book is not available and the system need not inform you about that.) Simplicity of this scenario notwithstanding, this is actually our whole story.*

There are two distinct assumptions which make us relay on the system when we order a book. The first is that when you get the message that the book is available it really is. *The system will not play* fool with you saying "Hi, the book is here" while it is still on loan with another user. We trust that what the system says ("The book is here") is true. This property is what we call "soundness" of the system – it never provides us with false information.

But there is also another important aspect making up our trust in the system. Suppose that the book actually becomes available, but you do not get the appropriate message. The system is still sound – it does not give you any wrong information – but only because it does not give you any information whatsoever. It keeps silent although it should have said that the book is there and you can borrow it. The other aspect of our trust is that whenever there is a fact to be reported ('the book became available'), the system will do it – this is what we call "completeness".

Just like a system may be sound without being complete (keep silent even though the book arrived to the library), it may be complete without being sound. If it constantly sent messages that the book you ordered was available, it would, sooner or later (namely when the book eventually became available) report the true fact. However, in the meantime, it would provide you with a series of incorrect information – it would be unsound.

Thus, soundness of a system means that whatever it says is correct: it says "The book is here" *only if* it is here. Completeness means that everything that is correct will be said by the system: it says "The book is here" *if* (always when) the book is here. In the latter case, we should pay attention to the phrase "everything that is correct". It makes sense because our setting is very limited. We have one command 'order the book ...', and one possible response of the system: the message that the book became avilable. "Everything that is correct" means here simply that the book you ordered actually is available. It is only this limited context (i.e., limited and well-defined amount of true facts) which makes the notion of completeness meaningfull.

In connection with axiomatic systems one often resorts to another analogy. The axioms and the deduction rules together define the scope of the system's knowledge about the world. If all aspects of this knowledge (all the theorems) are true about the world, the system is sound. This idea has enough intuitive content to be grasped with reference to vague notions of 'knowledge', 'the world', etc. and our illustration with the system saying "The book is here" only when it actually is, merely makes it more specific.

Completeness, on the other hand, would mean that everything that is true about the world (and expressible in the actual language), is also reflected in the system's knowledge (theorems). Here it becomes less clear what the intuitive content of 'completeness' might be. What can one possibly mean by "everything that is true"? In our library example, the user and the system use only very limited language allowing the user to 'order the book ...' and the system to state that it is avaialable. Thus, the possible meaning of "everything" is limited to the book being available or not. One should keep this difference between 'real world' and 'availability of a book' in mind because the notion of completeness is as unnatural in the context of natural language and real world, as it is adequate in the context of bounded, sharply delineated worlds of formal semantics. The limited expressiveness of a formal language plays here crucial role of limiting the discourse to a well-defined set of expressible facts.

The library system should be both sound *and* complete to be useful. For axiomatic systems, the minimal requirement is that that they are sound – completeness is a desirable feature which, typically, is much harder to prove. Also, it is known that there are axiomatic systems which are sound but inherently incomplete. We will not study such systems but merely mention one in the last chapter, theorem 11.21. [5]

◊————————————————————◊

Definition 5.8 introduced, among other concepts, the validity relation $\models A$, stating that A is satisfied by all structures. On the other hand, we studied the syntactic notion of a proof in a given axiomatic system $\mathcal{C}$, which we wrote as $\vdash_{\mathcal{C}} A$. We also saw a generalization of the provability predicate $\vdash_{\mathcal{H}}$ in Hilbert's system to the relation $\Gamma \vdash_{\mathcal{N}} A$, where Γ is a theory – a set of formulae. We now define the *semantic* relation $\Gamma \models A$ of "A being a (tauto)logical consequence of Γ".

Definition 6.11 Given a set $\Gamma \subseteq \mathsf{WFF}_{\mathsf{PL}}$ and $A \in \mathsf{WFF}_{\mathsf{PL}}$ we write:

- $V \models \Gamma$ iff $V \models G$ for all $G \in \Gamma$, where V is some valuation, called a *model* of Γ
- $Mod(\Gamma) = \{V : V \models \Gamma\}$ – the set of all models of Γ
- $\models \Gamma$ iff for all $V : V \models \Gamma$
- $\Gamma \models A$ iff for all $V \in Mod(\Gamma) : V \models A$, i.e., $\forall V : V \models \Gamma \Rightarrow V \models A$

The analogy between the symbols $\models$ and $\vdash$ is not accidental. The former

[5] Thanks to Eivind Kolflaath for the library analogy.

refers to a semantic, while the later to a syntactic notion and, ideally, these two notions should be equivalent in some sense. The following table gives the picture of the intended "equivalences":

Syntactic	*Semantic*
$\vdash_{\mathcal{H}} A$	$\models A$
$\Gamma \vdash_{\mathcal{N}} A$	$\Gamma \models A$

We will see that for $\mathcal{H}$ and $\mathcal{N}$ there is such an equivalence.The equivalence we desire is

$$\Gamma \vdash A \;\Leftrightarrow\; \Gamma \models A \tag{6.12}$$

The implication $\Gamma \vdash_{\mathcal{C}} A \;\Rightarrow\; \Gamma \models A$ is called *soundness* of the proof system $\mathcal{C}$: whatever we can prove in $\mathcal{C}$ from the assumptions Γ, is true in every structure satisfying Γ. This is usually easy to establish as we will see shortly. The problematic implication is the other one – *completeness* – stating that any formula which is true in all models of Γ is provable from the assumptions Γ. ($\Gamma = \varnothing$ is a special case: the theorems $\vdash_{\mathcal{C}} A$ are tautologies and the formulae $\models A$ are those satisfied by all possible structures (since any structure V satisfies the empty set of assumptions).)

Remark 6.13 **[Soundness and Completeness]**
Another way of viewing these two implications is as follows. Given an axiomatic system $\mathcal{C}$ and a theory Γ, the relation $\Gamma \vdash_{\mathcal{C}} \_$ defines the set of formulae – the theorems – $Th_{\mathcal{C}}(\Gamma) = \{A : \Gamma \vdash_{\mathcal{C}} A\}$. On the other hand, given the definition of $\Gamma \models \_$, we obtain a (possibly different) set of formulae, namely, the set $\Gamma^* = \{B : \Gamma \models B\}$ of (tauto)logical consequences of Γ. Soundness of $\mathcal{C}$, i.e., the implication $\Gamma \vdash_{\mathcal{C}} A \Rightarrow \Gamma \models A$ means that any provable consequence is also a (tauto)logical consequence and amounts to the inclusion $Th_{\mathcal{C}}(\Gamma) \subseteq \Gamma^*$. Completeness means that any (tauto)logical consequence of Γ is also provable and amounts to the opposite inclusion $\Gamma^* \subseteq Th_{\mathcal{C}}(\Gamma)$.

For proving soundness of a system consisting of axioms and proof rules (like $\mathcal{H}$ or $\mathcal{N}$), one has to show that the axioms of the system are valid and that the rules preserve truth: whenever the assumptions of the rule are satisfied in a model M, then so is the conclusion. (Since $\mathcal{H}$ treats only tautologies, this claim reduces there to preservation of validity: whenever the assumptions of the rule are valid, so is the conclusion.) When these two facts are established, a straightforward induction proof shows that all theorems of the system must be valid. We show soundness and completeness of $\mathcal{N}$.

Theorem 6.14 **[Soundness]** Let $\Gamma \subseteq \mathsf{WFF}_{\mathsf{PL}}$ and $A \in \mathsf{WFF}_{\mathsf{PL}}$. Then $\Gamma \vdash_{\mathcal{N}} A \;\Rightarrow\; \Gamma \models A$

Proof. From the above remarks, we have to show that all axioms are valid, and that MP preserves validity:

A1–A3 :: In exercise 5.2 we have seen that all axioms of $\mathcal{H}$ are valid, i.e., satisfied by any structure. In particular, the axioms are satisfied by all models of Γ, for any Γ.

A0 :: The axiom schema $A0$ allows us to conclude $\Gamma \vdash_{\mathcal{N}} B$ for any $B \in \Gamma$. This is obviously sound: any model V of Γ must satisfy all the formulae of Γ and, in particular, B.

MP :: Suppose $\Gamma \models A$ and $\Gamma \models A \to B$. Then, for an arbitrary $V \in Mod(\Gamma)$ we have $\widehat{V}(A) = \mathbf{1}$ and $\widehat{V}(A \to B) = \mathbf{1}$. Consulting the boolean table for $\underline{\to}$: the first assumption reduces the possibilites for V to the two rows for which $\overline{V}(A) = \mathbf{1}$, and then, the second assumption to the only possible row in which $\overline{V}(A \to B) = \mathbf{1}$. In this row $\overline{V}(B) = \mathbf{1}$, so $V \models B$. Since V was arbitrary model of Γ, we conclude that $\Gamma \models B$. **QED** (6.14)

You should not have any problems with simplifying this proof to obtain the soundness of $\mathcal{H}$.

Corollary 6.15 Every satisfiable theory is consistent.

Proof. We show the equivalent statement that every inconsistent theory is unsatisfiable:
if $\Gamma \vdash_{\mathcal{N}} \bot$ then $\Gamma \models \bot$ by theorem 6.14, hence Γ is not satisfiable (since $\underline{\neg}(x \underline{\to} x) = \mathbf{0}$). **QED** (6.15)

Remark 6.16 **[Equivalence of two soundness notions]**
Soundness is often expressed in the form of corollary 6.15. In fact, the two formulations are equivalent:

6.14. $\Gamma \vdash_{\mathcal{N}} A \Rightarrow \Gamma \models A$

6.15. (exists $V : V \models \Gamma) \Rightarrow \Gamma \not\vdash_{\mathcal{N}} \bot$

The implication 6.14 $\Rightarrow$ 6.15 is given in the proof of corollary 6.15. For the opposite: if $\Gamma \vdash_{\mathcal{N}} A$ then $\Gamma \cup \{\neg A\}$ is inconsistent (Exercise 4.5) and hence (by 6.15) unsatisfiable, i.e., for any $V : V \models \Gamma \Rightarrow V \not\models \neg A$. But if $V \not\models \neg A$ then $V \models A$, and so, since V was arbitrary, $\Gamma \models A$.

4: Completeness

The proof of completeness involves several lemmas which we now proceed to establish. Just as there are two equivalent ways of expressing soundness (remark 6.16), there are two equivalent ways of expressing completeness. One (corresponding to corollary 6.15) says that every consistent theory is satisfiable and the other that any valid formula is provable.

Lemma 6.17 The following two formulations of completeness are equivalent:

(1) $\Gamma \nvdash_{\mathcal{N}} \bot \;\Rightarrow\; Mod(\Gamma) \neq \varnothing$
(2) $\Gamma \models A \;\Rightarrow\; \Gamma \vdash_{\mathcal{N}} A$

Proof. 1. $\Rightarrow$ 2.) Assume 1. and $\Gamma \models A$, i.e., for any $V : V \models \Gamma \Rightarrow V \models A$. Then $\Gamma \cup \{\neg A\}$ has no model and, by 1., $\Gamma, \neg A \vdash_{\mathcal{N}} \bot$. By Deduction Theorem $\Gamma \vdash_{\mathcal{N}} \neg A \to \bot$, and so $\Gamma \vdash_{\mathcal{N}} A$ by Exercise 4.1.5 and lemma 4.10.1.
2. $\Rightarrow$ 1.) Assume 2. and $\Gamma \nvdash_{\mathcal{N}} \bot$. By (the observation before) lemma 4.28 this means that there is an A such that $\Gamma \nvdash_{\mathcal{N}} A$ and, by 2., that $\Gamma \not\models A$. This means that there is a structure V such that $V \not\models A$ and $V \models \Gamma$. Thus 1. holds. **QED** (6.17)

We prove the first of the above formulations: we take an arbitrary Γ and, assuming that it is consistent, i.e., $\Gamma \nvdash_{\mathcal{N}} \bot$, we show that $Mod(\Gamma) \neq \varnothing$ by constructing a particular structure which we prove to be a model of Γ. This proof is not the simplest possible for PL. However, we choose to do it this way because it illustrates the general strategy used later in the completeness proof for FOL. Our proof uses the notion of a *maximal consistent* theory:

Definition 6.18 A theory $\Gamma \subset \mathsf{WFF}^{\Sigma}_{\mathsf{PL}}$ is *maximal consistent* iff it is consistent and, for any formula $A \in \mathsf{WFF}^{\Sigma}_{\mathsf{PL}}$, $\Gamma \vdash_{\mathcal{N}} A$ or $\Gamma \vdash_{\mathcal{N}} \neg A$.

From exercise 4.5 we know that if Γ is consistent then for any A at most one of $\Gamma \vdash_{\mathcal{N}} A$ and $\Gamma \vdash_{\mathcal{N}} \neg A$ is the case – Γ can not prove too much. Put a bit differently, if Γ is consistent then the following holds for any formula A :

$$\Gamma \vdash_{\mathcal{N}} A \Rightarrow \Gamma \nvdash_{\mathcal{N}} \neg A \quad \text{or equivalently} \quad \Gamma \nvdash_{\mathcal{N}} A \textit{ or } \Gamma \nvdash_{\mathcal{N}} \neg A \tag{6.19}$$

– if Γ proves something (A) then there is something else (namely $\neg A$) which Γ does not prove.

Maximality is a kind of the opposite – Γ can not prove too little: if Γ does *not* prove something ($\neg A$) then there must be something else it proves (namely A):

$$\Gamma \vdash_{\mathcal{N}} A \Leftarrow \Gamma \nvdash_{\mathcal{N}} \neg A \quad \text{or equivalently} \quad \Gamma \vdash_{\mathcal{N}} A \textit{ or } \Gamma \vdash_{\mathcal{N}} \neg A \tag{6.20}$$

If Γ is maximal consistent it satisfies both (6.19) and (6.20) and hence, for any formula A, *exactly* one of $\Gamma \vdash_{\mathcal{N}} A$ and $\Gamma \vdash_{\mathcal{N}} \neg A$ is the case.

For instance, given $\Sigma = \{a, b\}$, the theory $\Gamma = \{a \to b\}$ is consistent. However, it is not maximal consistent because, for instance, $\Gamma \nvdash_{\mathcal{N}} a$ and $\Gamma \nvdash_{\mathcal{N}} \neg a$. (The same holds if we replace a by b.) In fact, we have an alternative, equivalent, and easier to check formulation of maximal consistency for PL.

Fact 6.21. A theory $\Gamma \subset \mathsf{WFF}^{\Sigma}_{\mathsf{PL}}$ is maximal consistent iff it is consistent and for all $a \in \Sigma : \Gamma \vdash_{\mathcal{N}} a$ or $\Gamma \vdash_{\mathcal{N}} \neg a$.

Proof. 'Only if' part, i.e. $\Rightarrow$, is trivial from definition 6.18 which ensures that $\Gamma \vdash_{\mathcal{N}} A$ or $\Gamma \vdash_{\mathcal{N}} \neg A$ for all formulae, in particular all atomic ones.

The opposite implication is shown by induction on the complexity of A.

<u>A IS :</u>

$a \in \Sigma$ :: This basis case is trivial, since it is exactly what is given.

$\neg B$ :: By IH, we have that $\Gamma \vdash_{\mathcal{N}} B$ or $\Gamma \vdash_{\mathcal{N}} \neg B$. In the latter case, we are done ($\Gamma \vdash_{\mathcal{N}} A$), while in the former we obtain $\Gamma \vdash_{\mathcal{N}} \neg A$, i.e., $\Gamma \vdash_{\mathcal{N}} \neg\neg B$ from lemma 4.10.

$C \to D$ :: By IH we have that either $\Gamma \vdash_{\mathcal{N}} D$ or $\Gamma \vdash_{\mathcal{N}} \neg D$. In the former case, we obtain $\Gamma \vdash_{\mathcal{N}} C \to D$ by lemma 4.9. In the latter case, we have to consider two subcases – by IH either $\Gamma \vdash_{\mathcal{N}} C$ or $\Gamma \vdash_{\mathcal{N}} \neg C$. If $\Gamma \vdash_{\mathcal{N}} \neg C$ then, by lemma 4.9, $\Gamma \vdash_{\mathcal{N}} \neg D \to \neg C$. Applying MP to this and axiom A3, we obtain $\Gamma \vdash_{\mathcal{N}} C \to D$. So, finally, assume $\Gamma \vdash_{\mathcal{N}} C$ (and $\Gamma \vdash_{\mathcal{N}} \neg D$). But then $\Gamma \vdash_{\mathcal{N}} \neg(C \to D)$ by exercise 4.1.3.

QED (6.21)

The maximality property of a maximal consistent theory makes it easier to construct a model for it. We prove first this special case of the completeness theorem:

Lemma 6.22 Every maximal consistent theory is satisfiable.

Proof. Let Γ be any maximal consistent theory, and let Σ be the set of propositional variables. We define the valuation $V : \Sigma \to \{\mathbf{1}, \mathbf{0}\}$ by the equivalence

$$V(a) = \mathbf{1} \text{ iff } \Gamma \vdash_{\mathcal{N}} a$$

for every $a \in \Sigma$. (Hence also $V(a) = \mathbf{0}$ iff $\Gamma \nvdash_{\mathcal{N}} a$.)

We now show that V is a model of Γ, i.e., for any formula B : if $B \in \Gamma$ then $\widehat{V}(B) = \mathbf{1}$. In fact we prove the stronger result that for any formula B,

$$\widehat{V}(B) = \mathbf{1} \text{ iff } \Gamma \vdash_{\mathcal{N}} B.$$

The proof goes by induction on (the complexity of) B.

<u>B IS :</u>

a :: Immediate from the definition of V.

$\neg C$ :: $\widehat{V}(\neg C) = \mathbf{1}$ iff $\widehat{V}(C) = \mathbf{0}$. By IH, the latter holds iff $\Gamma \nvdash_{\mathcal{N}} C$, i.e., iff $\Gamma \vdash_{\mathcal{N}} \neg C$.

$C \to D$ :: We consider two cases:

- $\widehat{V}(C \to D) = \mathbf{1}$ implies ($\widehat{V}(C) = \mathbf{0}$ or $\widehat{V}(D) = \mathbf{1}$). By the IH, this implies ($\Gamma \nvdash_{\mathcal{N}} C$ or $\Gamma \vdash_{\mathcal{N}} D$), i.e., ($\Gamma \vdash_{\mathcal{N}} \neg C$ or $\Gamma \vdash_{\mathcal{N}} D$). In the former case Exercise 4.1.1, and in the latter lemma 4.12.2 gives that $\Gamma \vdash_{\mathcal{N}} C \to D$.
- $\widehat{V}(C \to D) = \mathbf{0}$ implies $\widehat{V}(C) = \mathbf{1}$ and $\widehat{V}(D) = \mathbf{0}$, which by the IH imply $\Gamma \vdash_{\mathcal{N}} C$ and $\Gamma \nvdash_{\mathcal{N}} D$, i.e., $\Gamma \vdash_{\mathcal{N}} C$ and $\Gamma \vdash_{\mathcal{N}} \neg D$, which by exercise 4.1.3 and two applications of MP imply $\Gamma \vdash_{\mathcal{N}} \neg(C \to D)$, i.e., $\Gamma \nvdash_{\mathcal{N}} C \to D$.

QED (6.22)

Next we use this result to show that *every* consistent theory is satisfiable. What we need, is a result stating that every consistent theory is a subset of some maximal consistent theory.

Lemma 6.23 Every consistent theory can be extended to a maximal consistent theory.

Proof. Let Γ be a consistent theory, and let $\{a_0, \ldots, a_n\}$ be the set of propositional variables used in Γ. [The case when $n = \omega$ (is countably infinite) is treated in the small font within the square brackets.] Let

- $\Gamma_0 = \Gamma$
- $\Gamma_{i+1} = \begin{cases} \Gamma_i, a_i & \text{if this is consistent} \\ \Gamma_i, \neg a_i & \text{otherwise} \end{cases}$
- $\widehat{\Gamma} = \Gamma_{n+1} \quad [= \bigcup_{i<\omega} \Gamma_i, \text{ if } n = \omega]$

We show by induction on i that for any i, Γ_i is consistent.

BASIS :: $\Gamma_0 = \Gamma$ is consistent by assumption.

IND. :: Suppose Γ_j is consistent. If Γ_{j+1} is inconsistent, then from the definition of Γ_{j+1} we know that both Γ_j, a_j and $\Gamma_j, \neg a_j$ are inconsistent, hence by Deduction Theorem both $a_j \to \bot$ and $\neg a_j \to \bot$ are provable from Γ_j. Exercise 4.1.4 tells us that in this case both $\neg a_j$ and $\neg\neg a_j$ are provable from Γ_j, contradicting (see exercise 4.5) the assumption that Γ_j is consistent.

In particular, $\Gamma_n = \widehat{\Gamma}$ is consistent. [For the infinite case, we use the above proof, the fact that any finite subtheory of $\overline{\Gamma}$ must be contained in a Γ_i for some i, and then the compactness theorem 4.29.]
To finish the proof, we have to show that $\widehat{\Gamma}$ is not only consistent but also maximal, i.e., that for every A, $\widehat{\Gamma} \vdash_{\mathcal{N}} A$ or $\widehat{\Gamma} \vdash_{\mathcal{N}} \neg A$. But this was shown in Fact 6.21, and so the proof is complete. **QED** (6.23)

The completeness theorem is now an immediate consequence:

Theorem 6.24 Every consistent theory is satisfiable.

Proof. Let Γ be consistent. By lemma 6.23 it can be extended to a maximal consistent theory $\widehat{\Gamma}$ which, by lemma 6.22, is satisfiable, i.e., has a model V. Since $\Gamma \subseteq \widehat{\Gamma}$, this model satisfies also Γ. **QED** (6.24)

Corollary 6.25 Let $\Gamma \subseteq \mathsf{WFF}_{\mathsf{PL}}$ and $A \in \mathsf{WFF}_{\mathsf{PL}}$. Then
(1) $\Gamma \models A$ implies $\Gamma \vdash_{\mathcal{N}} A$, and
(2) $\models A$ implies $\vdash_{\mathcal{H}} A$.

Proof. In view of lemma 4.18, 2. follows from 1. But 1. follows from theorem 6.24 by lemma 6.17. **QED** (6.25)

The soundness and completeness results are gathered in

Corollary 6.26 For any $\Gamma \subseteq \mathsf{WFF}_{\mathsf{PL}}$, $A \in \mathsf{WFF}_{\mathsf{PL}} : \Gamma \models A \Leftrightarrow \Gamma \vdash_{\mathcal{N}} A$.

The proof of completeness could be formulated much simpler. We chose the above, more complicated formulation, because it illustrates a general technique, which is applicable in many situations. We will encounter the same form of argument, when proving completeness of first-order logic.

4.1: Some Applications of Soundness and Completeness

Having a sound and complete axiomatic system allows us to switch freely between the syntactic (concerning provability) and semantic (concerning validity) arguments – depending on which one is easier in a given context.

1. Is a formula valid?
In case of PL it is, typically, easiest to verify it by making the appropriate boolean table. However, we have also proved several formulae. Thus, for instance, asked whether $(A \to (B \to C)) \to ((A \to B) \to (A \to C))$ is valid, we have a direct answer – it is axiom $A2$ of $\mathcal{H}$ and thus, **by soundness** of $\mathcal{H}$, we can immediately conclude that the formula is valid.

2. Is a formula provable?
In theorem 4.30 we gave an argument showing decidability of membership in $\vdash_{\mathcal{N}}$. In a bit roundabout way, we transformed $\mathcal{N}$ expressions into corresponding $\mathcal{G}$ expressions, and used $\mathcal{G}$ to decide their derivability (which, we said, was equivalent to derivability in $\mathcal{N}$).

Corollary 6.26 gives us another, semantic, way of deciding membership in $\vdash_{\mathcal{N}}$. It says that $\mathcal{N}$-derivable formulae are exactly the ones which are valid. Thus, to see if $G = A_1, ..., A_n \vdash_{\mathcal{N}} B$ is derivable in $\mathcal{N}$ it suffices to see if $A_1, ..., A_n \models B$. Since G is derivable iff $G' = \vdash_{\mathcal{N}} A_1 \to (A_2 \to ...(A_n \to B)...)$ is (lemma 4.19), the problem can be decided by checking if G' is valid. But this is trivial! Just make the boolean table for G', fill out all the rows and see if the last column contains only **1**. If it does, G' is valid and so, **by completeness**, derivable. If it does not (contains some **0**), G' is not valid and, **by soundness**, is not derivable.

Example 6.27
Is $\vdash_{\mathcal{N}} A \to (B \to (B \to A))$? We make the boolean table:

A	B	$B \underline{\to} A$	$B \underline{\to} (B \underline{\to} A)$	$A \underline{\to} (B \underline{\to} (B \underline{\to} A))$
1	**1**	**1**	**1**	**1**
1	**0**	**1**	**1**	**1**
0	**1**	**0**	**0**	**1**
0	**0**	**1**	**1**	**1**

The table tells us that $\models A \to (B \to (B \to A))$ and thus, **by completeness of** $\mathcal{N}$, we conclude that the formula is derivable in $\mathcal{N}$.

Now, is $\vdash_{\mathcal{N}} B \to (B \to A)$? The truth table is like the one above without the last column. The formula is not valid (third row gives **0**), $\not\models B \to (B \to A)$ and thus, **by soundness of** $\mathcal{N}$, we can conclude that it is not derivable in $\mathcal{N}$. □

Notice that to *decide* provability by such a reference to semantics we need both properties – completeness guarantees that whatever is valid is provable, while soundness that whatever is not valid is not provable.

This application of soundness/completeness is not typical because, usually, axiomatic systems are designed exactly to faciliate answering the more complicated question about validity of formulae by proving them. In PL, however, the semantics is so simple and decidable that it is easier to work with it directly than using respective axiomatic systems (except, perhaps, for $\mathcal{G}$).

3. Is a rule admissible?

For instance, is the rule $\dfrac{\vdash_{\mathcal{N}} A \to B \;\; ; \;\; \vdash_{\mathcal{N}} \neg B}{\vdash_{\mathcal{N}} \neg A}$ admissible in $\mathcal{N}$?

First, we have to verify if the rule itself is sound. So let V be an arbitrary structure (valuation) such that $V \models A \to B$ and $V \models \neg B$. From the latter we have that $V(B) = \mathbf{0}$ and so, using the definition of $\underline{\to}$, we obtain that since $V(A \to B) = \mathbf{1}$, we must have $V(A) = \mathbf{0}$. This means that $V \models \neg A$. Since V was arbitrary, we conclude that the rule is sound.

Now comes the application of soundess/completeness of $\mathcal{N}$. If $\vdash_{\mathcal{N}} A \to B$ and $\vdash_{\mathcal{N}} \neg B$ then, **by soundness of** $\mathcal{N}$, we also have $\models A \to B$ and $\models \neg B$. Then, by soundness of the rule itself, $\models \neg A$. And finally, **by completeness of** $\mathcal{N}$, this implies $\vdash_{\mathcal{N}} \neg A$. Thus, the rule is, indeed, admissible in $\mathcal{N}$, even though we have not shown how exactly the actual proof of $\neg A$ would be constructed. This form of an argument can be applied to show that *any sound* rule, will be admissible in a sound and complete axiomatic system.

On the other hand, if a rule is not sound, the soundness of the axiomatic system immediately implies that the rule will not be admissible in it. For instance, the rule $\dfrac{\vdash_{\mathcal{N}} A \to B \;\; ; \;\; \vdash_{\mathcal{N}} B}{\vdash_{\mathcal{N}} A}$ is not sound (verify it – find a valuation making both premises true and the conclusion false). **By soundness** of $\mathcal{N}$, we may conclude that it isn't admissible there.

Exercises 6.

EXERCISE 6.1 Translate the following argument into a formula of PL and show *both* semantically *and* syntactically (assuming soundness and completeness of any of the reasoning systems we have seen) – that it is a tautology:

- If I press the gas pedal and turn the key, the car will start. Hence, either if I press the gas pedal the car will start or if I turn the key the car will start.

The confusion arises from the fact that in the daily language there is no clear distinction between $\rightarrow$ and $\Rightarrow$. The immediate understanding of the implications in the conclusions of these arguments will interpret them as $\Rightarrow$ rather than $\rightarrow$. This interpretation makes them sound absurd.

EXERCISE 6.2 Define binary connective $\downarrow$ (so called, Sheffer's stroke) as follows:

x	y	$x{\downarrow}y$
1	**1**	**0**
1	**0**	**0**
0	**1**	**0**
0	**0**	**1**

Show that $\{\downarrow\}$ is an adequate set. (Hint: Express some set you know is adequate using $\downarrow$.)

EXERCISE 6.3 Show that $\{\vee, \wedge\}$ is not an adequate set.

EXERCISE 6.4 Let $\underline{F}$, $\underline{G}$ be two boolean functions given by

- $\underline{F}(x, y) = \mathbf{1}$ iff $x = \mathbf{1}$,
- $\underline{G}(x, y, z) = \mathbf{1}$ iff $y = \mathbf{1}$ or $z = \mathbf{0}$.

Write each of these functions as formulae in CNF and DNF.

EXERCISE 6.5 Find DNF and CNF formulae inducing the same boolean functions as the formulae:

(1) $(\neg a \wedge b) \rightarrow c$
(2) $(a \rightarrow b) \wedge (b \rightarrow c) \wedge (a \vee \neg c)$
(3) $(\neg a \vee b) \vee (a \vee (\neg a \wedge \neg b))$

[Recall first exercise 5.10. You may follow the construction from the proof of Theorem 6.2. But you may just use the laws which you have learned so far to perform purely syntactic manipulations. Which of these two ways is simpler?]

EXERCISE 6.6 Let $\Sigma = \{a, b, c\}$ and consider two sets of formulae $\Delta = \{a \wedge (a \rightarrow b)\}$ and $\Gamma = \{a, a \rightarrow b, \neg c\}$. Give an example of

- an $A \in \mathsf{WFF}^{\Sigma}_{\mathsf{PL}}$ such that $\Delta \not\models A$ and $\Gamma \models A$
- a model (valuation) V such that $V \models \Delta$ and $V \not\models \Gamma$.

EXERCISE 6.7 Let $\Sigma = \{a, b, c\}$. Which of the following sets of formulae are maximal consistent?

(1) $\{a, b, \neg c\}$
(2) $\{a, b, c\}$
(3) $\{\neg b \rightarrow a, \neg a \vee c\}$
(4) $\{\neg a \vee b, b \rightarrow c, \neg c\}$
(5) $\{\neg a \vee b, b \rightarrow c, \neg c, a\}$

EXERCISE 6.8 Show that Γ is maximal-consistent (in $\mathcal{N}$) iff it has only one model.

EXERCISE 6.9 We consider the Gentzen system for PL.

We define the interpretation of sequents as follows. A structure (valuation) V *satisfies* the sequent $\Gamma \vdash_{\mathcal{G}} \Delta$, $V \models \Gamma \vdash_{\mathcal{G}} \Delta$, iff either there is a formula $\gamma \in \Gamma$ such that $V \not\models \gamma$, or there is a formula $\delta \in \Delta$ such that $V \models \delta$. The definition can be abbreviated as requiring that $V \models \bigwedge\Gamma \rightarrow \bigvee\Delta$ or, writing explicitly the sequent

$$V \models \gamma_1, ..., \gamma_g \vdash_{\mathcal{G}} \delta_1, ..., \delta_d \quad \Longleftrightarrow \quad V \models \gamma_1 \wedge ... \wedge \gamma_g \rightarrow \delta_1 \vee ... \vee \delta_d.$$

A sequent $\Gamma \vdash_{\mathcal{G}} \Delta$ is valid iff it is satisfied under every valuation, i.e., iff $\bigwedge\Gamma \Rightarrow \bigvee\Delta$.

We have remarked that the same formulae are provable whether in the axioms we require only atoms or allow general formulae. Here we take the version with *only* atomic formulae in the axioms (i.e., axioms are seqeuents $\Gamma \vdash_{\mathcal{G}} \Delta$ with $\Gamma \cap \Delta \neq \varnothing$ and where all formulae in $\Gamma \cup \Delta$ are atomic (i.e., propositional variables)). Also, we consider only the basic system with the rules for the connectives $\neg$ and $\rightarrow$.

(1) (a) Say (in *only one* sentence!) why the axioms of $\vdash_{\mathcal{G}}$ are valid, i.e., why every valuation V satisfies every axiom.
(b) Given an irreducible sequent S (containing only atomic formulae) which is *not* an axiom, describe a valution V making $V \not\models S$ (a so called "counter-model" for S).
(2) Verify that the rules are *invertible*, that is, are sound in the direction "bottom up": for any valuation V, if V satisfies the conclusion of the rule, then V satisfies *all* the premises.
(3) Suppose that $\Gamma \not\vdash_{\mathcal{G}} \Delta$. Then, at least one of the branches of the proof (built bottom-up from this given sequent) ends with a non-axiomatic sequent, S (containing only atoms). Hence, as shown in point 1.(b), S has a counter-model.
(a) On the basis of (one of) the above points, explain (in *only one* sentence!) why this counter-model V for S will also be a counter-model for $\Gamma \vdash_{\mathcal{G}} \Delta$, i.e., why it will be the case that $V \not\models \bigwedge\Gamma \rightarrow \bigvee\Delta$.
(b) You have actually proved completeness of the system $\vdash_{\mathcal{G}}$. Explain (in *one* sentence or formula) why one can claim that.

optional

EXERCISE 6.10 Apply de Morgan's laws to show directly (without using corollaries 6.6-6.7) that

(1) if A is in CNF then $\neg A$ is equivalent to a formula in DNF
(2) if A is in DNF then $\neg A$ is equivalent to a formula in CNF

EXERCISE 6.11 The following rules, called (respectively) the *constructive* and *destructive* dilemma, are often handy in constructing proofs

$$\text{(CD)} \quad \frac{A \vee B \ ; \ A \to C \ ; \ B \to D}{C \vee D} \qquad \text{(DD)} \quad \frac{\neg C \vee \neg D \ ; \ A \to C \ ; \ B \to D}{\neg A \vee \neg B}$$

Show that they are sound, i.e., in any structure V which makes the assumptions of (each) rule true, the (respective) conclusion is true as well. Give an argument that these rules are admissible in $\mathcal{N}$. (If the syntax $X \vee Y$ seems confusing, it can be written as $\neg X \to Y$.)

EXERCISE 6.12 [Galois connections]

A) We used the word "theory" for an arbitrary set of formulae. Another, common, usage distinguishes between such a set – calling it *non-logical axioms* – and its closure under $\vdash_{\mathcal{C}}$ which is called a theory (relatively to some given proof system $\mathcal{C}$). In remark 6.13 we defined this set – the theory of Γ – as $Th_{\mathcal{C}}(\Gamma) = \{B \in \mathsf{WFF} : \Gamma \vdash_{\mathcal{C}} B\}$. Similarly, the models of Γ are defined as $Mod(\Gamma) = \{V : V \models \Gamma\}$. As an example, we can instantiate the generic notions of $\vdash$ and $\models$ to $\mathcal{N}$, obtaining:

- $Th_{\mathcal{N}}(\Gamma) = \{B \in \mathsf{WFF}_{\mathsf{PL}} : \Gamma \vdash_{\mathcal{N}} B\}$
- $Mod(\Gamma) = \{V : \Sigma \to \{\mathbf{1}, \mathbf{0}\} : \widehat{V}(G) = \mathbf{1} \text{ for all } G \in \Gamma\}$

Show the following statements:

(1) $\Delta \subseteq \Gamma \Rightarrow Th_{\mathcal{N}}(\Delta) \subseteq Th_{\mathcal{N}}(\Gamma)$
(2) $\Delta \subseteq \Gamma \Rightarrow Mod(\Delta) \supseteq Mod(\Gamma)$

B) Notice that we can view Mod as a function $Mod : \wp(\mathsf{WFF}) \to \wp(Str)$ assigning to a theory Γ (a set of non-logical axioms) the set of all *Str*uctures (valuations) which satisfy all the axioms from Γ. On the other hand, we can define a function $Th : \wp(Str) \to \wp(\mathsf{WFF})$ assigning to an arbitrary set of *Str*uctures the set of all formulae satisfied by all these structures. (This Th should not be confused with $Th_{\mathcal{C}}$ above which was defined relatively to a proof system!) That is:

- $Mod : \wp(\mathsf{WFF}) \to \wp(Str)$ is defined by $Mod(\Gamma) \stackrel{\text{def}}{=} \{V : V \models \Gamma\}$
- $Th : \wp(Str) \to \wp(\mathsf{WFF})$ is defined by $Th(K) \stackrel{\text{def}}{=} \{A : K \models A\}$

Assuming a sound and complete proof system $\mathcal{C}$, show that

(1) $Th_{\mathcal{C}}(\Gamma) = Th(Mod(\Gamma))$

(2) What would you have to change in the above equation if you merely knew that $\mathcal{C}$ is sound (but not complete) or complete (but not sound)?

Now, show that these two functions form the so called "Galois connection", i.e. satisfy:

(3) $K \subseteq L \Rightarrow Th(K) \supseteq Th(L)$
(4) $\Delta \subseteq \Gamma \Rightarrow \mathit{Mod}(\Delta) \supseteq \mathit{Mod}(\Gamma)$
(5) $K \subseteq Mod(Th(K))$
(6) $\Gamma \subseteq Th(Mod(\Gamma))$

Chapter 7
Syntax and Proof System of FOL

- Syntax of FOL
- Proof System $\mathcal{N}$ for FOL
- Gentzen's Proof System $\mathcal{G}$ for FOL

◊——————a Background Story——————◊

Statement logic is a rudimentary system with very limited expressive power. The only relation between various statements it can handle is that of identity and difference – the axiom schema $\vdash_{\mathcal{H}} A \to (B \to A)$ sees merely that the first and last statements must be identical while the middle one is arbitrary (possibly different from A). Consider, however, the following argument:

	A:	`Every man is mortal;`
and	B:	`Socrates is a man;`
hence	C:	`Socrates is mortal.`

Since all the involved statements are different, the representation in PL *will amount to $A \wedge B \to C$ which, obviously, is not a valid formula. The validity of this argument rests on the fact that the minor premise B makes Socrates, so to speak, an instance of the subject of the major premise A – whatever applies to every man, applies to each particular man.*

In this particular case, one might try to refine the representation of the involved statements a bit. Say that we use the following propositional variables: Man (for 'being a man'), Mo (for 'being mortal') and So (for 'being Socrates'). Then we obtain: $So \to Man$ and $Man \to Mo$, which does entail $So \to Mo$. We were lucky! - because the involved statements had rather simple structure. If we, for instance, wanted to say that "Some men are thieves" in addition to "All men are mortal", we could hardly use the same Man in both cases. The following argument, too, is very simple, but it illustrates the need for talking not only about atomic statements, but also about involved enitites and relations between them:

	`Every horse is an animal;`
hence	`Every head of a horse is a head of an animal.`

Its validity relies not so much on the form, let alone identity and difference, of the involved statements as on the relations between the involved entities, in particular, that of 'being a head of...' Since each horse is an animal, whatever applies to animals (or their heads) applies to horses as well. It is hard to imagine how such an argument might possibly be represented (as a valid statement) in PL.

Intuitiviely, the semantic "view of the world" underlying FOL can be summarised as follows.

(1) We have a universe of discourse U comprising all entities of (current) interest.
(2) We may have particular means of picking some entities.
   (a) For instance, a name, 'Socrates' can be used to designate a particular individual: Such names correspond to constants, or functions with 0 arguments.
   (b) An individual may also be picked by saying 'the father of Socrates'. Here, 'the father of ...' (just like 'the head of ...') is a function taking 1 argument (preferably a man, but generally an arbitrary entity from U, since the considered functions are total) and pointing at another individual. Functions may have arbitrary arities, e.g., 'the children of x and y' is a function of 2 arguments returning a set of children, 'the solutions of $x^2 - y^2 = 0$' is a function of 2 arguments returning a set of numbers.
   (c) To faciliate flexible means of expression, we may use variables, x, y, etc. to stand for arbitrary entities in more complex expressions.
(3) The entities from U can be classified and related using various predicates and relations:
   (a) We may identify subsets of U by means of (unary) predicates: the predicate $M(y)$ – true about those y's which are men, e.g. $M(Socrates)$, but not about inanimate things – identifies a subset of those elements of U about which it is true, i.e., $\{y \in U : M(y)\}$; the predicate $Mo(y)$ – true about the mortal beings and false about all other entities – identifies the subset of mortal beings; $H(y)$ – the subset of horses, $A(y)$ – the animals, etc.

Definition 7.1 (iii) The entities may stand in various relations to each other: 'x is older than y' is a 2-argument relation which will hold between some individuals but not between others; similarly for 'x is the head of y', $Hd(x, y)$, etc.

(4) $Hd(x, y)$, stating that 'x is the head of y', is very different from the function $hd(y)$ returning, for every y, its head. The latter merely picks new individual objects. The former is a predicate – it states some fact. Predicates and relations are the means for stating the "atomic facts" about the world.[6] These can be then combined using the connectives, as in PL, 'and', 'or', 'not', 'if...then...', etc. In addition, we may also state facts about indefinite entities, for instance, 'for-every $x : M(x) \to Mo(x)$', 'for-no $x : H(x) \wedge M(x)$', etc.

Agreeing on such an interpretation of the introduced symbols, the opening arguments would be written :

$$\begin{array}{lll} \text{A:} & \texttt{for-every}\ y: & M(y) \to Mo(y) \\ \text{B:} & & M(Socrates) \\ \hline \text{C:} & & Mo(Socrates) \end{array}$$

$$\begin{array}{l} \texttt{for-every}\ y : H(y) \to A(y) \\ \hline \texttt{for-every}\ x : (\texttt{there-is}\ y : H(y) \wedge Hd(x,y)) \ \to (\texttt{there-is}\ y : A(y) \wedge Hd(x,y)) \end{array}$$

This illustrates the intention of the language of FOL, which we now begin to study, and its semantics which will be our object in the following chapters.

◇——————————◇

1: Syntax of FOL

In PL the non-logical (i.e., relative to the context of application) part of the language was only the set Σ of propositional variables. In FOL this part is much richer which also means that, in a context of particular application, the user can – and has to – make more detailed choices. Nevertheless, this non-logical part of the language has well defined components which will still make it possible to treat FOL in a uniform way, relatively independent of such contextual choices.

[6]For this reason, First Order Logic is also called "Predicate Logic"

The alphabet of predicate logic consist of two disjoint sets Σ and Φ, where

Σ : the non-logical alphabet contains non-logical symbols:

- *individual constants*: $\mathcal{I} = \{a, b, c...\}$
- *individual variables*: $\mathcal{V} = \{x, y, z...\}$
- *function symbols*: $\mathcal{F} = \{f, g, h...\}$ each taking a fixed finite number of arguments, called its *arity*.
- *relation symbols*: $\mathcal{R} = \{P, Q, R...\}$, each with a fixed arity.

Φ : contains the logical symbols:

- the connectives of PL : $\neg, \rightarrow$
- *quantifier*: $\exists$

We make no assumption about the sizes of $\mathcal{I}, \mathcal{F}, \mathcal{R}$; any one of them may be infinite, finite or empty. $\mathcal{V}$, on the other hand, is always a countably infinite set.

We also use some auxiliary symbols like parentheses and commas. Relation symbols are also called predicate symbols or just predicates. Individual variables and constants are usually called just variables and constants.

Example 7.2

Suppose we want to talk about stacks – standard data structures. We might start by setting up the following alphabet Σ_{Stack} (to indicate arities, we use the symbol U for the whole universe):

(1) $\mathcal{I} = \{empty\}$ – the only constant for representing empty stack;
(2) $\mathcal{V} = \{x, y, s, u, v, ...\}$ – we seldom lists these explicitly; just mark that something is a variable whenever it is used;
(3) $\mathcal{F} = \{top : U \rightarrow U,\ pop : U \rightarrow U,\ push : U^2 \rightarrow U\}$;
(4) $\mathcal{R} = \{St \subseteq U,\ El \subseteq U,\ \equiv\ \subseteq U^2\}$ – for identifying *St*acks, *El*ements, and for expressing equality.

□

Unlike PL, the language of FOL is designed so that we may write not only formulae but also terms. The former, as before, will denote some boolean values. Terms are ment to refer to “individuals” or some “objects” of the “world”. Formulae will be built using propositional connectives, as in PL, from simpler expressions – atomic formulae – which, however, are not merely propositional variables but have some internal structures involving terms. In addition, we will have a new formula-building operation for quantification over individual variables.

Definition 7.3 [**Terms**] The set of *terms* over Σ, $\mathcal{T}_\Sigma$, is defined inductively:

(1) all constants are in, $\mathcal{I} \subseteq \mathcal{T}_\Sigma$.
(2) all variables are in, $\mathcal{V} \subseteq \mathcal{T}_\Sigma$.
(3) if $f \in \mathcal{F}$ is an n-ary function symbol and $t_1, \ldots, t_n$ are in $\mathcal{T}_\Sigma$, then $f(t_1, \ldots, t_n) \in \mathcal{T}_\Sigma$.

Terms not containing any variables are called *ground terms.* The set of ground terms is denoted $\mathcal{GT}_\Sigma$.

Example 7.4
Consider the alphabet of stacks from Example 7.2. The only ground terms are the constant *empty* and applications of functions to it, e.g., $pop(empty)$, $top(empty)$, $pop(pop(empty))$. Notice, that also terms like $push(empty, empty)$, $push(empty, pop(empty))$, etc. are well-formed ground terms, even if they do not necessarily correspond to our intensions.

The non ground terms will be of the same kind but will involve variables, say x, s. For instance, $pop(s)$, $top(pop(s))$, $push(empty, x)$, $pop(push(x, empty))$, etc. □

Definition 7.5 **[Formulae]** The well-formed formulae of predicate logic over a given Σ, $\mathsf{WFF}^\Sigma_{\mathsf{FOL}}$, are defined inductively:

(1) If $P \in \mathcal{R}$ is an n-ary relation symbol and $t_1, \ldots, t_n$ are terms, then $P(t_1, \ldots, t_n) \in \mathsf{WFF}^\Sigma_{\mathsf{FOL}}$.
(2) If $A \in \mathsf{WFF}^\Sigma_{\mathsf{FOL}}$ and x is a variable, then $\exists x A$ is in $\mathsf{WFF}^\Sigma_{\mathsf{FOL}}$.
(3) If $A, B \in \mathsf{WFF}^\Sigma_{\mathsf{FOL}}$ then $\neg A, (A \rightarrow B) \in \mathsf{WFF}^\Sigma_{\mathsf{FOL}}$

Formulae from point 1. are called *atomic*, those from 2. are *quantified.* Thus the FOL language "refines" the PL language in that propositional connectives connect not just propositional variables but more detailed atomic or quantified formulae.

Remark.
As in the case of PL, these definitions are really parameterised by Σ, yielding a new instance of the FOL language for each particular choice of Σ. Nevertheless we will often speak about "the FOL language," taking Σ as an implicit, and arbitrary though non-empty, parameter.

Example 7.6
Continuing our example of stacks, atomic formulae will be, for instance: $El(empty)$, $St(x)$, $empty \equiv pop(empty)$, $push(s, x) \equiv pop(pop(empty))$, etc. The non-atomic ones are simply boolean combinations of atoms, e.g., $El(x) \wedge St(s) \rightarrow pop(push(s, x)) \equiv s$. □

1.1: Abbreviations

The symbol $\exists$ is called the *existential quantifier* – $\exists xA$ reads as "there exists an x such that A". For convenience, we define the following abbreviation:

Definition 7.7 We define $\forall xA \stackrel{\text{def}}{=} \neg\exists x\neg A$.

$\forall$ is the *universal quantifier* and $\forall xA$ is read "for all x A". Writing $\mathsf{Q}xA$, we will mean any one of the two quantifiers, i.e., $\forall xA$ or $\exists xA$.

We will, of course, use also the abbreviations $\vee$ and $\wedge$ for propositional connectives. Sometimes (when it is "safe") the arguments to $\vee$ and $\wedge$ will be written without the surrounding parentheses. Similarly for $\rightarrow$.

2: Scope of Quantifiers, Free Variables, Substitution

Definition 7.8 Consider the formula $\mathsf{Q}xB$. We say that B is the *scope* of $\mathsf{Q}x$

Example 7.9
To make the scope of quantifiers unambiguous we use parantheses whenever necessary. The scopes of the various quantifiers are underlined:

	in formula	the scope			
1.	$\forall x\underline{\exists y\underline{\underline{R(x,y)}}}$	of	$\exists y$	is	$R(x,y)$
		of	$\forall x$	is	$\exists yR(x,y)$
2.	$\forall x(\underline{R(x,y)\ \wedge\ R(x,x)})$	of	$\forall x$	is	$R(x,y) \wedge R(x,x)$
3.	$\forall x\underline{R(x,y)}\ \wedge\ R(x,x)$	of	$\forall x$	is	$R(x,y)$
4.	$\forall x\underline{R(x,x)} \rightarrow \exists y\underline{Q(x,y)}$	of	$\forall x$	is	$R(x,x)$
		of	$\exists y$	is	$Q(x,y)$
5.	$\forall x(\underline{R(x,x) \rightarrow \exists y\underline{\underline{Q(x,y)}}})$	of	$\exists y$	is	$Q(x,y)$
		of	$\forall x$	is	$R(x,x) \rightarrow \exists yQ(x,y)$
6.	$\forall x(\underline{R(x,x) \rightarrow \exists x\underline{\underline{Q(x,x)}}})$	of	$\exists x$	is	$Q(x,x)$
		of	$\forall x$	is	$R(x,x) \rightarrow \exists xQ(x,x)$

$\square$

Definition 7.10 For any formula A we say that

- An *occurrence* of a variable x which is not within the scope of any quantifier $\mathsf{Q}x$ in A is *free* in A.

- An occurrence of a variable x which is not free in A is said to be *bound*. Moreover, it is bound *by* the innermost (i.e., closest to the left) quantifier of the form Qx inside the scope of which it occurs,
- A is *closed* if it has no free occurrences of any variable. (We also say that A is a *sentence*.)
- A is *open* if it is not closed.

For any A, $\mathcal{V}(A)$ is the set of variables with free occurrences in A. Thus A is closed iff $\mathcal{V}(A) = \varnothing$.

Example 7.11
In example 7.9, y is *free* in 2. and 3. In 3. the occurrences of x in $R(x, x)$ are free too, but the occurrence of x in $R(x, y)$ is *bound*. Similarly, in 4. the occurrences of x in $R(x, x)$ are *bound*, but the one in $Q(x, y)$ is *free*. In 5. all occurrences of x are bound by the frontmost $\forall x$. In 6., however, the occurrences of x in $R(x, x)$ are bound by the frontmost $\forall x$, but the ones in $Q(x, x)$ are bound by the $\exists x$. Thus 2., 3. and 4. are open formulae while the others are closed. $\square$

Remark 7.12 **[Analogy to programming]**
As we will see in next chapter, the difference between bound and free variables is that the names of the former do not make any difference while of the latter do influence the interpretation of formulae. As a convenient analogy, one may think about free variables in a formula A as global variables in an imperative program A. The bound variables correspond to the local variables and quantifier to a block with declaration of local variables. For instance, in the following program `P1` on the left

```
P1: begin                          P2:   begin
      int x,y;                             int x,y;
      x:=5; y:=10;                         x:=5; y:=10;
      begin                                begin
        int x,z;                             int w,z;
        x:=0; x:=x+3;                        w:=0; w:=w+3;
        z:=20; y:=30;                        z:=20; y:=30;
      end;                                 end;
    end;                                 end;
```

the global variable x is redeclared in the inner block. This can be said to make the global x "invisible" within this block. y is another global variable, while z is a local variable in the block. At the exit, we will have $x = 5$ and $y = 30$ since these global variables are not affected by the assignment to the local ones within the block. Also, z will not be available after the exit from the inner block.

A formula with a similar scoping effect would be:

$$A(x,y) \wedge \mathsf{Q}x\mathsf{Q}zB(x,z,y) \quad \text{or alternatively} \quad \mathsf{Q}x\mathsf{Q}y(\ A(x,y) \wedge \mathsf{Q}x\mathsf{Q}zB(x,z,y)\ ) \tag{7.13}$$

where we ignore the meaning of the predicates A, B but concentrate only on the "visibility", i.e., scope of the quantifiers. Variable y is free (on the left, and within the scope of the same quantifier on the right) and thus its occurrence in $B(...y)$ corresponds to the same entity as its occurrence in $A(...y)$. On the other hand, x in $A(x...)$ – in the outermost block – is one thing, while x in $B(x...)$ – in the inner block – a completely different one, since the latter is in the scope of the innermost quantifier $\mathsf{Q}x$.

As one would expect, the program `P2` on the right is equivalent to `P1`, the only difference being the renaming of local x to w. In fact, the formula (7.13) will be equivalent to the one where the bound x has been renamed, e.g., to the following one

$$A(x,y) \wedge \mathsf{Q}w\mathsf{Q}zB(w,z,y) \quad \text{or alternatively} \quad \mathsf{Q}x\mathsf{Q}y\ (\ A(x,y) \wedge \mathsf{Q}w\mathsf{Q}zB(w,z,y)\ ) \tag{7.14}$$

Renaming of free variables will not be allowed in the same way.

At this point, the distinction free vs. bound may, and probably does, seem unclear – in any case with respect to it possible meaning and motivation. So, for now, it is best to accept the Definition 7.10 at its face value. It makes it at least clear and easy to distinguish free and bounded occurrences by simple syntactic check as illustrated in Example 7.9.

2.1: Some examples

Before we begin a closer discussion of the syntax of FOL, we give a few examples of vocabularies (alphabets) which can be used for describing some known structures. Notation $\mathsf{Q}x : A$, resp. $(\mathsf{Q}x : A)$ indicates that A is the scope of $\mathsf{Q}x$.

Example 7.15 **[Stacks]**
Using the alphabet of stacks from Example 7.2, we may now set up the following (non-logical) axioms Γ_{Stack} for the theory of stacks ($x, s \in \mathcal{V}$):

(1) $St(empty)$
(2) $\forall x, s : El(x) \wedge St(s) \to St(push(s,x))$
(3) $\forall s : St(s) \to St(pop(s))$
(4) $\forall s : St(s) \to El(top(s))$

These axioms describe merely the profiles of the involved functions and determine the extension of the respective predicates. According to the first axiom *empty* is a stack, while the second axiom says that if x is an element and s is a stack then also the result of $push(s,x)$ is stack. (Usually, one uses some abbreviated notation to capture this information. In typed programming languages, for instance, it is taken care of by the typing system.) The further (non-logical) axioms detemining more specific properties of stacks would then be:

(5) $pop(empty) \equiv empty$
(6) $\forall x,s : El(x) \wedge St(s) \rightarrow pop(push(s,x)) \equiv s$
(7) $\forall x,s : El(x) \wedge St(s) \rightarrow top(push(s,x)) \equiv x$

Notice that we have not given any axioms which would ensure that $\equiv$ behaves as the identity relation. We will discuss the issue of identity in a later chapter, so here we merely list the needed axioms (the first three make $\equiv$ an equivalence relation):

(8) $\forall x : x \equiv x$
(9) $\forall x,y : x \equiv y \rightarrow y \equiv x$
(10) $\forall x,y,z : x \equiv y \wedge y \equiv z \rightarrow x \equiv z$

and two axiom schemata:

(11) for any n-ary $f \in \mathcal{F}$: $\forall x_1, x_1' ... x_n, x_n' : x_1 \equiv x_1' \wedge ... \wedge x_n \equiv x_n' \rightarrow f(x_1...x_n) \equiv f(x_1'...x_n')$
(12) for any n-ary $R \in \mathcal{R}$: $\forall x_1, x_1' ... x_n, x_n' : x_1 \equiv x_1' \wedge ... \wedge x_n \equiv x_n' \wedge R(x_1...x_n) \rightarrow R(x_1'...x_n')$

□

Example 7.16 **[Queues]**
We use the same alphabet as for stacks, although we intend different meaning to some symbols. Thus St is now to be interpreted as the set of (FIFO) queues, *pop* is to be interpreted as *tail*, *push* as *add* (at the end) and *top* as the head, the frontmost element of the queue. We only need to replace the axioms 6-7 with the following:

6a. $\forall x,s : El(x) \wedge St(s) \wedge s \equiv empty \rightarrow pop(push(s,x)) \equiv s$
6b. $\forall x,s : El(x) \wedge St(s) \wedge \neg(s \equiv empty) \rightarrow pop(push(s,x)) \equiv push(pop(s),x)$
7a. $\forall x,s : El(x) \wedge St(s) \wedge s \equiv empty \rightarrow top(push(s,x)) \equiv x$
7b. $\forall x,s : El(x) \wedge St(s) \wedge \neg(s \equiv empty) \rightarrow top(push(s,x)) \equiv top(s)$

□

Example 7.17 **[Graphs]**
All axioms in the above two examples were universal formulae (with only

$\forall$-quantifier in front). We now describe graphs which require more specific formulae.

Graph is a structure with two sets: V – vertices, and E – edges. Each edge has a unique source and target vertex. Thus, we take as our non-logical alphabet Σ_{Graph} :

- $\mathcal{F} = \{sr, tr\}$ – for source and target functions
- $\mathcal{R} = \{V, E, \equiv\}$ – V, E unary predicates for the set of vertice and edges, and $\equiv$ for the identity relation.

The axiom

(1) $\forall e : E(e) \rightarrow (\ V(sr(e)) \wedge V(tr(e))\ )$

determines the profile of the functions sr and tr. This, in fact, is all that one need to say in order to get arbitrary graphs. Typically, we think of a graph with at most one edge between two vertices. For that case, we need to add the axiom:

(2) $\forall e_1, e_2 : (\ sr(e_1) \equiv sr(e_2) \wedge tr(e_1) \equiv tr(e_2)\ ) \rightarrow e_1 \equiv e_2$

The graphs, so far, are directed. An undirected graph can be seen as a directed graph where for any edge from x to y, there is also an opposite edge from y to x:

(3) $\forall x, y : (\exists e : sr(e) \equiv x \wedge tr(e) \equiv y) \rightarrow (\exists e : tr(e) \equiv x \wedge sr(e) \equiv y)$

In fact, the intension of the above formula could be captured by a simpler one:

$$\forall e_1 \exists e_2 : sr(e_1) \equiv tr(e_2) \wedge tr(e_1) \equiv sr(e_2).$$

Finally, we may make a special kind of *transitive* graphs: if there is an edge from x to y and from y to z, then there is also an edge from x to z, and this applies for any possible pair of edges:

(4) $\forall e_1, e_2 : \big(tr(e_1) \equiv sr(e_2) \rightarrow \exists e : (sr(e) \equiv sr(e_1) \wedge tr(e) \equiv tr(e_2))\big)$

$\Box$

Example 7.18 **[Simple graphs]**
If we want to consider only simple graphs, i.e., graphs satisfying the axiom 2 from the previous example, we can choose a much more convenient vocabulary Σ_{SG}: our universe is the set of all possible vertices, we need no function symbols, and we use one binary relation symbol: E – the edge

relation. Axioms 1 and 2 become then redundant. If we want to consider undirected graphs, we make E symmetric:

(1) $\forall x, y : E(x, y) \rightarrow E(y, x)$.

If we want to consider transitive graphs, we make E transitive:

(2) $\forall x, y, z : E(x, y) \wedge E(y, z) \rightarrow E(x, z)$.

Graphs *without* self-loops (i.e., where no edge leads from a vertex to itself) are ones where E is irreflexive:

(3) $\forall x : \neg E(x, x)$

Using the representation from the previous example, this axiom would be: $\forall e : E(e) \rightarrow \neg(sr(e) \equiv tr(e))$. □

Notice that all our axioms are sentences (i.e., closed formulae).

2.2: Substitution

Definition 7.19 In a given term t/formula A, we may substitute the term s for the *free* occurrences of the variable x – t^x_s/A^x_s denote the resulting term/formula. The operations are defined inductively:

$$\begin{aligned}
x &:: x^x_s \stackrel{\text{def}}{=} s \\
y &:: y^x_s \stackrel{\text{def}}{=} y, \text{ for any } y \neq x \\
f(t_1, \ldots, t_k) &:: f(t_1, \ldots, t_k)^x_s \stackrel{\text{def}}{=} f({t_1}^x_s, \ldots, {t_k}^x_s).
\end{aligned}$$

This determines t^x_s for any t and x and s. Building further on this, we obtain the corresponding definition for *formulae*:

$$\begin{aligned}
\text{Atomic} &:: P(t_1, \ldots, t_k)^x_s \stackrel{\text{def}}{=} P({t_1}^x_s, \ldots, {t_k}^x_s) \\
\neg B &:: (\neg B)^x_s \stackrel{\text{def}}{=} \neg(B^x_s) \\
B \rightarrow C &:: (B \rightarrow C)^x_s \stackrel{\text{def}}{=} (B^x_s \rightarrow C^x_s) \\
\exists x A &:: (\exists x A)^x_s \stackrel{\text{def}}{=} \exists x A \\
\exists y A &:: (\exists y A)^x_s \stackrel{\text{def}}{=} \exists y (A^x_s), \text{ for any } y \neq x.
\end{aligned}$$

Example 7.20

In example 7.9 formulae 1., 5. and 6. had no free variables, so the application of any substitution will leave these formulae unchanged. For formulae 2.,

3. and 4. from that example, we obtain:

$$\begin{array}{rrcl} 2. & (\forall x(R(x,y) \wedge R(x,x)))^x_t & = & \forall x(R(x,y) \wedge R(x,x)) \\ 2'. & (\forall x(R(x,y) \wedge R(x,x)))^y_s & = & \forall x(R(x,s) \wedge R(x,x)) \\ 3. & (\forall x R(x,y) \wedge R(x,x))^x_t & = & \forall x R(x,y) \wedge R(t,t) \\ 4. & (\forall x R(x,x) \to \exists y Q(x,y))^x_t & = & \forall x R(x,x) \to \exists y Q(t,y) \end{array}$$

□

The following example shows that some caution is needed when for a variable we substitute a term that itself contains variables (or even *is* a variable). If such variables are "captured" by quantifiers already present, there may be unexpected results:

Example 7.21
As mentioned in remark 7.12, renaming bound variables results in equivalent formulae, e.g., the formulae $\forall y R(x,y)$ and $\forall z R(x,z)$ mean the same thing. However, performing the same substitution on the two produces formulae with (as we'll see later) very different interpretations:

1. $(\forall y R(x,y))^x_z = \forall y R(z,y)$ *or, e.g.* $(\exists y(x < y))^x_z = \exists y(z < y)$
2. $(\forall z R(x,z))^x_z = \forall z R(z,z)$ *or, e.g.* $(\exists z(x < z))^x_z = \exists z(z < z)$

The variable z is free throughout the first example, but then gets captured by $\forall z$ in the second example. To guard against such behaviour we introduce the next definition. Note that according to this definition, z is substitutable for x in $\forall y R(x,y)$ but not in $\forall z R(x,z)$. □

Definition 7.22 Let A be a formula, x a variable and s a term. The property "s is substitutable for x in A" is defined by induction on A as follows:

ATOMIC :: If A is atomic, then s is substitutable for x in A.
$\neg B$ :: s is substitutable for x in $\neg B$ iff s is substitutable for x in B.
$B \to C$ :: s is substitutable for x in $B \to C$ iff s is substitutable for x in both B and C.
$\exists x A$ :: s is substitutable for x in $\exists x A$. (Since no substitution in fact takes place.)
$\exists y A$ :: If $y \neq x$ then s is substitutable for x in $\exists y A$ iff either x does not occur free in A, or both s does not contain the variable y, and s is substitutable for x in A.

Simply put, s is substitutable for x in A (or the substitution A^x_s is *legal*) iff there are no free occurrences of x in A inside the scope of any quantifier that binds any variable occurring in s.

We may occasionally talk more generally about the *replacement* of an arbitrary term by another term, rather than just *substituting* a term for a variable.

3: Proof System

The ND proof system $\mathcal{N}$ for FOL uses the predicate $\vdash_{\mathcal{N}} \subseteq \wp(\mathsf{WFF_{FOL}}) \times \mathsf{WFF_{FOL}}$ and is an extension of the ND system $\mathcal{N}$ for PL.

Definition 7.23 The $\mathcal{N}$ system for FOL consists of:

$$
\begin{array}{lll}
\text{Axioms} :: & \mathsf{A0}: & \Gamma \vdash_{\mathcal{N}} B, \text{ for all } B \in \Gamma; \\
& \mathsf{A1}: & \Gamma \vdash_{\mathcal{N}} A \to (B \to A); \\
& \mathsf{A2}: & \Gamma \vdash_{\mathcal{N}} (A \to (B \to C)) \ \to \ ((A \to B) \to (A \to C)); \\
& \mathsf{A3}: & \Gamma \vdash_{\mathcal{N}} (\neg B \to \neg A) \ \to \ (A \to B); \\
& \mathsf{A4}: & \Gamma \vdash_{\mathcal{N}} A^x_t \to \exists x A \quad \text{if } t \text{ is substitutable for } x \text{ in } A. \\
\text{Rules} :: & \mathsf{MP}: & \dfrac{\Gamma \vdash_{\mathcal{N}} A \ \ ; \ \ \Gamma \vdash_{\mathcal{N}} A \to B}{\vdash_{\mathcal{N}} B} \ ; \\
& \exists\mathsf{I}: & \dfrac{\Gamma \vdash_{\mathcal{N}} A \to B}{\Gamma \vdash_{\mathcal{N}} \exists x A \to B} \ \ \text{if } x \text{ has no free occurrence in } B
\end{array}
$$

Since we take with us all the axiom schemata and rules from definition 4.11, every theorem in that earlier system has a counterpart in this system. Hence for instance lemmas 4.7 and 4.15 can be taken over directly. In other words, the uppercase letters in the above rules and axioms stand now for arbitrary FOL-formulae. For instance, $\exists x A \to (\exists x \neg \exists y D \to \exists x A)$ is an instance of A1. Thus any operations (derivations) we performed using propositional variables in PL, can be now performed in the same way, provided that the FOL-formulae involved are syntactically identical whenever required (like in the above axiom instance).

Admissible rules are a different matter, but in most cases these also carry over to the extended system. Thus the next lemma corresponds exactly to lemmas 4.9 and 4.12, and is proved in exactly the same way.

Lemma 7.24 The following rules are admissible in $\mathcal{N}$:

1. $$\frac{\Gamma \vdash_{\mathcal{N}} A \to B \ \ ; \ \ \Gamma \vdash_{\mathcal{N}} B \to C}{\Gamma \vdash_{\mathcal{N}} A \to C}$$
2. $$\frac{\Gamma \vdash_{\mathcal{N}} B}{\Gamma \vdash_{\mathcal{N}} A \to B}$$

The next lemma illustrates the use of the new elements, i.e., A4 and the

"$\exists$ introduction" rule $\exists$I.

Lemma 7.25 Formula 1. is provable (from any Γ) and rules 2.-5. are admissible in $\mathcal{N}$.

(1) $\Gamma \vdash_{\mathcal{N}} \forall x A \to A$

(2) $\dfrac{\Gamma \vdash_{\mathcal{N}} A \to B}{\Gamma \vdash_{\mathcal{N}} \exists x A \to \exists x B}$

(3) $\forall\mathrm{I} : \dfrac{\Gamma \vdash_{\mathcal{N}} B \to A}{\Gamma \vdash_{\mathcal{N}} B \to \forall x A}$ if x has no free occurrence in B

(4) $\forall\mathrm{G} : \dfrac{\Gamma \vdash_{\mathcal{N}} A}{\Gamma \vdash_{\mathcal{N}} \forall x A}$

(5) SB: $\dfrac{\Gamma \vdash_{\mathcal{N}} A}{\Gamma \vdash_{\mathcal{N}} A^x_t}$ if t is substitutable for x in A

Proof. If something is provable using only A0–A3 and MP we write just PL to the right – these parts have been proven earlier or are left as exercises. (In view of the completeness theorem of PL, it is sufficient to convince oneself that it corresponds to a tautology.)

1.
$1 : \Gamma \vdash_{\mathcal{N}} \neg A \to \exists x \neg A$ — $A4$
$2 : \Gamma \vdash_{\mathcal{N}} (\neg A \to \exists x \neg A) \to (\neg\exists x \neg A \to A)$ — PL
$3 : \Gamma \vdash_{\mathcal{N}} \neg\exists x \neg A \to A$ — $MP(1,2)$

2.
$1 : \Gamma \vdash_{\mathcal{N}} A \to B$ — $assumption$
$2 : \Gamma \vdash_{\mathcal{N}} B \to \exists x B$ — $A4$
$3 : \Gamma \vdash_{\mathcal{N}} A \to \exists x B$ — $L.7.24.1(1,2)$
$4 : \Gamma \vdash_{\mathcal{N}} \exists x A \to \exists x B$ — $\exists$I (3)

3.
$1 : \Gamma \vdash_{\mathcal{N}} B \to A$ — $assumption$
$2 : \Gamma \vdash_{\mathcal{N}} \neg A \to \neg B$ — $MP(1, L.4.15)$
$3 : \Gamma \vdash_{\mathcal{N}} (\exists x \neg A) \to \neg B$ — $\exists$I $(2) + x$ not free in B
$4 : \Gamma \vdash_{\mathcal{N}} ((\exists x \neg A) \to \neg B) \to (B \to \neg\exists x \neg A)$ — PL
$5 : \Gamma \vdash_{\mathcal{N}} B \to \neg\exists x \neg A$ — $MP(3,4)$

4.
$1 : \Gamma \vdash_{\mathcal{N}} A$ — $assumption$
$2 : \Gamma \vdash_{\mathcal{N}} (\exists x \neg A) \to A$ — $L.7.24.2(1)$
$3 : \Gamma \vdash_{\mathcal{N}} (\exists x \neg A) \to \neg\exists x \neg A$ — $\forall$I (2)
$4 : \Gamma \vdash_{\mathcal{N}} ((\exists x \neg A) \to \neg\exists x \neg A) \to \neg\exists x \neg A$ — PL
$5 : \Gamma \vdash_{\mathcal{N}} \neg\exists x \neg A$ — $MP(3,4)$

5.
$1 : \Gamma \vdash_{\mathcal{N}} A$ — $assumption$
$2 : \Gamma \vdash_{\mathcal{N}} \neg\exists x \neg A$ — $L.7.25.4$
$3 : \Gamma \vdash_{\mathcal{N}} \neg A^x_t \to \exists x \neg A$ — $A4$
$4 : \Gamma \vdash_{\mathcal{N}} (\neg\exists x \neg A) \to A^x_t$ — PL(3)
$5 : \Gamma \vdash_{\mathcal{N}} A^x_t$ — $MP(2,4)$

QED (7.25)

3.1: Deduction Theorem in FOL

Notice the difference between

i) $\Gamma \vdash_{\mathcal{N}} A \to B$ and

ii) the admissible rule $\dfrac{\Gamma \vdash_{\mathcal{N}} A}{\Gamma \vdash_{\mathcal{N}} B}$.

Point 1 of the above lemma, enables us to conclude, by a single aplication of MP, that the rule inverse to the one in point 4, namely, $\dfrac{\Gamma \vdash_{\mathcal{N}} \forall x A}{\Gamma \vdash_{\mathcal{N}} A}$ is admissible. In fact, quite generally, i) implies ii), for having i) and the assumption of ii), single application of MP yields the conclusion of ii).

Now, in the $\mathcal{N}$ system for PL, the opposite implication holds, too. For assume ii), i.e., that the rule $\dfrac{\Gamma \vdash_{\mathcal{N}} \forall x A}{\Gamma \vdash_{\mathcal{N}} A}$ is admissible. In particular, we have $\Gamma, A \vdash_{\mathcal{N}} A$ by axiom A0, from which $\Gamma, A \vdash_{\mathcal{N}} B$ follows by ii), and then $\Gamma \vdash_{\mathcal{N}} A \to B$ by the Deduction Theorem.

In FOL, however, the implication from ii) to i) is not necessarily true, and this is related to the limited validity of Deduction Theorem. For instance, point 4 does not allow us to conclude that also $\Gamma \vdash_{\mathcal{N}} A \to \forall x A$. In fact, this is not the case, but we have to postpone a precise argument showing that until we have discussed semantics. At this point, let us only observe that if this formula were provable, then we would also have $\Gamma \vdash_{\mathcal{N}} \exists x A \to \forall x A$ by a single application of $\exists$I. But this looks unsound: "if there exists an x such that A, then for all x A". (Sure, in case the assumption of the rule 4 from lemma 7.25 is satisfied we can obtain: $\Gamma \vdash_{\mathcal{N}} A$, $\Gamma \vdash_{\mathcal{N}} \forall x A$, and so, by lemma 7.24.2 $\Gamma \vdash_{\mathcal{N}} A \to \forall x A$. But this is only a very special case dependent on the assumption $\Gamma \vdash_{\mathcal{N}} A$.)

Example 7.26

Let us consider the example with horse-heads and animal-heads from the background story at the begining of this chapter. We design an alphabet with two unary predicates $\{H, A\}$ for 'being a horse' and 'being an animal', respectively, and a binary relation $Hd(x, y)$ for 'x being a head of y'. The argument was then captured in the following form:

$$\frac{\vdash_{\mathcal{N}} \forall y(H(y) \to A(y))}{\vdash_{\mathcal{N}} \forall x(\ \exists y(H(y) \land Hd(x, y)) \to \exists y(A(y) \land Hd(x, y))\ )} \tag{7.27}$$

We show that it is provable, that is, the above (a bit strange and particular) rule is admissible in $\mathcal{N}$:

$1: \Gamma \vdash_{\mathcal{N}} \forall y(H(y) \to A(y))$ *assumption*
$2: \Gamma \vdash_{\mathcal{N}} H(y) \to A(y)$ $L.7.25.1$ *remark above* (1)
$3: \Gamma \vdash_{\mathcal{N}} H(y) \wedge Hd(x,y) \to A(y) \wedge Hd(x,y)$ PL (2)
$4: \Gamma \vdash_{\mathcal{N}} \exists y(H(y) \wedge Hd(x,y)) \to \exists y(A(y) \wedge Hd(x,y))$ $L.7.25.2$ (3)
$5: \Gamma \vdash_{\mathcal{N}} \forall x(\ \exists y(H(y) \wedge Hd(x,y)) \to \exists y(A(y) \wedge Hd(x,y))\ )$ $\forall$I : $L.7.25.4$ (4)

This shows the claim (admissibility of (7.27)) for arbitrary Γ. In particular, if we take $\Gamma = \{\forall y(H(y) \to A(y))\}$, the first line (assumption) becomes an instance of the axiom $A0$ and the last line becomes an uncoditional statement:

$$\forall y(H(y) \to A(y)) \ \vdash_{\mathcal{N}} \ \forall x(\ \exists y(H(y) \wedge Hd(x,y)) \to \exists y(A(y) \wedge Hd(x,y))\ ). \tag{7.28}$$

As we observed before this example, admissibility of a rule ii), like (7.27), does not necessarily mean that the corresponding implication i) is provable, i.e., we are *not* entitled to conclude from (7.28) that also the following holds:

$$\vdash_{\mathcal{N}} \forall y(H(y) \to A(y)) \quad \to \quad \forall x(\ \exists y(H(y) \wedge Hd(x,y)) \to \exists y(A(y) \wedge Hd(x,y))\ ). \tag{7.29}$$

This could be obtained from (7.28) if we had Deduction Theorem for FOL– we now turn to this issue. □

As a matter of fact, the unrestricted Deduction Theorem from PL is not an admissible proof rule of FOL. It will be easier to see why when we turn to the semantics. For now we just prove the weaker version that does hold. Note the restriction on A.

Theorem 7.30 **[Deduction Theorem for** FOL**]** If $\Gamma, A \vdash_{\mathcal{N}} B$, and A is *closed* then $\Gamma \vdash_{\mathcal{N}} A \to B$.

Proof. By induction on the length of the proof $\Gamma, A \vdash_{\mathcal{N}} B$. The cases of axioms and MP are treated exactly as in the proof of the theorem for PL, 4.13 (using lemmas 4.7 and 7.24.2). We have to verify the induction step for the last step of the proof using $\exists$I, i.e.:

$$\frac{\Gamma, A \vdash_{\mathcal{N}} C \to D}{\Gamma, A \vdash_{\mathcal{N}} \exists x C \to D} \ x \text{ not free in } D$$

By IH, we have the first line of the following proof:

$1: \Gamma \vdash_{\mathcal{N}} A \to (C \to D)$
$2: \Gamma \vdash_{\mathcal{N}} C \to (A \to D)$ PL ($C.4.17$)
$3: \Gamma \vdash_{\mathcal{N}} \exists x C \to (A \to D)$ $\exists$I (2) + A closed and x not free in D
$4: \Gamma \vdash_{\mathcal{N}} A \to (\exists x C \to D)$ PL ($C.4.17$) **QED** (7.30)

Revisiting (7.28) from example 7.26, we see that the assumption of Deduction Theorem is satisfied (i.e., $\forall y(H(y) \to A(y))$ is closed). Thus, in this particular case, we actually may conclude that also (7.29) holds. However, due to the restriction in Deduction Theorem, such a transition will not be possible in general.

Just like in the case of PL, MP is a kind of dual to this theorem and we have the corollary corresponding to 4.16, with the same proof.

Corollary 7.31 If A is closed then: $\Gamma, A \vdash_{\mathcal{N}} B$ iff $\Gamma \vdash_{\mathcal{N}} A \to B$.

Notice that the assumption that A is closed is needed because of the deduction theorem, i.e., only for the implication $\Rightarrow$. The opposite $\Leftarrow$ does not require A to be closed and is valid for any A.

4: Gentzen's system for FOL

Recall, that $\mathcal{G}$ for PL worked with sequents $\Gamma \vdash_{\mathcal{G}} \Delta$, where both Γ, Δ are (finite) sets of formulae. The axioms 1. and rules 2. through 5. and 2'. through 5'. give a sound and complete Gentzen system for PL – with all the connectives. Since the set $\{\neg, \to\}$ is adequate we restricted earlier our attention to these connectives and the rules 4.,4'.,5. and 5'. The current rules may be easier to use in the presence of other connectives. It is easy to see that treating, for instance, $\vee$ and $\wedge$ as abbreviations, the corresponding rules $(2, 2', 3, 3')$ are derivable from the other rules (cf. 8.2 in week 4.). The Gentzen system for FOL is obtained by adding the quantifier rules 6.

through 7'.

1. $Ax \quad \Gamma \vdash_{\mathcal{G}} \Delta \quad where \;\; \Gamma \cap \Delta \neq \varnothing$

$$2.\ \vdash\vee\ \frac{\Gamma \vdash_{\mathcal{G}} A, B, \Delta}{\Gamma \vdash_{\mathcal{G}} A \vee B, \Delta} \qquad 2'.\ \vee\vdash\ \frac{\Gamma, A \vdash_{\mathcal{G}} \Delta \quad ; \quad \Gamma, B \vdash_{\mathcal{G}} \Delta}{\Gamma, A \vee B \vdash_{\mathcal{G}} \Delta}$$

$$3.\ \vdash\wedge\ \frac{\Gamma \vdash_{\mathcal{G}} A, \Delta \quad ; \quad \Gamma \vdash_{\mathcal{G}} B, \Delta}{\Gamma \vdash_{\mathcal{G}} A \wedge B, \Delta} \qquad 3'.\ \wedge\vdash\ \frac{\Gamma, A, B \vdash_{\mathcal{G}} \Delta}{\Gamma, A \wedge B \vdash_{\mathcal{G}} \Delta}$$

$$4.\ \vdash\neg\ \frac{\Gamma, B \vdash_{\mathcal{G}} \Delta}{\Gamma \vdash_{\mathcal{G}} \neg B, \Delta} \qquad 4'.\ \neg\vdash\ \frac{\Gamma \vdash_{\mathcal{G}} B, \Delta}{\Gamma, \neg B \vdash_{\mathcal{G}} \Delta}$$

$$5.\ \vdash\to\ \frac{\Gamma, A \vdash_{\mathcal{G}} B, \Delta}{\Gamma \vdash_{\mathcal{G}} A \to B, \Delta} \qquad 5'.\ \to\vdash\ \frac{\Gamma \vdash_{\mathcal{G}} \Delta, A \quad ; \quad \Gamma, B \vdash_{\mathcal{G}} \Delta}{\Gamma, A \to B \vdash_{\mathcal{G}} \Delta}$$

$$6.\ \vdash\exists\ \frac{\Gamma \vdash_{\mathcal{G}} \Delta, \exists x A, A^x_t}{\Gamma \vdash_{\mathcal{G}} \Delta, \exists x A}\ A^x_t\ legal \qquad 6'.\ \forall\vdash\ \frac{A^x_t, \forall x A, \Gamma \vdash_{\mathcal{G}} \Delta}{\forall x A, \Gamma \vdash_{\mathcal{G}} \Delta}\ A^x_t\ legal$$

$$7.\ \vdash\forall\ \frac{\Gamma \vdash_{\mathcal{G}} A^x_{x'}, \Delta}{\Gamma, \vdash_{\mathcal{G}} \forall x A, \Delta}\ x'\ fresh \qquad 7'.\ \exists\vdash\ \frac{\Gamma, A^x_{x'}, \vdash_{\mathcal{G}} \Delta}{\Gamma, \exists x A \vdash_{\mathcal{G}} \Delta}\ x'\ fresh$$

The requirement on a variable 'x' to be fresh' means that it must be a new variable not occurring in the sequent. (One may require only that it does not occur freely in the sequent, but we will usually mean that it does not occur at all.) This, in particular, means that its substitution for x is legal.

Notice the peculiar repetition of $\exists x A$, resp. $\forall x A$ in rules 6., resp. 6'. Semantically, they make the rules trivially invertible (exercise 6.9). In terms of building a bottom-up proof, they enable us to choose all the time new witnesses until, eventually and hopefully, we arrive at an instance of the axiom. In principle, however, and typically, there is infinitely many terms t which might be substituted and so these rules do not solve the problem of decidability. As an example of their application, the axiom A4 of $\mathcal{N}$ is proved as follows:

3. $A^x_t \vdash_{\mathcal{G}} \exists x A, A^x_t$ 1. $\quad A^x_t\ legal\ by\ assumption$
2. $A^x_t \vdash_{\mathcal{G}} \exists x A$ 6.
1. $\vdash_{\mathcal{G}} A^x_t \to \exists x A$ 5. $\quad A^x_t\ legal$

The point with these, apparently redundant, repetitions is that building a proof in a somehow mechanical fashion, we may "try" various terms in order. If we simply removed $\exists x A$ and replaced it by a "wrong" A^x_t, we would have to stop. For instance, if we have another term s which is substitutable for x in A – and are "unlucky" – we might run the proof as follows:

4. $A^x_t \vdash_{\mathcal{G}} \exists xA, A^x_s, A^x_t$	1.	A^x_t *legal*
3. $A^x_t \vdash_{\mathcal{G}} \exists xA, A^x_s$	6.	A^x_s *legal*
2. $A^x_t \vdash_{\mathcal{G}} \exists xA$	6.	
1. $\vdash_{\mathcal{G}} A^x_t \to \exists xA$	5.	A^x_t *legal*

Keeping $\exists xA$ enables us (or a machine!) to continue building the proof bottom-up past the "unlucky substitution" A^x_s in line 3. This indicates the difficulties with treatment of the quantifiers. In general, a wrong choice of a witness in 6. or $6'$. may terminate the proof with an inappropriate conclusion. On the other hand, even if the proof does not terminate, there may be no mechanical way of ensuring that all possible witnesses will be tried. These are the reasons for *undecidability* of $\mathcal{G}$ for FOL. As we will later see, theoremhood and validity of FOL formulae is generally undecidable, so this is no fault of the system $\mathcal{G}$.

Observe also that the rule 5. is an *unrestricted* Deduction Theorem (unlike 7.30). We will comment on this issue in the following chapter.

Example 7.32

We show provability of the formula from (7.29) in $\mathcal{G}$. As in PL, we use $\mathcal{G}$ for constructing the proof bottom-up.

All $x, y, z, w \in \mathcal{V}$. In applications admitting repetition of a formula ($\vdash\exists$ in line 5. and $\forall\vdash$ in line 3.) we have dropped these repetitions to make the proof more readable – of course, we have to choose the substituted terms in a way making the proof to go through, in particular, so that the corresponding substitutions at lines 6. and 8. are both legal. In these latter applications, the F, resp. G in the marking 'F/G^j_i *legal*', refers only to the formula where the substitution actually takes place – in line 6. F refers to $A(y) \wedge H(z, y)$, and in line 8. G to $H(y) \to A(y)$.

This is a standard rule to observe when constructing (bottom-up) proofs in $\mathcal{G}$ – one always tries first to perform the subsitutions requiring a fresh variable (apply rules $\vdash\forall$ and/or $\exists\vdash$), and only later the ones requiring merely the substitution to be legal (i.e., the rules $\vdash\exists$ and/or $\forall\vdash$). These latter applications allow, namely, to choose substituted terms freely and to adjust their choice to the other terms occurring already in the sequent.

$$\begin{array}{lccc}
 & \quad axiom \quad ; \quad axiom & & \\
10.\ \to\vdash & \underline{H, Hd \vdash_{\mathcal{G}} A, H \ ; \ H, Hd, A \vdash_{\mathcal{G}} A} & & axiom \\
9.\ \vdash\wedge & \multicolumn{3}{c}{\underline{H, Hd, H \to A \vdash_{\mathcal{G}} A \quad ; \quad H, Hd, H \to A \vdash_{\mathcal{G}} Hd}} \\
8. & \multicolumn{3}{c}{H, Hd, H \to A \vdash_{\mathcal{G}} A \wedge Hd}
\end{array}$$

We now have propositional case so we don′t write variables in the rest of the proof

$$\begin{array}{lrl}
8. & H(w), Hd(z,w), H(w) \to A(w) \vdash_{\mathcal{G}} A(w) \wedge Hd(z,w) & \forall\vdash : G^y_w\ legal \\
7. & H(w), Hd(z,w), \forall y(H(y) \to A(y)) \vdash_{\mathcal{G}} A(w) \wedge Hd(z,w) & \wedge\vdash \\
6. & H(w) \wedge Hd(z,w), \forall y(H(y) \to A(y)) \vdash_{\mathcal{G}} A(w) \wedge Hd(z,w) & \vdash\exists : F^y_w\ legal \\
5. & H(w) \wedge Hd(z,w), \forall y(H(y) \to A(y)) \vdash_{\mathcal{G}} \exists y(A(y) \wedge Hd(z,y)) & \exists\vdash : w\ fresh \\
4. & \exists y(H(y) \wedge Hd(z,y)), \forall y(H(y) \to A(y)) \vdash_{\mathcal{G}} \exists y(A(y) \wedge Hd(z,y)) & \vdash\to \\
3. & \forall y(H(y) \to A(y)) \vdash_{\mathcal{G}} \exists y(H(y) \wedge Hd(z,y)) \to \exists y(A(y) \wedge Hd(z,y)) & \vdash\forall : z\ fresh \\
2. & \forall y(H(y) \to A(y)) \vdash_{\mathcal{G}} \forall x(\ \exists y(H(y) \wedge Hd(x,y)) \to \exists y(A(y) \wedge Hd(x,y))\ ) & \vdash\to \\
1. & \vdash_{\mathcal{G}} \forall y(H(y) \to A(y)) \ \to\ \forall x(\ \exists y(H(y) \wedge Hd(x,y)) \to \exists y(A(y) \wedge Hd(x,y))\ ) &
\end{array}$$

□

Exercises 7.

EXERCISE 7.1 Give an inductive definition of the function $\mathcal{V} : \mathsf{WFF}^{\Sigma}_{\mathsf{FOL}} \to \wp(\mathcal{V})$ returning the set of variables occuring freely in a formula.

EXERCISE 7.2 Prove the following:

(1) $\vdash_{\mathcal{N}} \exists y \exists x A \to \exists x \exists y A$

Hint: Complete the following proof by filling out appropriate things for '?':

1. $\vdash_{\mathcal{N}} A \to \exists y A$	$A4$
2. ?	$A4$
3. $\vdash_{\mathcal{N}} A \to \exists x \exists y A$	$L.7.24.1(1,2)$
4. $\vdash_{\mathcal{N}} \exists x A \to \exists x \exists y A$	? (3) (x not free in $\exists x \exists y A$)
5. $\vdash_{\mathcal{N}} \exists y \exists x A \to \exists x \exists y A$	? (4) (?)

(2) $\vdash_{\mathcal{N}} \exists y A^x_y \to \exists x A$ – if y is substitutable for x in A, and y is not free in A

(Hint: Two steps only! First an instance of A4, and then ∃I.)

(3) $\vdash_{\mathcal{N}} \exists x \forall y A \ \to\ \forall y \exists x A$

(Hint: Lemma 7.25, starting with point 1., then 2. and finally 3.)

(4) $\vdash_{\mathcal{N}} \forall x A \rightarrow \exists x A$
(Hint: Lemma 7.25.1, A4, and then lemma 7.24.1.)

EXERCISE 7.3 Is the following proof correct? If not, what is wrong?

$1 : \vdash_{\mathcal{N}} \forall x(P(x) \vee R(x)) \rightarrow (P(x) \vee R(x))$	$L.7.25.1$
$2 : \vdash_{\mathcal{N}} \forall x(P(x) \vee R(x)) \rightarrow (\forall x P(x) \vee R(x))$	$\forall$I
$3 : \vdash_{\mathcal{N}} \forall x(P(x) \vee R(x)) \rightarrow (\forall x P(x) \vee \forall x R(x))$	$\forall$I

EXERCISE 7.4 Re-wrok example 7.26 for the other argument from the background story at the begining of this chapter, i.e.:

(1) Design an alphabet for expressing the argument:

	`Every man is mortal;`
and	`Socrates is a man;`
hence	`Socrates is mortal.`

(2) Express it as a rule (analogous to (7.27)) and show that it is admissible in $\mathcal{N}$.
(3) Write it now as a single formula (an implication analogous to (7.29)) – you have to decide which connective(s) to choose to join the two premisses of the rule into antecedent of the implication! Use the previous point (and restricted version of Deduction Theorem 7.30) to show that this formula is provable in $\mathcal{N}$.
(4) Prove now your formula from the previous point using $\mathcal{G}$.

EXERCISE 7.5 The following statement:

(1) $\vdash_{\mathcal{N}} (A \rightarrow \exists x B) \rightarrow \exists x(A \rightarrow B)$

can be proven as follows:

1. $\vdash_{\mathcal{N}} B \rightarrow (A \rightarrow B)$	$A1$
2. $\vdash_{\mathcal{N}} \exists x B \rightarrow \exists x(A \rightarrow B)$	$L.7.25.2$
3. $\vdash_{\mathcal{N}} (A \rightarrow B) \rightarrow \exists x(A \rightarrow B)$	$A4$
4. $\vdash_{\mathcal{N}} \neg A \rightarrow \exists x(A \rightarrow B)$	PL(3)
5. $\vdash_{\mathcal{N}} (A \rightarrow \exists x B) \rightarrow \exists x(A \rightarrow B)$	PL(2, 4)

Verify that the lines 4. and 5. really are valid transitions according to PL. (Hint: They correspond to provability (or validity!), for instance in $\mathcal{G}$, of: $\vdash((A \rightarrow B) \rightarrow Z) \rightarrow (\neg A \rightarrow Z)$ and $B \rightarrow Z, \neg A \rightarrow Z \vdash (A \rightarrow B) \rightarrow Z$.)

EXERCISE 7.6 Assuming that x has no free occurrences in A, complete the proofs of the following:

(1) $\vdash_{\mathcal{N}} \exists x(A \rightarrow B) \rightarrow (A \rightarrow \exists x B)$

1. $\vdash_{\mathcal{N}} B \rightarrow \exists x B$	$A4$
2. $\vdash_{\mathcal{N}} A \rightarrow (B \rightarrow \exists x B)$	(?)
3. $\vdash_{\mathcal{N}} (A \rightarrow (B \rightarrow \exists x B)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow \exists x B))$	$A2$
...	

(2) $\vdash_{\mathcal{N}} \forall x(B \rightarrow A) \rightarrow (\exists x B \rightarrow A)$

1. $\vdash_{\mathcal{N}} \forall x(B \to A) \to (B \to A)$ (?)
2. $\vdash_{\mathcal{N}} B \to (\forall x(B \to A) \to A)$ PL(?)

   ...

(3) $\vdash_{\mathcal{N}} \exists x(B \to A) \to (\forall xB \to A)$

1. $\vdash_{\mathcal{N}} (\forall xB \to B) \to ((B \to A) \to (\forall xB \to A))$ (?)
2. $\vdash_{\mathcal{N}} \forall xB \to B$ (?)

   ...

<u>EXERCISE 7.7</u> Prove now all the formulae from exercises 7.2 and 7.6 using Gentzen system.

<u>EXERCISE 7.8</u> We will use Gentzen system

(1) Show provability of the formula:

$\vdash_{\mathcal{G}} \forall x(A \to B) \to (\forall xA \to \forall xB)$.

(2) Now try to construct a proof for the opposite implication, i.e.,

$\vdash_{\mathcal{G}} (\forall xA \to \forall xB) \to \forall x(A \to B)$.

Can you tell why this proof will not "succeed"?

Chapter 8
Semantics

- Semantic Definitions
- Semantic Properties of FOL Formulae
- Deduction theorem in $\mathcal{N}$ and $\mathcal{G}$ for FOL

1: Semantics of FOL

Definition 8.1 [FOL **Structure**] A FOL *structure* M over an alphabet Σ consists of

(1) a non-empty set $\underline{M}$ – the *interpretation domain*
(2) for each constant $a \in \mathcal{I}$, an individual $[\![a]\!]^M \in \underline{M}$
(3) for each function symbol $f \in \mathcal{F}$ with arity n, a function $[\![f]\!]^M : \underline{M}^n \to \underline{M}$
(4) for each relation symbol $P \in \mathcal{R}$ with arity n, a subset $[\![P]\!]^M \subseteq \underline{M}^n$

Thus, a structure is simply a set where all constant, function and relation symbols have been interpreted arbitrarily. The only restriction concerns the arities of the function and relation symbols which have to be respected by their interpretation.

Notice that according to definition 8.1, variables do not receive any fixed interpretation. Thus, it does not provide sufficient means for assigning meaning to *all syntactic expressions* from the language. The meaning of variables, and expressions involving variables, will depend on the choice of the assignment.

Definition 8.2 [**Interpretation of terms**] Any function $v : \mathcal{V} \to \underline{M}$ is called a *variable assignment* or just an assignment. Given a structure M and an assignment v, the interpretation of terms is defined inductively:

(1) For $x \in \mathcal{V} : [\![x]\!]^M_v = v(x)$
(2) For $a \in \mathcal{I} : [\![a]\!]^M_v = [\![a]\!]^M$
(3) For n-ary $f \in \mathcal{F}$ and $t_1, \ldots, t_n \in \mathcal{T}_\Sigma$: $[\![f(t_1, \ldots, t_n)]\!]^M_v = [\![f]\!]^M([\![t_1]\!]^M_v, \ldots, [\![t_n]\!]^M_v)$.

According to point 2., interpretation of constants does not depend on variable assignment. Similarly, the interpretation of a function symbol in point 3. does not either. This means that interpretation of an arbitrary ground term $t \in \mathcal{GT}$ is fixed in a given structure M, i.e., for any assignment $v : [\![t]\!]^M_v = [\![t]\!]^M$.

Example 8.3

Let Σ contain the constant symbol $\odot$, one unary function symbol s and a binary function symbol $\oplus$. Here are some examples of Σ-structures:

(1) A is the natural numbers $\mathbb{N}$ with $[\![\odot]\!]^A = 0$, $[\![s]\!]^A = +1$ and $[\![\oplus]\!]^A = +$. Here, we will understand $ss\odot\oplus sss\odot$ as 5, i.e., $[\![ss\odot \oplus sss\odot]\!]^A = 2+3 = 5$.
For an assignment v with $v(x) = 2$ and $v(y) = 3$, we get $[\![x \oplus y]\!]^A_v = [\![x]\!]^A_v\ [\![\oplus]\!]^A\ [\![y]\!]^A_v = 2 + 3 = 5$.
For an assignment v with $v(x) = 4$ and $v(y) = 7$, we get $[\![x \oplus y]\!]^A_v = [\![x]\!]^A_v\ [\![\oplus]\!]^A\ [\![y]\!]^A_v = 4 + 7 = 11$.

(2) B is the natural numbers $\mathbb{N}$ with $[\![\odot]\!]^B = 0$, $[\![s]\!]^B = +1$ and $[\![\oplus]\!]^B = *$. Here we will have : $[\![ss\odot \oplus sss\odot]\!]^B = 2 * 3 = 6$.

(3) C is the integers $\mathbb{Z}$ with $[\![\odot]\!]^C = 1$, $[\![s]\!]^C = +2$ and $[\![\oplus]\!]^C = -$. Here we will have : $[\![ss\odot \oplus sss\odot]\!]^C = 5 - 7 = -2$.
What will be the values of $x \oplus y$ under the assignments from 1.?

(4) Given a non-empty set (say, e.g., $S = \{a, b, c\}$), we let the domain of D be S^* (the finite strings over S), with $[\![\odot]\!]^D = \epsilon$ (the empty string), $[\![s]\!]^D(\epsilon) = \epsilon$ and $[\![s]\!]^D(wx) = w$ (where x is the last element in the string wx), and $[\![\oplus]\!]^D(p, t) = pt$ (i.e., concatenation of strings).

□

As we can see, the requirements on something being a Σ-structure are very weak – a non-empty set with *arbitrary* interpretation of constant, function and relation symbols respecting merely arity. Consequently, there is a huge number of structures for any alphabet – in fact, so huge that it is not even a set but a class. We will not, however, be concerned with this distinction. If necessary, we will denote the collection of *all* Σ-structures by $\mathsf{Str}(\Sigma)$.

A Σ-structure M, together with an assignment, induces the interpretation of $\mathsf{WFF}^{\Sigma}_{\mathsf{FOL}}$. As for PL, such an interpretation is a function $[\![\_]\!]^M_v : \mathsf{WFF}^{\Sigma}_{\mathsf{FOL}} \to \{\mathbf{1}, \mathbf{0}\}$.

Definition 8.4 [**Interpretation of formulae**] M determines a boolean value for every formula relative to every variable assignment v, according to the following rules:

(1) If $P \in \mathcal{R}$ is n-ary and $t_1, \ldots, t_n$ are terms, then
$$[\![P(t_1, \ldots, t_n)]\!]^M_v = \mathbf{1} \quad \Leftrightarrow \quad \langle [\![t_1]\!]^M_v, \ldots, [\![t_n]\!]^M_v \rangle \in [\![P]\!]^M$$

(2) Propositional connectives are combined as in PL. For $A, B \in \mathsf{WFF}^{\Sigma}_{\mathsf{FOL}}$:

$$\begin{aligned} [\![\neg A]\!]^M_v = \mathbf{1} &\Leftrightarrow [\![A]\!]^M_v = \mathbf{0} \\ [\![A \to B]\!]^M_v = \mathbf{1} &\Leftrightarrow [\![A]\!]^M_v \text{ implies } [\![B]\!]^M_v \\ &\Leftrightarrow [\![A]\!]^M_v = \mathbf{0} \text{ or } [\![B]\!]^M_v = \mathbf{1} \end{aligned}$$

(3) Quantified formulae:

$$[\![\exists x A]\!]^M_v = \mathbf{1} \Leftrightarrow \text{there is an } \underline{a} \in \underline{M} : [\![A]\!]^M_{v[x \mapsto \underline{a}]} = \mathbf{1}$$

Recall (remark 1.5) that the notation $v[x \mapsto a]$ denotes the function v modified at one point, namely so that now $v(x) = a$. Thus the definition says that the value assigned to the bound variable x by valuation v is inessential when determining the boolean value $[\![\exists x A]\!]_v$ – no matter what $v(x)$ is, it will be "modified" $v[x \mapsto \underline{a}]$ if an appropriate $\underline{a}$ can be found. We will observe consequences of this fact for the rest of the current subsection.

The following fact justifies the reading of $\forall x$ as "for all x".

Fact 8.5. For any structure M and assignment v, $[\![\forall x A]\!]^M_v = \mathbf{1} \Leftrightarrow$ for all $\underline{a} \in \underline{M} : [\![A]\!]^M_{v[x \mapsto \underline{a}]} = \mathbf{1}$.

Proof. We only expand the definition 7.7 of the abbreviation $\forall x$ and apply 8.4.

$$\begin{aligned} [\![\forall x A]\!]^M_v = \mathbf{1} &\overset{7.7}{\Longleftrightarrow} [\![\neg \exists x \neg A]\!]^M_v = \mathbf{1} \quad \overset{8.4}{\Longleftrightarrow} \quad [\![\exists x \neg A]\!]^M_v = \mathbf{0} \\ &\overset{8.4}{\Longleftrightarrow} [\![\neg A]\!]^M_{v[x \mapsto \underline{a}]} = \mathbf{1} \text{ for no } \underline{a} \in \underline{M} \quad \overset{8.4}{\Longleftrightarrow} \quad [\![A]\!]^M_{v[x \mapsto \underline{a}]} = \mathbf{0} \text{ for no } \underline{a} \in \underline{M} \\ &\overset{8.4}{\Longleftrightarrow} [\![A]\!]^M_{v[x \mapsto \underline{a}]} = \mathbf{1} \text{ for all } \underline{a} \in \underline{M} \end{aligned}$$

QED (8.5)

Again, notice that to evaluate $[\![\forall x A]\!]_v$, it does not matter what $v(x)$ is – we will have to check *all* possible modifications $v[x \mapsto \underline{a}]$ anyway.

Point 3 of definition 8.4 captures the crucial difference between free and bound variables – the truth of a formula depends on the *names* of the free variables but not of the bound ones. More precisely, a closed formula (sentence) is either true or false in a given structure – its interpretation according to the above definition will *not* depend on the assignment v. An open formula is neither true nor false – since we do not know what objects the free variables refer to. To determine the truth of an open formula, the above definition requires us to provide an assignment to its free variables.

Example 8.6

Consider an alphabet with one binary relation R and a structure M with three elements $\underline{M} = \{\underline{a}, \underline{b}, \underline{c}\}$ and $[\![R]\!]^M = \{\langle \underline{a}, \underline{b} \rangle\}$. Let A be the formula $\exists x R(x, y)$ – we check whether $[\![A]\!]^M_v = \mathbf{1}$, resp. $[\![A]\!]^M_w = \mathbf{1}$ for the following assignments v, w :

$$
\begin{aligned}
v &= \{x \mapsto \underline{a}, y \mapsto \underline{a}\} \\
\llbracket \exists x R(x,y) \rrbracket^M_v = \mathbf{1} &\Leftrightarrow \llbracket R(x,y) \rrbracket^M_{v[x \mapsto \underline{x}]} = \mathbf{1} \text{ for some } \underline{x} \in \underline{M} \\
&\Leftrightarrow \langle v[x \mapsto \underline{x}](x), v[x \mapsto \underline{x}](y) \rangle \in \llbracket R \rrbracket^M \text{ for some } \underline{x} \in \underline{M} \\
&\Leftrightarrow \langle v[x \mapsto \underline{x}](x), \underline{a} \rangle \in \llbracket R \rrbracket^M \text{ for some } \underline{x} \in \underline{M} \quad (\text{since } v[x \mapsto \underline{x}](y) = v(y) = \underline{a}) \\
&\Leftrightarrow \text{at least one of the following holds} \qquad (\text{we have } v[x \mapsto \underline{x}](x) = \underline{x})
\end{aligned}
$$

$$
\begin{array}{l|l|l}
v[x \mapsto \underline{a}] & v[x \mapsto \underline{b}] & v[x \mapsto \underline{c}] \\
\hline
\langle \underline{a}, \underline{a} \rangle \in \llbracket R \rrbracket^M & \langle \underline{b}, \underline{a} \rangle \in \llbracket R \rrbracket^M & \langle \underline{c}, \underline{a} \rangle \in \llbracket R \rrbracket^M \\
\text{but} & & \\
\langle \underline{a}, \underline{a} \rangle \notin \llbracket R \rrbracket^M & \langle \underline{b}, \underline{a} \rangle \notin \llbracket R \rrbracket^M & \langle \underline{c}, \underline{a} \rangle \notin \llbracket R \rrbracket^M
\end{array}
$$

$$
\Rightarrow \llbracket \exists x R(x,y) \rrbracket^M_v = \mathbf{0}
$$

$$
\begin{aligned}
w &= \{x \mapsto \underline{a}, y \mapsto \underline{b}\} \\
\llbracket \exists x R(x,y) \rrbracket^M_w = \mathbf{1} &\Leftrightarrow \llbracket R(x,y) \rrbracket^M_{w[x \mapsto \underline{x}]} = \mathbf{1} \text{ for some } \underline{x} \in \underline{M} \\
&\Leftrightarrow \langle w[x \mapsto \underline{x}](x), w[x \mapsto \underline{x}](y) \rangle \in \llbracket R \rrbracket^M \text{ for some } \underline{x} \in \underline{M} \\
&\Leftrightarrow \langle w[x \mapsto \underline{x}](x), \underline{b} \rangle \in \llbracket R \rrbracket^M \text{ for some } \underline{x} \in \underline{M} \quad (\text{since } w[x \mapsto \underline{x}](y) = w(y) = \underline{b}) \\
&\Leftrightarrow \text{at least one of the following holds} \qquad (\text{we have } w[x \mapsto \underline{x}](x) = \underline{x})
\end{aligned}
$$

$$
\begin{array}{l|l|l}
w[x \mapsto \underline{a}] & w[x \mapsto \underline{b}] & w[x \mapsto \underline{c}] \\
\hline
\langle \underline{a}, \underline{b} \rangle \in \llbracket R \rrbracket^M & \langle \underline{b}, \underline{b} \rangle \in \llbracket R \rrbracket^M & \langle \underline{c}, \underline{b} \rangle \in \llbracket R \rrbracket^M \\
\text{and} & & \\
\langle \underline{a}, \underline{b} \rangle \in \llbracket R \rrbracket^M & \langle \underline{b}, \underline{b} \rangle \notin \llbracket R \rrbracket^M & \langle \underline{c}, \underline{b} \rangle \notin \llbracket R \rrbracket^M
\end{array}
$$

$$
\Rightarrow \llbracket \exists x R(x,y) \rrbracket^M_w = \mathbf{1}
$$

Thus, $\llbracket A \rrbracket^M_v = \mathbf{0}$ while $\llbracket A \rrbracket^M_w = \mathbf{1}$. Notice that the values assigned to the bound variable x by v and w do not matter at all – one has to consider $v[x \mapsto \underline{x}]$, resp. $w[x \mapsto \underline{x}]$ for all possible cases of $\underline{x}$. What made the difference was the fact that $v(y) = \underline{a}$ – for which no $\underline{x}$ could be found with $\langle \underline{x}, \underline{a} \rangle \in \llbracket R \rrbracket^M$, while $w(y) = \underline{b}$ – for which we found such an $\underline{x}$, namely $\langle \underline{a}, \underline{b} \rangle \in \llbracket R \rrbracket^M$.

Universal quantifier $\forall x$ has an entirely analogous effect – with the above two assignments, you may use the fact 8.5 directly to check that $\llbracket \forall x \neg R(x,y) \rrbracket^M_v = \mathbf{1}$, while $\llbracket \forall x \neg R(x,y) \rrbracket^M_w = \mathbf{0}$. $\square$

These facts – the influence of the names of free variables and irrelevance of the names of the bound ones on the truth of formulae – are expressed in the following lemma.

Lemma 8.7 Let M be a structure, A a formula and v and w two assignments such that for every free variable in A, $x \in \mathcal{V}(A) : v(x) = w(x)$. Then $[\![A]\!]^M_v = [\![A]\!]^M_w$.

Proof. Induction on A. For atomic A, the claim is obvious (for terms in $A : [\![t]\!]^M_v = [\![t]\!]^M_w$), and induction passes trivially through the connectives. So let A be a quantified formula $\exists y B$

$$[\![A]\!]^M_v = \mathbf{1} \Leftrightarrow \text{ for some } \underline{a} \in \underline{M} : [\![B]\!]^M_{v[y\mapsto\underline{a}]} = \mathbf{1} \overset{\text{IH}}{\Leftrightarrow} \text{ for some } \underline{a} \in \underline{M} : [\![B]\!]^M_{w[y\mapsto\underline{a}]} = \mathbf{1} \Leftrightarrow [\![A]\!]^M_w = \mathbf{1}$$

By IH, $[\![B]\!]^M_{v[y\mapsto\underline{a}]} = [\![B]\!]^M_{w[y\mapsto\underline{a}]}$, since $v(x) = w(x)$ for all free variables $x \in \mathcal{V}(A)$, and the modification $[y \mapsto \underline{a}]$ makes them agree on y as well, and hence on all the free variables of B. **QED** (8.7)

Remark.
The interpretation of constants, function and relation symbols does not depend on the assignment. Similarly, the interpretation of ground terms and ground formulae is independent of the assignments. For formulae even more is true – by lemma 8.7, the boolean value of any *closed* formula A does not depend on the assignment (since $\mathcal{V}(A) = \varnothing$, for any assignments v, w, we will have that for all $x \in \varnothing : v(x) = w(x)$.)

For this reason we may drop the subscript "v" in "$[\![A]\!]^M_v$" and write simply "$[\![A]\!]^M$" in case A is closed. Analogously, we may drop the "v" in "$[\![t]\!]^M_v$" if t is a ground term.

Example 8.8
Consider the alphabet Σ_{Stack} from example 7.2. We design a Σ_{Stack}-structure M as follows:

- the underlying set $\underline{M} = A \uplus A^*$ consists of a non-empty set A and all finite strings A^* of A-elements; below $w, v \in \underline{M}$ are arbitrary, while $a \in A$:
- $[\![empty]\!]^M = \epsilon \in A^*$ – the empty string
- $[\![pop]\!]^M(wa) = w$ and $[\![pop]\!]^M(\epsilon) = \epsilon$
- $[\![top]\!]^M(wa) = a$ and $[\![pop]\!]^M(\epsilon) = a_0$ for some $a_0 \in A$
- $[\![push]\!]^M(w, v) = wv$ – concatenation of strings: the string w with v concatenated at the end
- finally, we let $[\![El]\!]^M = A$, $[\![St]\!]^M = A^*$ and $[\![\equiv]\!]^M = \{\langle m, m\rangle : m \in \underline{M}\}$, i.e., the identity relation.

We have given an interpretation to all the symbols from Σ_{Stack}, so M is a Σ_{Stack}-structure. Notice that all functions are total, i.e., defined for all elements of $\underline{M}$.

We now check the value of all the axioms from example 7.15 in M. We apply def. 8.4, fact 8.5 and lemma 8.7 (its consequence from the remark before this example, allowing us to drop assignments whenever interpreting ground terms and closed formulae.).

(1) $[\![St(empty)]\!]^M = [\![empty]\!]^M \in [\![St]\!]^M = \epsilon \in A^* = \mathbf{1}$.

(2) $[\![\forall x, s : El(x) \wedge St(s) \rightarrow St(push(s,x))]\!]^M = \mathbf{1} \quad \Leftrightarrow$
for all $c, d \in \underline{M} : [\![El(x) \wedge St(s) \rightarrow St(push(s,x))]\!]^M_{[x \mapsto c, s \mapsto d]} = \mathbf{1} \quad \Leftrightarrow$
for all $c, d \in \underline{M} : [\![El(x) \wedge St(s)]\!]^M_{[x \mapsto c, s \mapsto d]} = \mathbf{0}$ or $[\![St(push(s,x))]\!]^M_{[x \mapsto c, s \mapsto d]} = \mathbf{1} \quad \Leftrightarrow$
for all $c, d \in \underline{M} : [\![El(x)]\!]^M_{[x \mapsto c, s \mapsto d]} = \mathbf{0}$ or $[\![St(s)]\!]^M_{[x \mapsto c, s \mapsto d]} = \mathbf{0}$ or $[\![St(push(s,x))]\!]^M_{[x \mapsto c, s \mapsto d]} = \mathbf{1} \quad \Leftrightarrow$
for all $c, d \in \underline{M} : c \notin [\![El]\!]^M$ or $d \notin [\![St]\!]^M$ or $[\![push(s,x)]\!]^M_{[x \mapsto c, s \mapsto d]} \in [\![St]\!]^M \quad \Leftrightarrow$
for all $c, d \in \underline{M} : c \notin A$ or $d \notin A^*$ or $dc \in A^* \quad \Leftrightarrow$
true, since whenever $c \in A$ and $d \in A^*$ then also $dc \in A^*$.

(3) We drop axioms 3.-4. and go directly to 5. and 6.

(5) $[\![pop(empty) \equiv empty]\!]^M = \mathbf{1} \quad \Leftrightarrow \quad \langle [\![pop(empty)]\!]^M, [\![empty]\!]^M \rangle \in [\![\equiv]\!]^M \quad \Leftrightarrow$
$\langle [\![pop]\!]^M(\epsilon), \epsilon \rangle \in [\![\equiv]\!]^M \quad \Leftrightarrow \quad \langle \epsilon, \epsilon \rangle \in [\![\equiv]\!]^M \quad \Leftrightarrow \quad \epsilon = \epsilon \quad \Leftrightarrow$ true.

(6) $[\![\forall x, s : El(x) \wedge St(s) \rightarrow pop(push(s,x)) \equiv s]\!]^M = \mathbf{1} \quad \Leftrightarrow$
for all $c, d \in \underline{M} : [\![El(x) \wedge St(s) \rightarrow pop(push(s,x)) \equiv s]\!]^M_{[x \mapsto c, s \mapsto d]} = \mathbf{1} \quad \Leftrightarrow$
for all $c, d \in \underline{M} : [\![El(x) \wedge St(s)]\!]^M_{[x \mapsto c, s \mapsto d]} = \mathbf{0}$ or $[\![pop(push(s,x)) \equiv s]\!]^M_{[x \mapsto c, s \mapsto d]} = \mathbf{1} \quad \Leftrightarrow$
for all $c, d \in \underline{M} : c \notin [\![El]\!]^M$ or $d \notin [\![St]\!]^M$ or $\langle [\![pop]\!]^M([\![push]\!]^M(d,c)), d \rangle \in [\![\equiv]\!]^M \quad \Leftrightarrow$
for all $c, d \in \underline{M} : c \notin A$ or $d \notin A^*$ or $[\![pop]\!]^M([\![push]\!]^M(d,c)) = d \quad \Leftrightarrow$
for all $c, d \in \underline{M}$: if $c \in A$ and $d \in A^*$ then $[\![pop]\!]^M(dc) = d \quad \Leftrightarrow$
for all $c, d \in \underline{M}$: if $c \in A$ and $d \in A^*$ then $d = d \quad \Leftrightarrow$ true.
For the last transition it is essential that $c \in A$ – if c is a non-empty string, e.g. ab then $[\![pop]\!]^M(dab) = da \neq d$. Axiom 7. can be verified along the same lines.

□

Given a structure, there will be typically infinitely many formulae which evaluate to **1** in it. In this example, for instance, also the following formula will evaluate to **1** : $\forall w, x : St(x) \wedge \neg(x \equiv empty) \rightarrow pop(push(w,x)) \equiv$

$push(w, pop(x))$. It goes counter our intuitive understanding of stacks – we do not push stacks on stacks, only elements. This, however, is an accident caused by the fact that all functions must be total in any structure and, more importantly, by the specific definition of our structure M. What in programming languages are called types (or sorts) is expressed in FOL by additional predicates. Thus, we do not have a 'type' stacks or elements (integers, etc.) which would prevent one from applying the function *top* to the elements of the latter type. We have predicates which describe a part of the whole domain. The axioms, as in the example above, can be used to specify what should hold provided that argumants come from the right parts of the domain (types). But the requirement of totality on all functions forces them to yield some results also when applied outside such intended definition domains. Thus, the axioms from Example 7.15 do not describe *uniquely* the particular data type stacks we might have in mind. They only define some minimal properties which such data type whould satisfy.

Lemma 8.9 Let t_1, t_2 be both substitutable for x in A. If $[\![t_1]\!]^M_v = [\![t_2]\!]^M_v$ then $[\![A^x_{t_1}]\!]^M_v = [\![A^x_{t_2}]\!]^M_v$.

Proof. By induction on the complexity of A.

BASIS :: It is easy to show (by induction on the complexity of terms s) that $[\![s^x_{t_1}]\!]^M_v = [\![s^x_{t_2}]\!]^M_v$ if $[\![t_1]\!]^M_v = [\![t_2]\!]^M_v$. Hence $[\![R(s_1,\ldots,s_n)^x_{t_1}]\!]^M_v = [\![R(s_1,\ldots,s_n)^x_{t_2}]\!]^M_v$ if $[\![t_1]\!]^M_v = [\![t_2]\!]^M_v$.

IND. :: The induction steps for $\neg$ and $\rightarrow$ are trivial, as is the case for $\exists x$ for the same variable x. Now suppose A is $\exists yB$ and $y \neq x$.
Since t_1 and t_2 are substitutable for x in A it follows that either (1) x does not occur free in B, in which case the proof is trivial, or (2a) y does not occur in t_1, t_2 and (2b) t_1, t_2 are both substitutable for x in B. Now $(\exists yB)^x_{t_i} = \exists y(B^x_{t_i})$ and hence $[\![(\exists yB)^x_{t_i}]\!]^M_v = 1$ iff $[\![B^x_{t_i}]\!]^M_{v[y\mapsto \underline{a}]} = 1$ for some $\underline{a}$. By 2a we know that $[\![t_1]\!]^M_{v[y\mapsto \underline{a}]} = [\![t_2]\!]^M_{v[y\mapsto \underline{a}]}$ for all $\underline{a}$, so by 2b and the IH we know that $[\![B^x_{t_1}]\!]^M_{v[y\mapsto \underline{a}]} = 1$ for some $\underline{a}$ iff $[\![B^x_{t_2}]\!]^M_{v[y\mapsto \underline{a}]} = 1$ for some $\underline{a}$. **QED** (8.9)

Remark.
This result is crucial when you prove the soundness of $A4$ in exercise 8.4. A

useful special case arises if t_1 or t_2 is x itself. In that case the lemma says that the identity $v(x) = [\![t]\!]_v^M$ implies the identity $[\![A]\!]_v^M = [\![A_t^x]\!]_v^M$, provided t is substitutable for x in A.

2: Semantic properties of formulae

Definition 8.10 Below the letters "A", "M" and "v" range over formulae, structures, and variable-assignments into $\underline{M}$, respectively.

A is	iff	condition holds	notation:
true with respect to M and v	iff	$[\![A]\!]_v^M = \mathbf{1}$	$M \models_v A$
false with respect to M and v	iff	$[\![A]\!]_v^M = \mathbf{0}$	$M \not\models_v A$
satisfied in M	iff	for all $v : M \models_v A$	$M \models A$
not satisfied in M	iff	there is a $v : M \not\models_v A$	$M \not\models A$
valid	iff	for all $M : M \models A$	$\models A$
not valid	iff	there is an $M : M \not\models A$	$\not\models A$
satisfiable	iff	there is an $M : M \models A$	
unsatisfiable	iff	for all $M : M \not\models A$	

Notation.
Sometimes, instead of saying "A is satisfied (not satisfied) in M", one says that "A is true (false)", or even that it is "valid (not valid) in M".

Lemma 8.7 tells us that only a part of v is relevant in "$M \models_v A$", namely the partial function that identifies the values for $x \in \mathcal{V}(A)$. If $\mathcal{V}(A) \subseteq \{x_1, \ldots, x_n\}$ and $\underline{a}_i = v(x_i)$, we may write just "$M \models_{\{x_1 \mapsto \underline{a}_1, \ldots, x_n \mapsto \underline{a}_n\}} A$." We may even drop the curly braces, since they only clutter up the expression.

Example 8.11
In example 8.8 we have shown that the structure M models each axiom ϕ of stacks, $M \models \phi$, from example 7.15.

Recall now example 7.18 of an alphabet Σ_{SG} for simple graphs containing only one binary relation symbol E. Any set $\underline{U}$ with a binary relation R on it is an Σ_{SG}-structure, i.e., we let $[\![E]\!]^U = R \subseteq \underline{U} \times \underline{U}$.

We now want to make it to satisfy axiom 1 from 7.18: $U \models \forall x, y : E(x,y) \to E(y,x)$, i.e., by definition 8.10, we want

$$\text{for all assignments } v : [\![\forall x, y : E(x,y) \to E(y,x)]\!]_v^U = \mathbf{1}. \tag{8.12}$$

Since the formula is closed, we may ignore assignments – by fact 8.5 we need:

$$\text{for all } \underline{a}, \underline{b} \in \underline{U} : [\![E(x,y) \to E(y,x)]\!]^U_{[x \mapsto \underline{a}, y \mapsto \underline{b}]} = \mathbf{1}, \tag{8.13}$$

By definition 8.4.2 this means:

$$\text{for all } \underline{a}, \underline{b} \in \underline{U} : [\![E(x,y)]\!]^U_{[x\mapsto\underline{a}, y\mapsto\underline{b}]} = \mathbf{0} \text{ or } [\![E(y,x)]\!]^U_{[x\mapsto\underline{a}, y\mapsto\underline{b}]} = \mathbf{1}, \quad (8.14)$$

i.e., by definition 8.4.1:

$$\text{for all } \underline{a}, \underline{b} \in \underline{U} : \langle \underline{a}, \underline{b} \rangle \notin [\![E]\!]^U \text{ or } \langle \underline{b}, \underline{a} \rangle \in [\![E]\!]^U, \quad (8.15)$$

and since $[\![E]\!]^U = R$:

$$\text{for all } \underline{a}, \underline{b} \in \underline{U} : \langle \underline{a}, \underline{b} \rangle \notin R \text{ or } \langle \underline{b}, \underline{a} \rangle \in R. \quad (8.16)$$

But this says just that for any pair of elements $\underline{a}, \underline{b} \in \underline{U}$, if $R(\underline{a}, \underline{b})$ then also $R(\underline{b}, \underline{a})$. Thus the axiom holds only in the structures where the relation (here R) interpreting the symbol E is symmetric and does not hold in those where it is not symmetric. Put a bit differently – and this is the way one uses (non-logical) axioms – the axiom *selects only* those Σ_{SG}-structures where the relation interpreting E is symmetric – it narrows the relevant structures to those satisfying the axiom.

Quite an anlogous procedure would show that the other axioms from example 7.18, would narrow the possible interpretations to those where R is transitive, resp. irreflexive. □

3: Open vs. closed formulae

Semantic properties of closed formulae bear some resemblance to the respective properties of formulae of PL. This resemblance does not apply with equal strength to open formulae. We start by a rough comparison to PL.

Remark 8.17 **[Comparing with PL]**
Comparing the table from definition 8.10 with definition 5.8, we see that the last three double rows correspond exactly to the definition for PL. The first double row is new because now, in addition to formulae and structures, we also have valuation of individual variables. As before, *contradictions* are the unsatisfiable formulae, and the structure M is a *model* of A if A is satisfied (valid) in $M : M \models A$. (Notice that if A is valid in an M, it is not valid (in general) but only satisfiable.)

An important difference from PL concerns the relation between $M \not\models A$ and $M \models \neg A$. In PL, we had that for any structure V and formula A

$$V \not\models A \Rightarrow V \models \neg A \quad (8.18)$$

simply because any V induced unique boolean value for all formulae. The corresponding implication does not, in general, hold in FOL:

$$M \not\models A \not\Rightarrow M \models \neg A \quad (8.19)$$

In fact, we may have:

$$M \not\models A \textit{ and } M \not\models \neg A \tag{8.20}$$

(Of course, we will never get $M \models A$ and $M \models \neg A$.) To see (8.20), consider a formula $A = P(x)$ and a structure M with two elements $\{0, 1\}$ and $[\![P]\!]^M = \{1\}$. Then $M \not\models A \Leftrightarrow M \not\models_v A$ for some $v \Leftrightarrow [\![A]\!]^M_v = \mathbf{0}$ for some v, and this is indeed the case for $v(x) = 0$ since $0 \notin [\![P]\!]^M$. On the other hand, we also have $M \not\models \neg A$ since $[\![\neg A]\!]^M_w = \mathbf{0}$ for $w(x) = 1$.

It is the presence of free variables which causes the implication in (8.19) to fail because then the interpretation of a formula is not uniquely determined unless one specifies an assignment to the free variables. Given an assignment, we do have

$$M \not\models_v A \Rightarrow M \models_v \neg A \tag{8.21}$$

Consequently, the implication (8.19) holds for closed formulae.

Summarizing this remark, we can set up the following table which captures some of the essential difference between free and bound variables in terms of relations between negation of satisfaction of a formula, $M \not\models F$, and satisfaction of its negation, $M \models \neg F$ – M is an arbitrary structure:

$$\begin{array}{c|c} F \textit{ closed (or } \mathsf{PL}) & F \textit{ open} \\ \hline M \not\models F \;\Leftrightarrow\; M \models \neg F & M \not\models F \;\not\Rightarrow\; M \models \neg F \\ & M \not\models F \;\Leftarrow\; M \models \neg F \end{array} \tag{8.22}$$

For closed formulae, we can say that "negation commutes with satisfaction" (or else "satisfaction of negation *is the same as* negation of satisfaction"), while this is not the case for open formulae.

The above remark leads to another important difference between FOL and PL, which you should have noticed. The respective Deduction Theorems 4.13 and 7.30 differ in that the latter has the additional restriction of closedness. The reason for this is the semantic complications introduced by the free variables. For PL we defined two notions $B \Rightarrow A$ and $B \models A$ which, as a matter of fact, coincided. We had there the following picture:

$$\begin{array}{ccc} \text{for all } V : V \models B \to A \overset{\text{D.5.8}}{=} \models B \to A & & B \models A \overset{\text{D.6.11}}{=} \text{for all } V: \text{ if } V \models B \text{ then } V \models A \\ \Updownarrow {\scriptstyle \text{C.6.26}} & & {\scriptstyle \text{C.6.26}} \Updownarrow \\ \vdash_{\mathcal{N}} B \to A & \overset{\text{C.4.16}}{\Longleftrightarrow} & B \vdash_{\mathcal{N}} A \end{array}$$

The vertical equivalences follow from the soundness and completeness theorems, while the horizontal one follows from Deduction Theorem 4.13. This chain of equivalences is a roundabout way to verify that for $\mathsf{PL} : B \models A$ iff $B \Rightarrow A$.

The picture isn't that simple in FOL. We preserve the two definitions 5.8 and 6.11:

Definition 8.23 For FOL structures M and formulae A, B :

- $B \models A$ iff for any structure M : if $M \models B$ then $M \models A$.
- A is a logical consequence of B, written $B \Rightarrow A$, iff $\models B \to A$.

However, $M \models B$ means that $M \models_v B$ holds *for all assignments* v. Thus the definitions read:

$B \models A$	$B \Rightarrow A$
for all M : if (for all $v : M \models_v B$) then (for all $u : M \models_u A$)	for all M : for all $v : M \models_v B \to A$

It is not obvious that the two are equivalent – in fact, they are not, if there are free variables involved (exercise 8.7). If there are no free variables then, by Lemma 8.7, each formula has a fixed boolean value in any structure (and we can remove the quantification over v's and u's from the above table). The difference is expressed in the restriction of Deduction Theorem 7.30 requiring the formula B to be closed. (Assuming soundness and completeness (chapter 10), this restriction leads to the same equivalences as above for PL.) We write '$A \Leftrightarrow B$' for '$A \Rightarrow B$ and $B \Rightarrow A$' which, by the above remarks (and exercise 8.7) is *not* the same as '$A \models B$ and $B \models A$'.

Fact 8.24. For any formula $A : \neg\forall x A \Leftrightarrow \exists x \neg A$.

Proof. By the definition 7.7 of the abbreviation $\forall x A$, the left hand side is the same as $\neg\neg\exists x \neg A$, which is equivalent to the right-hand side. **QED** (8.24)

Fact 8.25. $\forall x A \Rightarrow A$

Proof. We have to show $\models \forall x A \to A$, that is, for arbitrary structure $M : M \models \forall x A \to A$, that is, for arbitrary assignment $v : M \models_v \forall x A \to A$. Let M, v be arbitrary. If $[\![\forall x A]\!]^M_v = \mathbf{0}$ then we are done, so assume $[\![\forall x A]\!]^M_v = \mathbf{1}$.

$$
\begin{array}{rll}
[\![\forall x A]\!]^M_v = \mathbf{1} & \Leftrightarrow [\![\neg\exists x \neg A]\!]^M_v = \mathbf{1} & \text{def. 7.7} \\
& \Leftrightarrow [\![\exists x \neg A]\!]^M_v = \mathbf{0} & \text{def. 8.4.2} \\
& \Leftrightarrow \text{for no } \underline{a} \in \underline{M} : [\![\neg A]\!]^M_{v[x \mapsto \underline{a}]} = \mathbf{1} & \text{def. 8.4.3} \\
& \Leftrightarrow \text{for all } \underline{a} \in \underline{M} : [\![\neg A]\!]^M_{v[x \mapsto \underline{a}]} = \mathbf{0} & \text{same as above} \\
& \Rightarrow [\![\neg A]\!]^M_{v[x \mapsto v(x)]} = \mathbf{0} & \text{particular case for } \underline{a} = v(x) \\
& \Leftrightarrow [\![\neg A]\!]^M_v = \mathbf{0} & \text{since } v = v[x \mapsto v(x)] \\
& \Leftrightarrow [\![A]\!]^M_v = \mathbf{1} & \text{def. 8.4.2}
\end{array}
$$

Since M and v were arbitrary, the claim follows. **QED** (8.25)

For a closed formula A we have, obviously, equivalence of A and $\forall xA$, since the latter does not modify the formula. For open formulae, however, the implication opposite to this from fact 8.25 does not hold.

Fact 8.26. For open A with a free variable $x : A \not\Rightarrow \forall xA$.

Proof. Let A be $P(x)$, where P is a predicate symbol. To see that $\not\models P(x) \to \forall xP(x)$, consider a structure M with $\underline{M} = \{p, q\}, \llbracket P \rrbracket^M = \{p\}$ and an assignment $v(x) = p$. We have that $M \not\models_v P(x) \to \forall xP(x)$, since $\llbracket P(x) \rrbracket^M_v = \mathbf{1}$ while $\llbracket \forall xP(x) \rrbracket^M_v = \llbracket \forall xP(x) \rrbracket^M = \mathbf{0}$. **QED** (8.26)

In spite of this fact, we have another close relation between satisfaction of A and $\forall xA$. Given a (not necessarily closed) formula A, the *universal closure* of A, written $\forall(A)$, is the formula $\forall x_1 \ldots \forall x_n\ A$, where $\{x_1, \ldots, x_n\} = \mathcal{V}(A)$. The following fact shows that satisfaction of a – possibly open! – formula in a given structure is, in fact, equivalent to the satisfaction of its universal closure.

Fact 8.27. For an arbitrary structure M and formula A, the following are equivalent:

1. $M \models A$ 2. $M \models \forall(A)$

Proof. Basically, we should proceed by induction on the number of free variables in A but this does not change anything essential in the proof. We therefore write the universal closure as $\forall\overline{x}A$, where '$\forall\overline{x}$' stands for '$\forall x_1 \forall x_2 ... \forall x_n$'. We show both implications contrapositively.

$1 \Leftarrow 2$		$2 \Leftarrow 1$	
$M \not\models A$	$\stackrel{8.4}{\iff} M \not\models_v A$ for some v	$M \not\models \forall\overline{x}A$	$\stackrel{7.7}{\iff} M \not\models \neg\exists\overline{x}\neg A$
	$\stackrel{8.10}{\iff} \llbracket A \rrbracket^M_v = \mathbf{0}$		$\stackrel{8.4}{\iff} \llbracket \neg\exists\overline{x}\neg A \rrbracket^M_v = \mathbf{0}$ for some v
	$\stackrel{8.4}{\iff} \llbracket \neg A \rrbracket^M_v = \mathbf{1}$		$\stackrel{8.4}{\iff} \llbracket \exists\overline{x}\neg A \rrbracket^M_v = \mathbf{1}$ for same v
	$\stackrel{8.4}{\Longrightarrow} \llbracket \exists\overline{x}\neg A \rrbracket^M_v = \mathbf{1}$		$\stackrel{8.4}{\iff} \llbracket \neg A \rrbracket^M_{v[\overline{x}\mapsto\overline{a}]} = \mathbf{1}$ for some $\overline{a} \in \underline{M}$
	$\stackrel{8.4}{\iff} \llbracket \neg\exists\overline{x}\neg A \rrbracket^M_v = \mathbf{0}$		$\stackrel{8.4}{\iff} \llbracket A \rrbracket^M_{v[\overline{x}\mapsto\overline{a}]} = \mathbf{0}$
	$\stackrel{8.10}{\iff} M \not\models_v \neg\exists\overline{x}\neg A$		$\stackrel{8.10}{\iff} M \not\models_{v[\overline{x}\mapsto\overline{a}]} A$
	$\stackrel{8.10}{\Longrightarrow} M \not\models \neg\exists\overline{x}\neg A$		$\stackrel{8.10}{\Longrightarrow} M \not\models A$
	$\stackrel{7.7}{\iff} M \not\models \forall\overline{x}A$		

QED (8.27)

3.1: Deduction Theorem in $\mathcal{G}$ and $\mathcal{N}$

Observe that Gentzen's rules 2. and 3'. (from chapter 7, section 4) indicate the semantics of sequents. $A_1 ... A_n \vdash_{\mathcal{G}} B_1 ... B_m$ corresponds to $A_1 \wedge ... \wedge A_n \vdash_{\mathcal{G}} B_1 \vee ... \vee B_m$. Then, we can use rule 5. to obtain $\vdash_{\mathcal{G}} (A_1 \wedge ... \wedge A_n) \to (B_1 \vee ... \vee B_m)$ which is a simple formula (not a proper sequent) with the expected semantics corresponding to the semantics of the original sequent.

Now, $\mathcal{G}$, unlike $\mathcal{N}$, for FOL is a *truly* natural deduction system. The rule 5. is the unrestricted Deduction Theorem built into $\mathcal{G}$. Recall that it was not so for $\mathcal{N}$ – Deduction Theorem 7.30 allowed us to use a restricted version of the rule: $\dfrac{\Gamma, A \vdash_{\mathcal{N}} B}{\Gamma \vdash_{\mathcal{N}} A \to B}$ only if A is closed! Without this restriction, the rule would be unsound, e.g.:

1. $A \vdash_{\mathcal{N}} A$ $\quad A0$
2. $A \vdash_{\mathcal{N}} \forall x A$ $\quad L.7.25.4$
3. $\vdash_{\mathcal{N}} A \to \forall x A$ $\quad DT!$
4. $\vdash_{\mathcal{N}} \exists x A \to \forall x A$ $\exists$I

The conclusion of this proof is obviously unsound (verify this) and we could derive it only using a wrong application of DT in line 3.

In $\mathcal{G}$, such a proof cannot proceed beyond step 1. Rule 7. requires replacement of x from $\forall x A$ by a *fresh* x' – and it must be fresh, i.e., not occurring, in the whole sequent! Attempting this proof in $\mathcal{G}$ would lead to the following:

4. $A(x') \vdash_{\mathcal{G}} A(x)$ $\quad 7.\ x\ fresh\ (x \neq x')$
3. $A(x') \vdash_{\mathcal{G}} \forall x A(x)$ $\quad 7'.\ x'\ fresh$
2. $\exists x A(x) \vdash_{\mathcal{G}} \forall x A(x)$ $\quad 5.$
1. $\vdash_{\mathcal{G}} \exists x A(x) \to \forall x A(x)$

But $A(x) \neq A(x')$ so line 4. is not an axiom. (If x does not occur in A (i.e., quantification $\forall x A$ is somehow redundant) then this would be an axiom and everything would be fine.)

It is this, different than in $\mathcal{N}$, treatement of variables (built into the different quantifier rules) which enables $\mathcal{G}$ to use unrestricted Deduction Theorem. It is reflected at the semantic level in that the semantics of $\vdash_{\mathcal{G}}$ is different from $\vdash_{\mathcal{N}}$. According to definition 8.23, $A \models B$ iff $\models \forall(A) \to \forall(B)$ and this is reflected in $\mathcal{N}$, e.g., in the fact that from $A \vdash_{\mathcal{N}} B$ we can deduce that $\forall(A) \vdash_{\mathcal{N}} \forall(B)$ from which $\vdash_{\mathcal{N}} \forall(A) \to \forall(B)$ follows now by Deduction Theorem.

The semantics of $A \vdash_{\mathcal{G}} B$ is different – such a sequent is interpreted as $A \Rightarrow B$, that is, $\models \forall(A \to B)$. The free variables occurring in both A and B are now interpreted in the same way across the sign $\vdash_{\mathcal{G}}$. Using

Definition 8.23, one translates easily between sequents and formulae of $\mathcal{N}$ (M is an arbitrary structure). The first line is the general form from which the rest follows. The last line expresses perhaps most directly the difference in treatement of free variables which we indicated above.

$$\begin{array}{lrcr}
M \models & (A_1, .., A_n \vdash_{\mathcal{G}} B_1, .., B_m) & \Leftrightarrow & M \models \forall((A_1 \wedge .. \wedge A_n) \to (B_1 \vee .. \vee B_m)) \\
M \models & (\varnothing \vdash_{\mathcal{G}} B_1, .., B_m) & \Leftrightarrow & M \models \forall(B_1 \vee .. \vee B_m) \\
M \models & (A_1, .., A_n \vdash_{\mathcal{G}} \varnothing) & \Leftrightarrow & M \models \forall(\neg(A_1 \wedge .. \wedge A_n)) \\
& \models (\forall(A_1), .., \forall(A_n) \vdash_{\mathcal{G}} B) & \Leftrightarrow & \{A_1 ... A_n\} \models B
\end{array}$$

Exercises 8.

EXERCISE 8.1 Show that the follwing rule is admissible: $\dfrac{\Gamma \vdash_{\mathcal{N}} A \to B}{\Gamma \vdash_{\mathcal{N}} \forall x A \to \forall x B}$

(where there are no side-conditions on x).

(Hint: Lemma 7.25.2, and two applications of some relevant result from PL.)

EXERCISE 8.2 Translate each of the following sentences into a FOL language (choose the needed relations yourself)

(1) *Everybody loves somebody.*
(2) *If everybody is loved (by somebody) then somebody loves everybody.*
(3) *If everybody loves somebody and John does not love anybody then John is nobody.*

At least two of the obtained formulae are not valid. Which ones? Is the third one valid?

EXERCISE 8.3 Using the previous exercise, construct a structure where the opposite implication to the one from exercise 7.2.3, i.e., $\forall y \exists x A \to \exists x \forall y A$, does not hold.

EXERCISE 8.4 Verify the following facts ($\models A \leftrightarrow B$ stands for $\models A \to B$ and $\models B \to A$)

(1) $\exists x A \Leftrightarrow \exists y A^x_y$, when y does not occur in A.
(2) $\models \forall x A \leftrightarrow \forall y A^x_y$, when y does not occur in A.
(3) $A^x_t \Rightarrow \exists x A$, when t is substitutable for x in A.
Show that the assumption about substitutability is necessary, i.e., give an example of a non-valid formula of the form $A^x_t \to \exists x A$, when t is *not* substitutable for x in A.

EXERCISE 8.5 Show that $(\forall x A \vee \forall x B) \Rightarrow \forall x(A \vee B)$. Give a counterexample demonstrating that the opposite implication (which you hopefully did not manage to prove in exercise 7.3) does not hold.

<u>EXERCISE 8.6</u> Show that $M \models \neg A$ implies $M \not\models A$. Give a counterexample demonstrating that the opposite implication need not hold, i.e. find an M and A over appropriate alphabet such that $M \not\models A$ *and* $M \not\models \neg A$. (Hint: A must have free variables.)

<u>EXERCISE 8.7</u> Recall remark 8.17 and discussion after definition 8.23 (as well as exercise 7.8).

(1) Show that $\forall(B \to A) \Rightarrow (\forall(B) \to \forall(A))$.
  – Use this to show that: if $B \Rightarrow A$ then $B \models A$.
(2) Give an argument (an example of A, B and structure) falsifying the opposite implication, i.e., showing $\not\models (\forall(B) \to \forall(A)) \to \forall(B \to A)$.
  – Use this to show that: if $B \models A$ then it need not be the case that $B \Rightarrow A$.

optional

<u>EXERCISE 8.8</u> Write the following sentences in FOL – choose appropriate alphabet of non-logical symbols (lower case letters $a, b, \ldots$ are individual constants):

1. Either a is small or both c and d are large.
2. d and e are both in back of b and larger than it.
3. Either e is not large or it is in back of a.
4. Neither e nor a are to the right of c and to the left of b.

<u>EXERCISE 8.9</u> Use fact 8.27 to show that the following two statements are equivalent for any formulae A, B

(1) $B \models A$
(2) $\forall(B) \models A$

<u>EXERCISE 8.10</u> In lemma 7.25.5 we showed admissibility of the substitution rule in $\mathcal{N}$:

SB: $\dfrac{\Gamma \vdash_{\mathcal{N}} A}{\Gamma \vdash_{\mathcal{N}} A^x_t}$ if t is substitutable for x in A

Show now that this rule is sound, i.e., for any FOL-structure M : if $M \models A$ then also $M \models A^x_t$ when t is substitutable for x in A.

Chapter 9
More Semantics

- Prenex NF
- A Bit of Model Theory
- Term structures

1: Prenex operations

We have shown (Corollary 6.6, 6.7) that each PL formula can be equivalently written in DNF and CNF. A normal form which is particularly useful in the study of FOL is

Definition 9.1 **[Prenex normal form]** A formula A is in PNF iff it has the form
$\mathsf{Q}_1 x_1 \ldots \mathsf{Q}_n x_n B$, where Q_i are quantifiers and B contains no quantifiers.

The quantifier part $\mathsf{Q}_1 x_1 \ldots \mathsf{Q}_n x_n$ is called the *prefix*, and the quantifier free part B the *matrix* of A.

To show that each formula is equivalent to some formula in PNF we need the next lemma.

Lemma 9.2 Let A, B be formulae, $F[A]$ be a formula with some occurrence(s) of A, and $F[B]$ be the same formula with the occurrence(s) of A replaced by B. If $A \Leftrightarrow B$ then $F[A] \Leftrightarrow F[B]$.

Proof. The version for PL was proved in exercise 5.9. The proof goes by induction on the complexity of $F[A]$, with a special case considered first:

$\underline{F[A] \text{ IS :}}$

A :: This is a special case in which we have trivially $F[A] = A \Leftrightarrow B = F[B]$. So assume that we are not in the special case.

Atomic :: If $F[A]$ is atomic then either we have the special case, or no replacement is made, i.e., $F[A] = F[B]$, since F has no subformula A.

$\neg C[A]$ :: By IH $C[A] \Leftrightarrow C[B]$. So $\neg C[A] \Leftrightarrow \neg C[B]$.

$C[A] \to D[A]$ :: Again, IH gives $C[A] \Leftrightarrow C[B]$ and $D[A] \Leftrightarrow D[B]$, from which the conclusion follows (as in exercise 5.9).

$\exists x C[A]$ :: By IH, $C[A] \Leftrightarrow C[B]$. It means, that for all assignments to the variables, including x, the two are equivalent. Hence $\exists x C[A] \Leftrightarrow \exists x C[B]$.

QED (9.2)

Lemma 9.3 The **prenex operations** are given by the following equivalences:

(1) Quantifier movement along $\to$:

$$A \to \forall x B \;\Leftrightarrow\; \forall x(A \to B) \qquad\qquad A \to \exists x B \;\Leftrightarrow\; \exists x(A \to B)$$
$$\text{if } x \text{ not free in } A$$
$$\forall x A \to B \;\Leftrightarrow\; \exists x(A \to B) \qquad\qquad \exists x A \to B \;\Leftrightarrow\; \forall x(A \to B)$$
$$\text{if } x \text{ not free in } B$$

(2) Quantifier movement along $\neg$: $\neg\exists x A \Leftrightarrow \forall x \neg A$, and $\neg\forall x A \Leftrightarrow \exists x \neg A$.

(3) Renaming of bound variables: $\exists x A \Leftrightarrow \exists y A^x_y$, and $\forall x A \Leftrightarrow \forall y A^x_y$, when y does not occur in A.

Proof. 3. was proved in exercise 8.4. We show the first and third equivalence of 1; the rest is left for exercise 9.3. Let M be an arbitrary structure and v an arbitrary assignment to the free variables occurring in A and $\forall x B$. We have

$$\begin{array}{lll}
1.\ M \models_v A \to \forall x B & \Leftrightarrow & [\![A \to \forall x B]\!]^M_v = \mathbf{1} \\
2. & \Leftrightarrow & [\![A]\!]^M_v = \mathbf{0} \text{ or } [\![\forall x B]\!]^M_v = \mathbf{1} \\
3. & \Leftrightarrow & [\![A]\!]^M_v = \mathbf{0} \text{ or for all } \underline{a} \in \underline{M} : [\![B]\!]^M_{v[x\mapsto\underline{a}]} = \mathbf{1} \\
4. & \Leftrightarrow & \text{for all } \underline{a} \in \underline{M}\ \ ([\![A]\!]^M_v = \mathbf{0} \text{ or } [\![B]\!]^M_{v[x\mapsto\underline{a}]} = \mathbf{1}) \\
5. & \Leftrightarrow & \text{for all } \underline{a} \in \underline{M}\ ([\![A]\!]^M_{v[x\mapsto\underline{a}]} = \mathbf{0} \text{ or } [\![B]\!]^M_{v[x\mapsto\underline{a}]} = \mathbf{1}) \\
6. & \Leftrightarrow & [\![\forall x(A \to B)]\!]^M_v = \mathbf{1} \\
7. & \Leftrightarrow & M \models_v \forall x(A \to B)
\end{array}$$

The equivalence between lines 4 and 5 follows from lemma 8.7 because

x is not free in A. For the third equivalence, we have

$$\begin{array}{lll}
1.\ M \models_v \forall xA \to B & \Leftrightarrow & [\![\forall xA \to B]\!]^M_v = \mathbf{1} \\
2. & \Leftrightarrow & [\![\forall xA]\!]^M_v = \mathbf{0} \text{ or } [\![B]\!]^M_v = \mathbf{1} \\
3. & \Leftrightarrow & [\![\neg\exists x\neg A]\!]^M_v = \mathbf{0} \text{ or } [\![B]\!]^M_v = \mathbf{1} \\
4. & \Leftrightarrow & [\![\exists x\neg A]\!]^M_v = \mathbf{1} \text{ or } [\![B]\!]^M_v = \mathbf{1} \\
5. & \Leftrightarrow & (\text{ for some } \underline{a} \in \underline{M}\ [\![\neg A]\!]^M_{v[x\mapsto\underline{a}]} = \mathbf{1}) \text{ or } [\![B]\!]^M_v = \mathbf{1} \\
6. & \Leftrightarrow & (\text{ for some } \underline{a} \in \underline{M}\ [\![A]\!]^M_{v[x\mapsto\underline{a}]} = \mathbf{0}) \text{ or } [\![B]\!]^M_v = \mathbf{1} \\
7. & \Leftrightarrow & \text{for some } \underline{a} \in \underline{M}\ (\ [\![A]\!]^M_{v[x\mapsto\underline{a}]} = \mathbf{0} \text{ or } [\![B]\!]^M_{v[x\mapsto\underline{a}]} = \mathbf{1}) \\
8. & \Leftrightarrow & \text{for some } \underline{a} \in \underline{M}\ [\![A \to B]\!]^M_{v[x\mapsto\underline{a}]} = \mathbf{1} \\
9. & \Leftrightarrow & [\![\exists x(A \to B)]\!]^M_v = \mathbf{1} \\
10. & \Leftrightarrow & M \models_v \exists x(A \to B)
\end{array}$$

Again, the crucial equivalence of lines 6 and 7 follows from lemma 8.7 because x is not free in B. **QED** (9.3)

Remark.
Notice the change of quantifier in the last two equivalences in point 1 of lemma 9.3. The second one, $\exists A \to B \Leftrightarrow \forall x(A \to B)$, assuming that $x \notin \mathcal{V}(B)$, can be illustrated as follows. Let $R(x)$ stand for 'x raises his voice' and T for 'there will be trouble' (which has no free variables). The sentence "If somebody raises his voice there will be trouble" can be represented as

$$\exists xR(x) \to T. \tag{9.4}$$

The intention here is to say that *no matter who* raises his voice, the trouble will ensue. Thus, intuitively, it is equivalent to say: "If *anyone* raises his voice, there will be trouble." This latter sentence can be easier seen to correspond to

$$\forall x(R(x) \to T). \tag{9.5}$$

The first equivalence from point 1 is not so natural. We would like to read

$$\forall xR(x) \to T \tag{9.6}$$

as "If everybody raises his voice, there will be trouble." This form is equivalent to

$$\exists x(R(x) \to T) \tag{9.7}$$

but it is not entirely clear what sentence in natural language should now correspond to this formula. In fact, one is tempted to ignore the different scope of the quantifier in (9.7) and in (9.4) and read both the same way. This is, again, a remainder that one has to be careful with formalizing natural language expressions. For the future, let us keep in mind the important difference induced by the scope of quantifiers as the one between (9.4) and (9.7).

Theorem 9.8 [**Prenex NF**] Each formula B is equivalent to a formula B_P in PNF.

Proof. By induction on the complexity of B. The IH gives us a PNF for the subformulae and lemma 9.2 allows us to replace these subformulae by their PNF.

<u>B IS :</u>

ATOMIC :: Having no quantifiers, B is obviously in PNF.

$\neg A$ :: By IH, A has a PNF, and by lemma 9.2, $B \Leftrightarrow \neg A_P$. Using 2 of Lemma 9.3, we can move $\neg$ inside changing all the quantifiers. The result will be B_P.

$\exists x A$ :: Replacing A with A_P gives a PNF $B_P = \exists x A_P$.

$A \to C$ :: By IH and lemma 9.2, this is equivalent to $A_P \to C_P$. First, use 3 of lemma 9.3 to rename all bound variables in A_P so that they are distinct from all the variables (bound or free) in C_P. Then do the same with C_P. Use lemma 9.3.1 to move the quantifiers outside the whole implication. (Because of the renaming, no bound variable will at any stage occur freely in the other formula.) The result is B_P.

QED (9.8)

Example 9.9
We obtain PNF using the prenex operations:

$$\begin{array}{rcll}
\forall x \exists y A(x,y) \to \neg \exists x B(x) & \Leftrightarrow & \exists x(\exists y A(x,y) \to \neg \exists x B(x)) & 1 \\
& \Leftrightarrow & \exists x \forall y (A(x,y) \to \neg \exists x B(x)) & 1 \\
& \Leftrightarrow & \exists x \forall y (A(x,y) \to \forall x \neg B(x)) & 2 \\
& \Leftrightarrow & \exists x \forall y (A(x,y) \to \forall z \neg B(z)) & 3 \\
& \Leftrightarrow & \exists x \forall y \forall z (A(x,y) \to \neg B(z)) & 1
\end{array}$$

Formulae with abbreviated connectives may be first rewritten to the form with $\neg$ and $\rightarrow$ only, before applying the prenex transformations:

$$\begin{array}{rcll} \exists x A(x,y) \vee \forall y B(y) & \Leftrightarrow & \neg\exists x A(x,y) \rightarrow \forall y B(y) & \\ & \Leftrightarrow & \forall x \neg A(x,y) \rightarrow \forall y B(y) & 2 \\ & \Leftrightarrow & \exists x(\neg A(x,y) \rightarrow \forall y B(y)) & 1 \\ & \Leftrightarrow & \exists x(\neg A(x,y) \rightarrow \forall z B(z)) & 3 \\ & \Leftrightarrow & \exists x \forall z(\neg A(x,y) \rightarrow B(z)) & 1 \\ & \Leftrightarrow & \exists x \forall z(A(x,y) \vee B(z)) & \end{array}$$

Alternatively, we may use direct prenex operations which are derivable from those defined in 9.3:

$$\begin{array}{ll} (\mathsf{Q}xA \vee B) \Leftrightarrow \mathsf{Q}x(A \vee B) & \text{provided } x \notin \mathcal{V}(B) \\ (\mathsf{Q}xA \wedge B) \Leftrightarrow \mathsf{Q}x(A \wedge B) & \text{provided } x \notin \mathcal{V}(B) \end{array}$$

$\square$

Notice that PNF is not unique, since the order in which we apply the prenex operations may be chosen arbitrarily.

Example 9.10
Let B be $(\forall x\ x > 0) \rightarrow (\exists y\ y = 1)$. We can apply the prenex operations in two ways:

$$(\forall x\ x > 0) \rightarrow (\exists y\ y = 1)$$
$$\Leftrightarrow$$
$$\begin{array}{r|c} \exists x\ (x > 0 \rightarrow (\exists y\ y = 1)) & \exists y\ ((\forall x\ x > 0) \rightarrow y = 1) \Leftrightarrow \\ \Leftrightarrow \exists x\ \exists y\ (x > 0 \rightarrow y = 1) & \exists y\ \exists x\ (x > 0 \rightarrow y = 1) \end{array}$$

Obviously, since the order of the quantifiers of the same kind does not matter (exercise 7.2.1), the two resulting formulae are equivalent. However, the quantifiers may also be of different kinds:

$$(\exists x\ x > 0) \rightarrow (\exists y\ y = 1)$$
$$\Leftrightarrow$$
$$\begin{array}{r|c} \forall x\ (x > 0 \rightarrow (\exists y\ y = 1)) & \exists y\ ((\exists x\ x > 0) \rightarrow y = 1) \Leftrightarrow \\ \Leftrightarrow \forall x\ \exists y\ (x > 0 \rightarrow y = 1) & \exists y\ \forall x\ (x > 0 \rightarrow y = 1) \end{array}$$

Although it is not true *in general* that $\forall x \exists y A \Leftrightarrow \exists y \forall x A$, the equivalence preserving prenex operations ensure – due to renaming of bound variables which avoids name clashes with variables in other subformulae – that the results (like the two formulae above) are equivalent. $\square$

2: A few bits of Model Theory

Roughly and approximately, model theory studies the properties of model classes. Notice that a model class is not just an arbitrary collection K of

FOL-structures – it is a collection of *models of some set* Γ of formulae, i.e., such that $\mathsf{K} = Mod(\Gamma)$ for some Γ. The important point is that the syntactic form of the formulae in Γ may have a heavy influence on the properties of its model class (as we illustrate in theorem 9.15). On the other hand, knowing some properties of a given class of structures, model theory may sometimes tell what syntactic forms of axioms are necessary/sufficient for axiomatizing this class. In general, there exist non-axiomatizable classes K, i.e., such that for no FOL-theory Γ, one can get $\mathsf{K} = Mod(\Gamma)$.

2.1: Substructures

As an elementary example of the property of a class of structures we will consider (in 2.2) closure under substructures and superstructures. Here we only define these notions.

Definition 9.11 Let Σ be a FOL alphabet and let M and N be Σ-structures: N is a *substructure* of M (or M is a *superstructure* (or *extension*) of N), written $N \sqsubseteq M$, iff:

- $\underline{N} \subseteq \underline{M}$
- For all $a \in \mathcal{I} : [\![a]\!]^N = [\![a]\!]^M$
- For all $f \in \mathcal{F}$, and $\underline{a}_1, \ldots, \underline{a}_n \in \underline{N} : [\![f]\!]^N(\underline{a}_1, \ldots, \underline{a}_n) = [\![f]\!]^M(\underline{a}_1, \ldots, \underline{a}_n) \in \underline{N}$
- For all $R \in \mathcal{R}$, and $\underline{a}_1, \ldots, \underline{a}_n \in \underline{N} : \langle \underline{a}_1, \ldots, \underline{a}_n \rangle \in [\![R]\!]^N \Leftrightarrow \langle \underline{a}_1, \ldots, \underline{a}_n \rangle \in [\![R]\!]^M$

Let K be an arbitrary class of structures. We say that K is:

- closed under substructures if whenever $M \in \mathsf{K}$ and $N \sqsubseteq M$, then also $N \in \mathsf{K}$
- closed under superstructures if whenever $N \in \mathsf{K}$ and $N \sqsubseteq M$, then also $M \in \mathsf{K}$

Thus $N \sqsubseteq M$ iff N has a more restricted interpretation domain than M, but all constants, function and relation symbols are interpreted identically within this restricted domain. Obviously, every structure is its own substructure, $M \sqsubseteq M$. If $N \sqsubseteq M$ and $N \neq M$, which means that $\underline{N}$ is a *proper* subset of $\underline{M}$, then we say that N is a proper substructure of M.

Example 9.12
Let Σ contain one individual constant c and one binary function symbol $\odot$. The structure Z with $\underline{Z} = \mathbb{Z}$ being the integers, $[\![c]\!]^Z = 0$ and $[\![\odot]\!]^Z(x, y) = x + y$ is a Σ-structure. The structure N with $\underline{N} = \mathbb{N}$ being only the

natural numbers with zero, $[\![c]\!]^N = 0$ and $[\![\odot]\!]^N(x, y) = x + y$ is obviously a substructure $N \sqsubseteq Z$.

Restricting furthermore the domain to the even numbers, i.e. taking P with $\underline{P}$ being the even numbers greater or equal zero, $[\![c]\!]^P = 0$ and $[\![\odot]\!]^P(x, y) = x + y$ yields again a substructure $P \sqsubseteq N$.

The class $\mathsf{K} = \{Z, N, P\}$ is not closed under substructures. One can easily find other Σ-substructures not belonging to K (for instance, all negative numbers with zero and addition is a substructure of Z).

Notice that, in general, to obtain a substructure it is *not* enough to select an arbitrary subset of the underlying set. If we restrict N to the set $\{0, 1, 2, 3\}$ it will not yield a substructure of N – because, for instance, $[\![\odot]\!]^N(1, 3) = 4$ and this element is not in our set. Any structure, and hence a substructure in particular, must be "closed under all operations", i.e., applying any operation to elements of the (underlying set of the) structure must produce an element in the structure.

On the other hand, a subset of the underlying set may fail to be a substructure if the operations are interpreted in different way. Let M be like Z only that now we let $[\![\odot]\!]^M(x, y) = x - y$. Neither N nor P are substructures of M since, in general, for $x, y \in \underline{N}$ (or $\in \underline{P}$): $[\![\odot(x, y)]\!]^N = x + y \neq x - y = [\![\odot(x, y)]\!]^M$. Modifying N so that $[\![\odot]\!]^{N'}(x, y) = x - y$ does not yield a substructure of M either, because this does not define $[\![\odot]\!]^{N'}$ for $x > y$. No matter how we define this operation for such cases (for instance, to return 0), we won't obtain a substructure of M – the result will be different than in M. $\square$

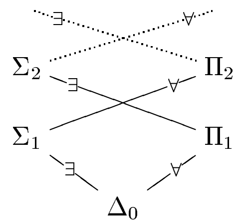
Remark 9.13

Given an FOL alphabet Σ, we may consider *all* Σ-structures, $\mathsf{Str}(\Sigma)$. Obviously, this class is closed under Σ-substructures. With the substructure relation, $\langle \mathsf{Str}(\Sigma), \sqsubseteq \rangle$ is a weak partial ordering: $\sqsubseteq$ is obviously reflexive (any structure is its own substructure), and also transitive (substructure X of a substructure of Y is itself a substructure of Y) and antisymmetric (if both $X \sqsubseteq Y$ and $Y \sqsubseteq X$ then $X = Y$). $\square$

2.2: Σ-Π CLASSIFICATION

A consequence of theorem 9.8 is that any axiomatizable class K, can be axiomatized by formulae in PNF. This fact has a model theoretic flavour, but the theory studies, in general, more specific phenomena. Since it is the relation between the classes of structures, on the one hand, and the syntactic form of formulae, on the other, one often introduces various syntactic classifications of formulae. We give here only one example.

The existence of PNF allows us to "measure the complexity" of formulae. Comparing the prefixes, we would say that $A_1 = \exists x \forall y \exists z B$ is "more complex" than $A_2 = \exists x \exists y \exists z B$. Roughly, a formula is the more complex, the more changes of quantifiers in its prefix.



Definition 9.14 A formula A is Δ_0 iff it has no quantifiers. It is:

- Σ_1 iff $A \Leftrightarrow \exists x_1 \ldots \exists x_n B$, where B is Δ_0.
- Π_1 iff $A \Leftrightarrow \forall x_1 \ldots \forall x_n B$, where B is Δ_0.
- Σ_{i+1} iff $A \Leftrightarrow \exists x_1 \ldots \exists x_n B$, where B is Π_i.
- Π_{i+1} iff $A \Leftrightarrow \forall x_1 \ldots \forall x_n B$, where B is Σ_i.

Since PNF is not unique, a formula can belong to several levels and we have to consider all possible PNFs for a formula in order to determine its complexity. Typically, saying that a formula is Σ_i, resp. Π_i, one means that this is the least such i.

A formula may be both Σ_i and Π_i – in Example 9.10 we saw (the second) formula equivalent to both $\forall x \exists y B$ and to $\exists y \forall x B$, i.e., one that is both Π_2 and Σ_2. Such formulae are called Δ_i.

We only consider the following (simple) example of a model theoretic result. Point 1 says that the validity of an existential formula is preserved when passing to the superstructures – the model class of existential sentences is closed under superstructures. Dually, 2 implies that model class of universal sentences is closed under substructures.

Theorem 9.15 Let A, B be closed formulae over some alphabet Σ, and assume A is Σ_1 and B is Π_1. Let M, N be Σ-structures and $N \sqsubseteq M$. If

(1) $N \models A$ then $M \models A$
(2) $M \models B$ then $N \models B$.

Proof. (1) A is closed Σ_1, i.e., it is (equivalent to) $\exists x_1 \ldots \exists x_n A'$ where A' has no quantifiers nor variables other that $x_1, \ldots, x_n$. If $N \models A$ then there exist $\underline{a}_1, \ldots, \underline{a}_n \in \underline{N}$ such that $N \models_{x_1 \mapsto \underline{a}_1, \ldots, x_n \mapsto \underline{a}_n} A'$. Since $N \sqsubseteq M$, we have $\underline{N} \subseteq \underline{M}$ and the interpretation of all symbols is the same in M as in N. Hence $M \models_{x_1 \mapsto \underline{a}_1, \ldots, x_n \mapsto \underline{a}_n} A'$, i.e., $M \models A$.

(2) This is a dual argument. Since $M \models \forall x_1 \ldots \forall x_n B'$, B' is true for

all elements of $\underline{M}$ and $\underline{N} \subseteq \underline{M}$, so B' will be true for all element of this subset as well. **QED** (9.15)

The theorem can be applied in at least two different ways which we illustrate in the following two examples. We consider only case 2., i.e., when the formulae of interest are Π_1 (universal).

Example 9.16 **[Constructing new structures for Π_1 axioms]**
First, given a set of Π_1 axioms and an arbitrary structure satisfying them, the theorem allows us to conclude that any substructure will also satisfy the axioms.

Let Σ contain only one binary relation symbol R. Recall definition 1.14 – a strict partial ordering is axiomatized by two formulae

(1) $\forall x \forall y \forall z : R(x,y) \wedge R(y,z) \to R(x,z)$ – transitivity, and
(2) $\forall x : \neg R(x,x)$ – irreflexivity.

Let N be an arbitrary strict partial ordering, i.e., an arbitrary Σ-structure satisfying these axioms. For instance, let $N = \langle \mathbb{N}, < \rangle$ be the natural numbers with less-than relation. Since both axioms are Π_1, the theorem tells us that any substructure of N, i.e., any subset of $\underline{N}$ with the interpretation of R restricted as in definition 9.11, will itself be a strict partial ordering. For instance, any subset $S \subseteq \mathbb{N}$ with the same less-than relation restricted to S is, by the theorem, a strict partial ordering. $\Box$

Example 9.17 **[Non-axiomatizability by Π_1 formulae]**
Second, given some class of structures, the theorem may be used to show that it is not Π_1-axiomatizable.

Let Σ be as in the previous example. Call a (strict partial) ordering "*dense*" if, in addition to the two axioms from the previous example, it also has the following property:

(3) whenever $R(x,y)$ then there exists a z such that $R(x,z)$ and $R(z,y)$.

For instance, the closed interval of all real numbers $M = \langle [0,1], < \rangle$ is a dense strict partial ordering. Now, remove all the numbers from the open interval $(0,1)$ – this leaves us with just two elements $\{0,1\}$ ordered $0 < 1$. This is a Σ substructure of M (Σ contains no ground terms, so any elements from the underlying set can be removed). But this is not a dense ordering! The class of dense orderings over Σ is not closed under substructures and so, by the theorem, it is not axiomatizable using only Π_1 formulae. □

3: "Syntactic" semantics and Computations

Model theory introduces also various classifications of structures. One may ask about the existence of finite models for a given Γ, about the existence of infinite models, uncountable models etc. One of such concepts, of central importance in computer science, is described below.

3.1: Reachable structures and Term structures

Definition 9.18 A Σ-structure T is *reachable* iff for each $\underline{a} \in \underline{T}$ there is a ground term $t \in \mathcal{GT}_\Sigma$ with $\underline{a} = [\![t]\!]^T$. A *reachable model* of a Σ-theory Γ is a Σ-reachable structure M such that $M \models \Gamma$.

Intuitively, a reachable structure for a Σ contains only elements which can be denoted (reached/pointed to) by some ground term.

Example 9.19
Let Σ contain only three constants a, b, c. Define a Σ-structure M by:

- $\underline{M} = \mathbb{N}$, and
- $[\![a]\!]^M = 0,\ [\![b]\!]^M = 1,\ [\![c]\!]^M = 2.$

M contains a lot of "junk" elements (all natural numbers greater than 2) which are not required to be there in order for M to be a Σ-structure – M is not a *reachable* Σ-structure. Define N by

- $\underline{N} = \{0, 1, 2, 3, 4\}$, and
- $[\![a]\!]^N = 0,\ [\![b]\!]^N = 1,\ [\![c]\!]^N = 2.$

Obviously, $N \sqsubseteq M$. Still N contains unreachable elements 3 and 4. Restricting $\underline{N}$ to $\{0, 1, 2\}$, we obtain yet another substructure $T \sqsubseteq N$. T is the only reachable structure of the three.
Yet another structure is given by:

- $\underline{S} = \{0, 1\}$, and
- $[\![a]\!]^S = 0,\ [\![b]\!]^S = 1,\ [\![c]\!]^S = 1.$

S is reachable too, but it is not a substructure of any previous one: although $\underline{S} \subset \underline{T}$, we have that $[\![c]\!]^S = 1 \neq 2 = [\![c]\!]^T$. $\square$

Proposition 9.20 If T is a reachable Σ-structure then it has no proper substructure.

Proof. The claim is that there is no Σ-structure M with $M \sqsubseteq T$ and $M \neq T$, i.e., such that $\underline{M} \subset \underline{T}$. Indeed, since each element $\underline{a} \in \underline{T}$ is the (unique) interpretation of some ground term $t \in \mathcal{GT}_\Sigma$, $\underline{a} = [\![t]\!]^T$, if we remove $\underline{a}$ from $\underline{T}$, t will have no interpretation in the resulting subset, which could coincide with its interpretation $[\![t]\!]^T$ in T. **QED** (9.20)

The proposition shows also a particular property of reachable structures with respect to the partial ordering $\langle \mathsf{Str}(\Sigma), \sqsubseteq \rangle$ defined in remark 9.13.

Corollary 9.21 A Σ-reachable structure T is a minimal element of $\langle \mathsf{Str}(\Sigma), \sqsubseteq \rangle$.

The opposite, however, need not be the case. If Σ contains no ground terms, the minimal Σ-structure, if they exist, will not be reachable.

Example 9.22
Let Σ contain only one binary function symbol $\oplus$. A Σ-structure M with $\underline{M} = \{\bullet\}$ and $[\![\oplus]\!]^M(\bullet, \bullet) = \bullet$ has no proper Σ-substructure. If such a structure N existed, it would require $\underline{N} = \varnothing$, but this is forbidden by the definition of Σ-structure (8.1 requires the underlying set of any structure to be non-empty). However, M is not Σ-reachable, since $\mathcal{GT}_\Sigma = \varnothing$. $\square$

Proposition 9.23 If Σ has at least 1 constant symbol, then any Σ-structure M has a reachable substructure.

Proof. Since $\mathcal{GT}_\Sigma \neq \varnothing$, we can take only the part of $\underline{M}$ consisting of the interpretations of ground terms, keeping the interpretation of relation and function symbols for these elements intact. **QED** (9.23)

By corollary 9.21, such a reachable substructure (of any M) will be minimal element of the partial ordering $\langle \mathsf{Str}(\Sigma), \sqsubseteq \rangle$. One could feel tempted to conclude that this shows that this ordering is well-founded but this is not the case as the following example shows.

Example 9.24
Let Σ contain one constant symbol $\odot$ and let N be a Σ-structure with

$\underline{N} = \mathbb{N} = \{0, 1, 2, 3...\}$ and $[\![\odot]\!]^N = 0$. The structure N_0 with $\underline{N_0} = \{0\}$ is the reachable (and hence minimal) substructure of N. However, we may also form the following chain of substructures: let N_i be given by the underlying set $\underline{N_i} = \mathbb{N}\setminus\{1, 2...i\}$ for $i > 0$, and $[\![\odot]\!]^{N_i} = 0 = [\![\odot]\!]^N$. We thus have that $N \sqsupseteq N_1 \sqsupseteq N_2 \sqsupseteq N_3 \sqsupseteq ...$, i.e., we obtain an infinite descending chain of substructures. Hence the relation $\sqsubseteq$ is not well-founded (even if we restrict it to the set of substructures of N). □

It follows from proposition 9.23 that for any Σ with at least one constant symbol there is a Σ-reachable structure. A special type of reachable structure is of particular interest, namely the *term structure.* In a term structure over Σ, the domain of interpretation is the set of ground terms over Σ, and moreover every term is interpreted as itself:

Definition 9.25 Let Σ be an alphabet with at least one constant. A *term structure* T_Σ over Σ has the following properties:

- The domain of interpretation is the set of all ground Σ-terms, i.e., $\underline{T_\Sigma} = \mathcal{GT}_\Sigma$.
- For each constant symbol $a \in \mathcal{I}$, we let a be its own interpretation:
$$[\![a]\!]^{T_\Sigma} = a$$
- For each function symbol $f \in \mathcal{F}$ of arity n, and terms $t_1, \ldots, t_n$, we let
$$[\![f]\!]^{T_\Sigma}(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$$

This may look a bit strange at first but is a perfectly legal specification of (part of) a structure according to definition 8.1 which requires an arbitrary non-empty interpretation domain. The only specical thing is that we, in a sense, look at terms from two different perspectives. On the one hand, as terms – syntactic objects – to be interpreted, i.e., to be assigned meaning by the operation $[\![\_]\!]^M$. Taking M to be T_Σ, we now find the same terms as the elements – semantic objects – interpreting the syntactic terms, i.e., $[\![t]\!]^{T_\Sigma} = t$.

Such structures are interesting because they provide mechanic means of *constructing* a structure, i.e., a semantic object, from the mere syntax defined by the alphabet.

Example 9.26
Let Σ contain only two constant symbols a, b. The term structure T_Σ will be given by: $\underline{T_\Sigma} = \{a, b\}$, $[\![a]\!]^{T_\Sigma} = a$, $[\![b]\!]^{T_\Sigma} = b$. (If this still looks confusing, you may think of T_Σ with all symbols underlined, i.e. $\underline{T_\Sigma} = \{\underline{a}, \underline{b}\}$, $[\![a]\!]^{T_\Sigma} = \underline{a}$, $[\![b]\!]^{T_\Sigma} = \underline{b}$.)

Now extend Σ with one unary function symbol s. The corresponding term structure T_Σ will be now:

$$\begin{aligned} \underline{T_\Sigma} &= \{\ a,\ s(a),\ s(s(a)),\ s(s(s(a))),\ \ldots \\ &\qquad b,\ s(b),\ s(s(b)),\ s(s(s(b))),\ \ldots\ \} \\ [\![a]\!]^{T_\Sigma} &= a \\ [\![b]\!]^{T_\Sigma} &= b \\ [\![s]\!]^{T_\Sigma}(x) &= s(x) \ \text{ for all } x \in \underline{T_\Sigma} \end{aligned}$$

□

Thus every term structure is reachable. Note that nothing is said in definition 9.25 about the interpretation of relation symbols. Thus a term structure for a given Σ is not a full FOL-structure. However, for every Σ with at least one constant there is at least one FOL term structure over Σ, for instance the one where each relation symbol $R \in \mathcal{R}$ is interpreted as the empty set:

$$[\![R]\!]^{T_\Sigma} = \varnothing$$

Such a structure is, most probably, of little interest. Typically, one is interested in obtaining a *term model*, i.e., to endow the term structure T_Σ with the interpretation of predicate symbols in such a way that one obtains a model for some given theory.

Remark 9.27 **[Term models, reachable models and other models]**
Let Σ be given by $\mathcal{I} = \{p, q\}$, $\mathcal{F}_1 = \{f\}$ and $\mathcal{R}_1 = \{P, Q\}$. The term structure is then

$$\begin{aligned} T_\Sigma = \{&p,\ f(p),\ f(f(p)),\ f(f(f(p))),\ \ldots \\ &q,\ f(q),\ f(f(q)),\ f(f(f(q))),\ \ldots\ \} \end{aligned}$$

Now, consider a theory $\Gamma = \{P(p), Q(q), \forall x(P(x) \rightarrow Q(f(x))), \forall x(Q(x) \rightarrow P(f(x)))\}$. We may turn T_Σ into a (reachable) model of Γ, so called *term model* T_Γ with $\underline{T_\Gamma} = T_\Sigma$, by letting

- $[\![P]\!]^{T_\Gamma} = \{f^{2n}(p) : n \geq 0\} \cup \{f^{2n+1}(q) : n \geq 0\}$ and
- $[\![Q]\!]^{T_\Gamma} = \{f^{2n+1}(p) : n \geq 0\} \cup \{f^{2n}(q) : n \geq 0\}$.

It is easy to verify that, indeed, $T_\Gamma \models \Gamma$. We show that

$$T_\Gamma \models \forall x(P(x) \vee Q(x)). \tag{9.28}$$

We have to show that $[\![P(x) \vee Q(x)]\!]_v^{T_\Gamma} = \mathbf{1}$ for all assignments $v : \{x\} \rightarrow \underline{T_\Gamma}$. But all such assignments assign a ground term to x, that is, we have to show that $T_\Gamma \models P(t) \vee Q(t)$ for all ground terms $t \in \mathcal{GT}_\Sigma$. We show this by induction on the complexity of ground terms.

$t \in \mathcal{I}$ :: Since $T_\Gamma \models \Gamma$ we have $T_\Gamma \models P(p)$ and hence $T_\Gamma \models P(p) \vee Q(p)$. In the same way, $T_\Gamma \models Q(q)$ gives that $T_\Gamma \models P(q) \vee Q(q)$.

$f(t)$ :: By IH, we have $T_\Gamma \models P(t) \vee Q(t)$, i.e., either $T_\Gamma \models P(t)$ or $T_\Gamma \models Q(t)$. But also $T_\Gamma \models \Gamma$, so, in the first case, we obtain that $T_\Gamma \models Q(f(t))$, while in the second $T_\Gamma \models P(f(t))$. Hence $T_\Gamma \models P(f(t)) \vee Q(f(t))$.

Thus the claim (9.28) is proved. As a matter of fact, we have proved more than that. Inspecting the proof, we can see that the only assumption we have used was that $T_\Gamma \models \Gamma$. On the other hand, the inductive proof on $\mathcal{GT}_\Sigma$ was possible because any assignment $v : \{x\} \to \underline{T_\Gamma}$ assigned to x an interpretation of some ground term, i.e., $v(x) = [\![t]\!]^{T_\Gamma}$ for some $t \in \mathcal{GT}_\Sigma$. In other words, the only assumptions were that T_Γ was a reachable model of Γ, and what we have proved is:

$$\text{for any reachable } T : T \models \Gamma \Rightarrow T \models \forall x(P(x) \vee Q(x)). \tag{9.29}$$

It is typical that proofs by induction on ground terms like the one above, show us such more general statement like (9.29) and not merely (9.28).

The point now is that the qualification "reachable" is essential and cannot be dropped – it is not the case that $\Gamma \models \forall x(P(x) \vee Q(x))$! Consider a structure M which is exactly like T_Γ but has one additional element, i.e., $\underline{M} = T_\Sigma \cup \{*\}$, such that $* \notin [\![P]\!]^M$ and $* \notin [\![Q]\!]^M$ and $[\![f]\!]^M(*) = *$. M is still a model of Γ but not a reachable one due to the presence of $*$. We also see that $M \not\models \forall x(P(x) \vee Q(x))$ since $[\![P(x)]\!]^M_{x\mapsto *} = \mathbf{0}$ and $[\![Q(x)]\!]^M_{x\mapsto *} = \mathbf{0}$. In short, the proof that something holds for all ground terms, shows that the statement holds for all reachable models but, typically, not that it holds for arbitrary models.

Example 9.30

Notice that although a reachable structure always exists (when $\mathcal{GT}_\Sigma \neq \varnothing$), there may be no reachable *model* for some Σ-theory Γ. Let Σ contain only two constants a, b and one predicate R. T_Σ is given by $\{a, b\}$ and $[\![R]\!]^{T_\Sigma} = \varnothing$. Let Γ be $\{\neg R(a), \neg R(b), \exists x R(x)\}$. It is impossible to construct a model for Γ – i.e. interpret $[\![R]\!]$ so that all formulae in Γ are satisfied – which is reachable over Σ.

But let us extend the alphabet to Σ' with a new constant c. Let T' be $T_{\Sigma'}$ but interpret R as $[\![R]\!]^{T'} = \{c\}$. Then, obviously, $T' \models \Gamma$. T' is not a Σ-structure since it contains the interpretation of c which is not in Σ. To turn T' into a Σ-structure satisfying Γ we only have to "forget" the interpretation of c. The resulting structure T is identical to T', except that T is a Σ-structure and the element c of $\underline{T} = \underline{T'}$ has no corresponding term in the alphabet. Thus:

- T' is a reachable Σ'-structure but it is not a Σ-structure
- T' is a Σ'-reachable model of Γ
- T is a Σ-structure but not a Σ'-structure

- T is a Σ-structure but not a Σ-reachable structure
- T is a model of Γ but it is not a Σ-reachable model

□

As an important corollary of Theorem 9.15, we obtain the following sufficient conditions for the existence of reachable models.

Corollary 9.31 Let Γ be a collection of Π_1 formulae, over an alphabet Σ with $\mathcal{GT}_\Sigma \neq \varnothing$. If $Mod(\Gamma) \neq \varnothing$ then Γ has a reachable model.

Simply, the existence of some model of Γ allows us, by Proposition 9.23, to restrict it to its reachable substructure. By Theorem 9.15, this substructure is also a model of Γ.

Term structures are used primarily as the basis for constructing the domain of interpretation – namely, the ground terms – for some reachable structure, which can, sometimes, be obtained from T_Σ by imposing appropriate interpretation of the relation symbols. Such use is very common in theoretical computer science for constructing cannonical models for specifications (theories). This will also be the use we will make of it in the proof of completeness of $\mathcal{N}$ for FOL in the next chapter.

"Syntactic" models have typically an important property of being canonical representatives of the whole model class. When model class comes equipped with the mappings between models (some form of homomorphisms) forming a category, the syntactically constructed models happen typically to be initial ones. We won't describe this concept in detail here and take it simply as a vague synonym for being syntactically generated. In addition to that, they have a close relationship to the possibility of carrying out mechanic computations – they give often some possibility of defining a computational strategy.

3.2: Herbrand's theorem

In exercise 11.6 we state the Herbrand theorem which is intimately related to the so called Herbrand models. These models are, in fact, nothing else but the term models we encounter in the proof of completeness for FOL. But to avoid the need of "saturating" the theory, one makes additional restrictions.

> Suppose Σ contains at least one constant, and let Γ be a set of quantifier free Σ-formulae. Then: $\Gamma \vdash_{\mathcal{N}}^{\mathsf{FOL}} \bot \iff GI(\Gamma) \vdash_{\mathcal{N}}^{\mathsf{PL}} \bot$.

Requirement on the formulae in Γ to be quantifier-free amounts to their universal closure and is crucial for the reduction of FOL-inconsistency to PL-inconsistency in the theorem. $GI(\Gamma)$ is the set of all ground instances of the formulae in Γ which, by the requirement on the alphabet Σ, is guaranteed to be non-empty. The implication $GI(\Gamma) \vdash_{\mathcal{N}}^{\mathsf{PL}} \bot \Rightarrow \Gamma \vdash_{\mathcal{N}}^{\mathsf{FOL}} \bot$ holds, of course, always, since $GI(\Gamma)$ is, in fact, a weaker theory than Γ – it axiomatizes *only* ground instances which are also consequences of Γ. Consider, for instance, language with one constant symbol $\mathbf{a}$ and $\Gamma = \{\forall x.P(x)\}$. Then $GI(\Gamma) = \{P(\mathbf{a})\}$. Obviously, $GI(\Gamma) = P(\mathbf{a}) \not\vdash_{\mathcal{N}}^{\mathsf{FOL}} \forall x.P(x)$ which, however, is trivially a provable consequence of Γ itself.

The importance of Herbrand's theorem lies in the indentification of the form of Γ's allowing also the opposite implication, namely, $\Gamma \vdash_{\mathcal{N}}^{\mathsf{FOL}} \bot \Rightarrow GI(\Gamma) \vdash_{\mathcal{N}}^{\mathsf{PL}} \bot$. It amounts to a kind of reduction of the FOL-theory Γ to all its "syntactically generated" instances. Using this reduction, one can attempt to check whether $\Gamma \vdash_{\mathcal{N}}^{\mathsf{FOL}} \phi$ by reducing *ad absurdum* – in PL!!! – the assumption $GI(\Gamma, \neg\phi)$. Such a proof strategy is refered to as "refutational proof" or "proof by contradiction" and proceeds as follows.

Assume that ϕ is closed and quantifier free (i.e., ground). The theorem says that then $\Gamma, \neg\phi \vdash_{\mathcal{N}}^{\mathsf{FOL}} \bot \Leftrightarrow GI(\Gamma), \neg\phi \vdash_{\mathcal{N}}^{\mathsf{PL}} \bot$. Thus, we have to check if some ground instances of (some formulae from) Γ, together with $\neg\phi$ lead to a contradiction. This is not yet any effective algorithm but we can imagine that such a checking of some formulae yielding an PL-contradiction can be, at least in principle and at least in some cases, performed. Having succeeded, i.e., showing $GI(\Gamma), \neg\phi \vdash_{\mathcal{N}}^{\mathsf{PL}} \bot$, we obtain $\Gamma, \neg\phi \vdash_{\mathcal{N}}^{\mathsf{FOL}} \bot$ which implies $\Gamma \vdash_{\mathcal{N}}^{\mathsf{FOL}} \neg\phi \to \bot$ and this, again, $\Gamma \vdash_{\mathcal{N}}^{\mathsf{FOL}} \phi$.

(9.32)

The procedure is slightly generalized when ϕ contains variables (which are then interpreted also in a specific way). Various computational mechanisms utilizing this principle will thus restrict their theories to quantifier free (i.e., universally quantified, according to exercise 8.9) formulae and alphabets with non-empty set of ground terms. In the following, we will see a particular – and quite central – example.

3.3: Horn clauses and logic programming

An important issue with the utilization of Herbrand theorem concerns the actual strategy to determine if $GI(\Gamma) \vdash \bot$. Various choices are possible and we suggest only a typical restriction leading to one such possibility (which, in fact, does not even bother to follow the strategy (9.32) of proof by contradiction, but derives the consequences of a theory directly).

A clause is a formula of the form $L_1 \vee ... \vee L_n$ where each L_i is a literal – a positive or negated atom (i.e., formula of the form $P(\overline{t})$ for a sequence of some (not necessarily ground) terms $\overline{t}$.) By the general fact that $M \models A \Leftrightarrow M \models \forall(A)$, one does not write the universal quantifiers which are present implicitly. A Horn clause is a clause having exactly one positive literal, i.e., $\neg A_1 \vee ... \vee \neg A_n \vee A$, which therefore can also be writtem as

$$A_1 \wedge A_2 \wedge ... \wedge A_n \to A \tag{9.33}$$

where all A_i's are positive atoms. The particular case of a Horn clause with $n = 0$ is called a "fact". The conjunction of the assumptions is called the "body" and the conclusion the "head" of the clause. (This terminology is used only in the context of logic programming.)

MP and the chaining rule 7.24.1 can be generalized to the following rule operating on and yielding only Horn clauses:

$$\frac{\Gamma \vdash_{\mathcal{N}} A_1 \wedge ... \wedge A_k \to \mathbf{B_i} \quad ; \quad \Gamma \vdash_{\mathcal{N}} B_1 \wedge ... \wedge \mathbf{B_i} \wedge ... \wedge B_n \to C}{\Gamma \vdash_{\mathcal{N}} B_1 \wedge ... \wedge (\mathbf{A_1} \wedge ... \wedge \mathbf{A_k}) \wedge ... \wedge B_n \to C} \tag{9.34}$$

(This is a special case of the general resolution rule which "joins" two clauses removing from each a single literal – a variable occurring positively in one clause and negatively in the other.) In particular, when B_i is a fact, it can be simply removed from the body. When all atoms from the body of a clause get thus removed, its conclusion becomes a new fact.

Suppose, we have the following theory, Γ:

$$\begin{array}{lr}
1. & Parent(Ben, Ada) \\
2. & Parent(Cyril, Ben) \\
3. & Parent(Cecilie, Ben) \\
4. & Parent(David, Cyril) \\
5. & Ancestor(Eve, x) \\
6. & Parent(y, x) \to Ancestor(y, x) \\
7. & Ancestor(z, y) \wedge Ancestor(y, x) \to Ancestor(z, x)
\end{array} \tag{9.35}$$

The questions on the left have then the answers on the right, and you should have no problems with convincing yourself about that:

Does $\Gamma \vdash$..				
? $Ancestor(Eve, Ada)$	:	Yes	1.	
? $Parent(David, Ada)$	:	No	2.	
? $Ancestor(David, Ada)$	:	Yes	3.	(9.36)
? $Ancestor(Herod, Ben)$	:	No	4.	
? $Ancestor(Cyril, x)$	:	Ben, Ada	5.	
? $Ancestor(x, Ben)$	:	$Cecilie, Cyril, David, Eve$	6.	

The language of Horn clauses has some important properties. On the one hand, it is the most general sublanguage of FOL which guarantees the existence of initial models. These are just the Herbrand models obtained by collecting all positive ground atoms.

In more detail, we know that any universal theory (Π_1) has model class closed under substructures (theorem 9.15). If the language has some ground terms then, given any model, we can remove all "junk" elements and obtain a reachable model. Herbrand model is a term model, i.e., one which does not identfy any distinct terms but, in fact, interprets a ground term t as itself. In the above example, a possible model would identify all the persons and made both $Parent$ and $Ancestor$ reflexive (and consequently also transitive and symmetric). This would be a reachable model but not the intended one. Herbrand model will be the one we actually intended writing the program.

To construct it, we start with the (ground instances of) all facts and iterate the process of resolving the assumptions of conditional clauses to add more facts (essentially, by applying the rule (9.34).) Given a Horn clause theory Γ, we define:

HU_Γ : the Herbrand universe = the set of all ground terms. In the example (abbreviating the names by their first initials): $\{\mathbf{a}, \mathbf{b}, \mathbf{c}_1, \mathbf{c}_2, \mathbf{d}, \mathbf{e}\}$

HB_Γ : the Herbrand base = the set of all ground atoms. The model will be obtained as a subset of this set, namely, all ground atoms which must be true.

H_Γ : The construction of the Herbrand model proceeds inductively as follows. Write $\Gamma = (\mathcal{F}, \mathcal{C})$ as the pair of facts and clauses:

(1) $H_0 = GI(\mathcal{F})$ – all ground instances of all facts;

(2) $H_{i+1} = H_i \cup \{\theta(C) : A_1 \wedge ... \wedge A_n \to C \in \mathcal{C}\ \&\ \theta(A_1), ..., \theta(A_n) \in H_i\}$ – with θ ranging over all ground substitutions (to the assumed given and fixed set of all variables $X \to HU_\Gamma$);

(3) $H_\Gamma = \bigcup_{i<\omega} H_i$.

In the above example, we would obtain the model:

- $HU_\Gamma = \{\mathbf{a}, \mathbf{b}, \mathbf{c}_1, \mathbf{c}_2, \mathbf{d}, \mathbf{e}\}$
- H_Γ : $Parent = \{\langle \mathbf{b}, \mathbf{a}\rangle, \langle \mathbf{c}_1, \mathbf{b}\rangle, \langle \mathbf{c}_2, \mathbf{b}\rangle, \langle \mathbf{d}, \mathbf{c}_1\rangle\}$ and $Ancestor = \{\langle \mathbf{e}, x\rangle : x \in HU_\Gamma\} \cup Parent^+$ (i.e., transitive closure of $Parent$)

It is trivial to see that $H_\Gamma \models \Gamma$. Assignments to HU_Γ amount actually to ground substitutions so, for all facts, this holds by point 1. For a clause $A_1 ... A_n \to C$ and an assignment θ, assume that $H_\Gamma \models_\theta A_1 \wedge ... \wedge A_n$. By construction and point 3. this means that, at some step i, we obtained $\theta(A_k) \in H_i$ for all $1 \leq k \leq n$. But then, by point 2., $\theta(C)$ is also included in H_Γ at the step H_{i+1}.

In fact, we have that

$$H_\Gamma = \{A \in HB_\Gamma : \Gamma \models A\}. \tag{9.37}$$

This is the property of *minimality* – H_Γ does not satisfy more atoms than those which are satisfied in *every* model of the theory.

Computing with Horn Clauses

The other crucial property of Horn clauses is the possibility of operational interpretation of $\to$, according to which $A_1 \wedge ... \wedge A_n \to A$ means that, in order to establish/obtain A, one has to establish $A_1, ..., A_n$. This trivial chaining mechanism must be coupled with the treatement of variables. For instance, to establish $Ancestor(e, a)$ one uses the given fact $Ancestor(e, x)$ which, however, requires unification of two terms: x and a. Here, it is trivial since it amounts to a simple substitution. In general, unification may be more invovled. For instance, to unify $f(x, g(h, x))$ and $f(d(z), y)$ requires finding the substitutions $x \mapsto d(z)$ and $y \mapsto g(h, d(z))$, after which the two terms become equal to $f(d(z), g(h, d(z)))$.

The query $Ancestor(David, Ada)$ is now processed as follows:

(9.35)	goal	justification	unification
	$?Ancestor(\mathbf{d}, \mathbf{a})$	$\sim Ancestor(\mathbf{e}, x)$	fails : $\mathbf{e} \neq \mathbf{d}$
6.		$\leftarrow Parent(\mathbf{d}, \mathbf{a})$	no such fact
7.		$\leftarrow Ancestor(\mathbf{d}, y) \wedge Ancestor(y, \mathbf{a})$	?
	the search starts for y satisfying both literals in the body		
	$?Ancestor(\mathbf{d}, y)$	$\sim Ancestor(\mathbf{e}, x)$	fails : $\mathbf{e} \neq \mathbf{d}$
6.		$\leftarrow Parent(\mathbf{d}, \mathbf{c}_1)$	$y = \mathbf{c}_1$
	$?Ancestor(\mathbf{c}_1, \mathbf{a})$	$\sim Ancestor(\mathbf{e}, x)$	fails : $\mathbf{e} \neq \mathbf{c}_1$
6.		$\leftarrow Parent(\mathbf{c}_1, \mathbf{a})$	no such fact
7.		$\leftarrow Ancestor(\mathbf{c}_1, z) \wedge Ancestor(z, \mathbf{a})$	?
	$\vdots$	$\vdots$	$z = \mathbf{b}$
	YES		

Thus, we can actually *compute* the facts provable in Horn theories by means of the above mechanism based on the resolution rule (9.34) and unification algorithm. Observe the kind of "backward" process of computing: one starts with the query and performs a "backward chaining" along the available clauses until one manages to resolve all the assumptions by matching them against available facts.

In more detail, we use the (Horn clause) language defined as follows:

$T(erms) := C(onstants) \mid V(ariables) \mid F(T...T)$
C, V and Function symbols (and their arities) depend on the context/program; and so do the Predicate symbols:

$$\begin{aligned}
S(implegoal) &:= P(T...T)\\
G(oals) &:= S \mid S \land G\\
C(lause) &:= G \to S \mid S\\
P(rogram) &:= C^*\\
Q(uery) &:= ?S
\end{aligned}$$

One implements then the rule (9.34) by the following algorithm. Given Horn clauses: $A_1 \land ... \land A_n \to A$ and $B_1 \land ... \land B_m \to B$, when trying to establish A, we will have to resolve all A_i. We attempt to replace them by facts, and if this fails, by B_i's (until we arrive at facts):

(1) select an atom A_i and
(2) try to unify it with B (see further down for unification)
(3) if unification succeeded, replace A_i by $B_1 \land ... \land B_m$
(4) apply to the resulting clause the unifying substitution from 2.

If unification in 2. fails, try the next A_{i+1}. If none can be unified with B, try other clauses.

Unification of two atoms $P(t_1...t_n)$ and $R(s_1...s_n)$ requires first that $P \equiv R$ are syntactically identical (the same relation symbol) and then it amounts to finding a *unifier*, namely, a substitution θ such that for each $i : \theta(t_i) = \theta(s_i)$. If two terms have a unifier they also have a *most general unifier*, mgu. For instance, a possible unifier of $t_1 = f(x, g(a, x))$ and $t_2 = f(d(z), y)$ is $\alpha = \{x \mapsto d(d(a)), y \mapsto g(a, d(d(a))), z \mapsto d(a)\}$, which yields the term $\alpha(t_1) = \alpha(t_2) = f(d(d(a)), g(a, d(d(a))))$. However, the unifier $\theta = \{x \mapsto d(z), y \mapsto g(a, d(z))\}$ is more general – it yields the term $\theta(t_1) = \theta(t_2) = f(d(z), g(a, d(z)))$, from which $\alpha(t_1)$ can be obtained by further substitution $\beta = \{z \mapsto d(a)\}$. The most general unifier of terms t_i is a substitution θ such that for any other unifier α, there is a substitution β such that $\alpha(t_i) = \beta(\theta(t_i))$.

The following algorithm finds the most general unifier solving a set of equations $\{s_1 = t_1...s_n = t_n\}$ or reports the nonexistence of any unifier. It chooses repeatedly and nondeterministically one of the equations in the set and, performing the associated action, transforms the set (or halts). $\equiv$ stands here for the *syntactic* identity.

	if $s_i = t_i$ has the form		then
1.	$f(s'_1...s'_k) = f(t'_1...t'_k)$		replace it by the k equations $s'_i = t'_i$
2.	$f(s'_1...s'_k) = g(t'_1...t'_l)$	and $f \not\equiv g$	terminate with **failure**
3.	$x = x$		delete it
4.	$t = x$	and $t \notin \mathcal{V}$	replace it with $x = t$
5.	$x = t$	and $t \not\equiv x$ and x occurs in the rest of equations	if $x \in \mathcal{V}(t)$ – terminate with **failure** – otherwise, perform the substitution $x \mapsto t$ in all other equations

On successful termination, the set has the form $\{x_1 = r_1 ... x_m = r_m\}$, where x_i's are distinct variables from the initial terms and r_i's determine the substitution to be performed. (Variable x not occurring in this solution set is substituted by itself.)

Let us write $\Gamma \rightsquigarrow A$ iff the ground atom A can be obtained from a set of Horn clauses Γ using the above strategy. The following equivalences express then the soundness and completeness of the strategy with respect to the least Herbrand model as well as the whole model class (since the last two are equivalent for ground atoms by (9.37)):

$$\begin{aligned} \Gamma \rightsquigarrow A &\iff H_\Gamma \models A \\ &\iff \quad \Gamma \models A \end{aligned} \tag{9.38}$$

Thus, we can say that our computational strategy is just a way of checking if some fact holds in the least Herbrand model, H_Γ, of a given Horn clause theory Γ. Notice, however, that the equivalences hold here only for the ground atoms (they hold a bit more generally, but it does not affect our point here). We do have that $H_\Gamma \models A \Rightarrow \Gamma \rightsquigarrow A$ which says that every valid atom will be generated by the strategy. Conversely, we have also that $H_\Gamma \not\models A \Rightarrow \Gamma \not\rightsquigarrow A$. This, however, says only that if an atom is not satisfied, the strategy will not derive it. But there is no way to ensure derivability in our strategy of $\neg A$, since such literals are not part of the Horn clause language for which our results obtain.

COMPUTATIONAL COMPLETENESS

What is perhaps more surprising than the operational interpretation sketched above, is that we can thus compute *everything* which can be computed on a Turing machine:

The mechanism of unification and resolution of Horn clauses is Turing complete. (9.39)

Probably the simplest proof of this fact shows that register machine programs, rmp's, can be simulated by Horn clause programs. We have to take for granted the result stating that the rmp's as described below are computationally equivalent to Turing machines.

An rmp for a register machine over m registers $x_1 ... x_m$ is a sequence $I_1 ... I_n$ of n numbered instructions, each in one of the two forms:

(inc) $x_i := x_i + 1$ – increment register x_i;
(cnd) **if** $x_i \neq 0$ **then** $x_i := x_i - 1$ **and goto** j – conditional decrement and jump.

If, on reaching the instruction of the second form, $x_i = 0$, the program simply proceeds to the next instruction. The program terminates on reaching the **halt** instruction, always implicitly present as the I_{n+1}-th instruction. Such an rmp is said to compute a (partial) function $f : \mathbb{N}^l \to \mathbb{N}$, $l \leq m$ if $\forall n_1...n_l \in \mathbb{N}$ the execution starting with the register values $n_1...n_l, 0...0_m$ (the additional registeres $x_{l+1}...x_m$ which are not input are initialized to 0), terminates with $x_1 = f(n_1...n_l)$, whenever $f(n_1...n_l)$ is defined, and does not terminate otherwise.

Such an rmp is simulated by a Horn clause program P as follows. For each instruction I_k, $1 \leq k \leq n+1$, we have a predicate symbol $P_k(x_1...x_m, y)$ – the x_i's corresponding to the registeres and y to the result of the computation. Each I_k is either (inc) or (cnd) and these are simulated, respectively, by:

$$\text{(inc)}\ P_{k+1}(x_1...s(x_i)...x_m, y) \to P_k(x_1...x_i...x_m, y)$$

$$\begin{array}{ll} \text{(cnd)} & P_{k+1}(x_1...0...x_m, y) \to P_k(x_1...0...x_m, y) \\ & P_j(x_1...x_i...x_m, y) \to P_k(x_1...s(x_i)...x_m, y) \end{array} \qquad (I'_k)$$

In addition, we also have the **halt** instruction tranferring x_1 to the result postion:

$$\text{(hlt)} \qquad P_{n+1}(x_1...x_m, x_1) \qquad (I'_{n+1})$$

The query $P_1(n_1...n_l, 0...0_m, y)$ will result in the computation simulating step by step the execution of the corresponding rmp.

As a very simple corollary of the above (fact 9.39), we obtain:

First order logic is undecidable.

(9.40)

How does it follow? We have to take for granted yet another fact, namely, that halting problem for the rmp's (and hence also for Horn clause programs) is, just as it is for Turing machines, undecidable. (It should not be all too difficult to imagine that all these halting problems are, in fact, one and the same problem.) But halting of an rmp is equivalent to the existence of a result for the initial query, i.e., to the truth of the formula $\exists y P_1(n_1...n_l, 0...0_m, y)$ under the assumptions gathering all the clauses of the program P, in short, to the entailment

$$\forall(I'_1), ..., \forall(I'_n), \forall(I'_{n+1}) \models \exists y P_1(n_1...n_l, 0...0_m, y). \qquad (9.41)$$

If this entailment could be decided, we could decide the problem if our rmp's halt or not. But such a procedure does not exist by the undecidability of the halting problem for Turing machines (if only we have accepted the equivalence of the two problems).

PROLOG

The fact (9.39), and the preceding computation strategy, underlie Prolog – the programming language in which programs are sets of Horn clauses. It enables one asking queries about single facts but also about the possible instances making up facts, like the queries 6. or 5. in (9.36), about all x such that $Ancestor(Cyril, x)$. There are, of course, various operational details of not quite logical character which, in some cases, yield unexpected and even unsound results. They have mainly to do with the treatement of negation.

Wondering about the computational power of such a simple mechanism, one should ask oneself, considering the fact (9.39): where does the undecidability enter the stage here? The answer – suggested already in discussion after (9.38) – is: with the treatement of negation. Obviously, any positive atom following from a given set of clauses (and facts) can be computed in a finite time. In the above example of family relations the universe of terms was finite so, at least in principle, one can terminate the search for matching pairs of ancestors with the answer 'no' to the query $?Ancestor(Ada, Cyril)$. But in general, in the presence of function symbols, this universe is potentially infinite. Negation turns thus out to be tricky issue: one simply does not have a general rule for terminating a prolonged and unsuccessful search for matching substitutions – which would guarantee that the answer 'no' is always correct. (There is an extensive literature on the (im)possible treatement of this issue in Prolog.)

Exercises 9.

EXERCISE 9.1 Give an example contradicting the opposite implication to the one from Lemma 7.25.1, i.e., show that $\not\models A \to \forall x A$.

EXERCISE 9.2 Find PNFs for the following formulae

(1) $\forall x(f(g(x)) = x) \to \forall x \exists y(f(y) = x)$
(2) $\exists z \forall x A(x, z) \to \forall x \exists z A(x, z)$
(3) $\forall x \exists y(x + y = 0) \ \wedge \ \forall x \forall y(x + y = 0 \to y + x = 0)$

Since the matrix of a formula in PNF contains no quantifiers, it is often useful to assume that it is in DNF (or CNF). This can be obtained by the same manipulations as for PL. Transform the matrix of the result of the point 3 into DNF.

EXERCISE 9.3 Verify that the remaining equivalences claimed in Lemma 9.3 do hold.

EXERCISE 9.4 Show that $\langle \mathsf{Str}(\Sigma), \sqsubseteq \rangle$ is a weak partial ordering as claimed in remark 9.13.

<u>EXERCISE 9.5</u> Show that all structures of example 8.3, except 4., are reachable.

<u>EXERCISE 9.6</u> Explain why, in point 1. of theorem 9.15 it is necessary to assume that A is closed. Let, for instance, $A = \exists x R(x,y)$. Is it a Σ_1 formula? Could the statement be proved for this A? [Recall fact 8.27!]

optional

<u>EXERCISE 9.7</u> A more precise formulation of lemma 9.2 can be given along the following lines:

We may assume that the language of FOL contains propositional variables – these can be viewed as nullary relation symbols. Moreover, it is always possible to substitute a formula for such a propositional variable, and obtain a new formula: F^a_A is the formula obtained by substitution of the formula A for the propositional variable a in the formula F. Now a reformulation of lemma 9.2 says that $A \Leftrightarrow B$ implies $F^a_A \Leftrightarrow F^a_B$.

Compare this result to lemma 8.9. Would it be possible to formulate also a version saying that $[\![A]\!]^M_v = [\![B]\!]^M_v$ implies $[\![F^a_A]\!]^M_v = [\![F^a_B]\!]^M_v$?

<u>EXERCISE 9.8</u> Theorem 9.8 states logical (i.e., semantic) equivalence of each formula A and its PNF A_P. Now, show that we also have a corresponding proof theoretic (syntactic) result: for each A, there is a formula A_P in PNF such that $\vdash_{\mathcal{N}} A \leftrightarrow A_P$.

<u>EXERCISE 9.9</u> [Skolem Normal Form]

This exercise involves some intricacies similar to those from Example 9.30. Let A be the closed formula $\forall x \exists y \forall z B(x,y,z)$, where B has no quantifiers.

(1) Let f be a new unary function symbol, and let A_S be $\forall x \forall z B(x, f(x), z)$.
(2) Show that A is satisfiable if and only if A_S is satisfiable.
(Hint: Show that a model for any one of the two formulae can be transformed into a model for the other.)
(3) Show that A and A_S are not logically equivalent.
(Hint: Find a structure which does not satisfy one of the implications $A \to A_S$ or $A_S \to A$.)
(4) Repeat the analogous steps 1 and 2 to the closed formula $\forall x \forall u \exists y \forall z B(x,u,y,z)$.
(5) By Theorem 9.8, each formula is equivalent to a PNF formula. Use induction on the length of the prefix of PNF A_P to show that for each FOL formula A, there is a formula A_S with only universal quantifiers (but usually new function symbols) such that A is satisfiable if and only if A_S is satisfiable. [Such form A_S of a formula A is called

"Skolem normal form".]

Chapter 10
Soundness, Completeness

- Soundness
- Completeness

1: Soundness

We show the soundness and completeness theorems for the introduced proof system $\mathcal{N}$ for FOL. As in the case of PL, soundness is an easy task.

Theorem 10.1 **[Soundness]** Let $\Gamma \subseteq \mathsf{WFF}_{\mathsf{FOL}}$ and $A \in \mathsf{WFF}_{\mathsf{FOL}}$. If $\Gamma \vdash_{\mathcal{N}} A$ then $\Gamma \models A$.

Proof. Axioms A0-A3 and MP are the same as for PL and their validity/soundness follows from the proof of the soundness theorem 6.14 for PL. Validity of A4 was shown in exercise 8.4.
It remains to show that $\exists$I preserves validity, i.e. that $M \models B \rightarrow C$ implies $M \models \exists x B \rightarrow C$ for arbitrary models M, provided x is not free in C. In fact, it is easier to show the contrapositive implication from $M \not\models \exists x B \rightarrow C$ to $M \not\models B \rightarrow C$. So suppose $M \not\models \exists x B \rightarrow C$, i.e., $M \not\models_v \exists x B \rightarrow C$ for some v. Then $M \models_v \exists x B$ and $M \not\models_v C$. Hence $M \models_{v[x \mapsto \underline{a}]} B$ for some $\underline{a}$. Since $M \not\models_v C$ and $x \notin \mathcal{V}(C)$, it follows from lemma 8.7 that also $M \not\models_{v[x \mapsto \underline{a}]} C$, hence $M \not\models_{v[x \mapsto \underline{a}]} B \rightarrow C$, i.e., $M \not\models B \rightarrow C$. **QED** (10.1)

By the same argument as in corollary 6.15, we have that every satisfiable theory is consistent or, equivalently, that inconsistent theory is not satisfiable:

Corollary 10.2 $\Gamma \vdash_{\mathcal{N}} \bot \Rightarrow Mod(\Gamma) = \varnothing$.

Proof. If $\Gamma \vdash_{\mathcal{N}} \bot$ then, by the theorem, $\Gamma \models \bot$, i.e., for any $M : M \models \Gamma \Rightarrow M \models \bot$. But there is no M such that $M \models \bot$, so $Mod(\Gamma) = \varnothing$. **QED** (10.2)

Remark.
In remark 6.16 we showed equivalence of the two soundness notions for PL. The equivalence holds also for FOL. However, the proof of the opposite implication, i.e., that

i) $\Gamma \vdash_{\mathcal{N}} \bot \Rightarrow Mod(\Gamma) = \varnothing$ implies

ii) $\Gamma \vdash_{\mathcal{N}} A \Rightarrow \Gamma \models A$,

involves an additional subtelty concerning possible presence of free variables in A. Assuming $\Gamma \vdash_{\mathcal{N}} A$ we first observe that then we also have $\Gamma \vdash_{\mathcal{N}} \forall(A)$ by lemma 7.25.4. Hence $\Gamma, \neg\forall(A) \vdash_{\mathcal{N}} \bot$ and, by i), it has no models: for any M, if $M \models \Gamma$ then $M \not\models \neg\forall(A)$. From this last fact we can conclude that $M \models \forall(A)$ – because $\forall(A)$ is closed (recall remark 8.17). By fact 8.25, we can now conclude that $M \models A$ and, since M was an arbitrary model of Γ, that $\Gamma \models A$.

2: Completeness

As in the case of PL, we prove the opposite of corollary 10.2, namely, that every consistent FOL-theory is satisfiable. Starting with a consistent theory Γ, we have to show that there is a model satisfying Γ. The procedure is thus very similar to the one applied for PL (which you might repeat before reading this section). Its main point was expressed in lemma 6.17, which has the following counterpart:

Lemma 10.3 The following two formulations of completeness are equivalent:

(1) For any $\Gamma \subseteq \mathsf{WFF}_{\mathsf{FOL}}$: $\Gamma \nvdash_{\mathcal{N}} \bot \;\Rightarrow\; Mod(\Gamma) \neq \varnothing$
(2) For any $\Gamma \subseteq \mathsf{WFF}_{\mathsf{FOL}}$: $\Gamma \models A \;\Rightarrow\; \Gamma \vdash_{\mathcal{N}} A$

Proof. 1. $\Rightarrow$ 2.) Assume 1. and $\Gamma \models A$. Let us first consider special case when A is closed. Then $\Gamma, \neg A$ is unsatisfiable and therefore, by 1., inconsistent, i.e., $\Gamma, \neg A \vdash_{\mathcal{N}} \bot$. By Deduction Theorem $\Gamma \vdash_{\mathcal{N}} \neg A \to \bot$, so $\Gamma \vdash_{\mathcal{N}} A$ by PL.
This result for closed A yields the general version: If $\Gamma \models A$ then $\Gamma \models \forall(A)$ by fact 8.27. By the argument above $\Gamma \vdash_{\mathcal{N}} \forall(A)$, and so $\Gamma \vdash_{\mathcal{N}} A$ by lemma 7.25.1 and MP.

2. $\Rightarrow$ 1.) This is shown by exactly the same argument as for PL in the proof of lemma 6.17. **QED** (10.3)

Although the general procedure, based on the above lemma, is the same, the details are now more involved as both the language and its semantics are more complex. The model we will eventually construct will be a term model for an appropriate extension of Γ. The following definitions characterize the extension we will be looking for.

Definition 10.4 A theory Γ is said to be

- *maximal consistent* iff it is consistent and, for any *closed* formula A, $\Gamma \vdash_{\mathcal{N}} A$ or $\Gamma \vdash_{\mathcal{N}} \neg A$ (cf. definition 6.18);
- a *Henkin-theory* if for each closed formula of the type $\exists xA$ there is an individual constant c such that $\Gamma \vdash_{\mathcal{N}} \exists xA \rightarrow A^x_c$;
- a *complete Henkin-theory* if it is both a Henkin-theory and maximal consistent.

In particular, every complete Henkin-theory is consistent.

Remark.
The constant c in the definition above is called a *witness* – it witnesses to the truth of the formula $\exists xA$ by providing a ground term which validates the existential quantifier. The precise definition of a Henkin-theory may vary somewhat in the literature. The condition in the lemma below looks slightly weaker than the one in the definition, but turns out to be equivalent. The proof is an easy exercise.

Lemma 10.5 Let Γ be a theory, and suppose that for every formula A with exactly one variable x free, there is a constant c_A such that $\Gamma \vdash_{\mathcal{N}} \exists xA \rightarrow A^x_{c_A}$. Then Γ is a Henkin-theory.

The properties of a complete Henkin-theory make it easier to construct a model for it. We prove first this special case of the completeness theorem:

Lemma 10.6 Every complete Henkin-theory is satisfiable.

Proof. The alphabet of any Henkin-theory will always contain an individual constant. Now consider the term structure T_Γ (we index it with Γ and not merely the alphabet, as in def. 9.25, since we will make it into a model of Γ) where:

- for each relation symbol $R \in \mathcal{R}$ of arity n, and ground terms $t_1,\ldots,t_n \in \underline{T_\Gamma}$:

$$\langle t_1,\ldots,t_n\rangle \in [\![R]\!]^{T_\Gamma} \;\Leftrightarrow\; \Gamma \vdash_{\mathcal{N}} R(t_1,\ldots,t_n)$$

We show, by induction on the number of connectives and quantifiers in a formula, that for any *closed* formula A we have $T_\Gamma \models A$ iff $\Gamma \vdash_{\mathcal{N}} A$. (From this it follows that T_Γ is a model of Γ: if $A \in \Gamma$ then $\Gamma \vdash_{\mathcal{N}} \forall(A)$, so $T_\Gamma \models \forall(A)$, i.e., by fact 8.27, $T_\Gamma \models A$.)

<u>A IS :</u>

ATOMIC :: Follows directly from the construction of T_Γ.

$$T_\Gamma \models R(t_1,\ldots,t_n) \Leftrightarrow \langle t_1,\ldots,t_n\rangle \in [\![R]\!]^{T_\Gamma} \Leftrightarrow \Gamma \vdash_{\mathcal{N}} R(t_1,\ldots,t_n).$$

$\neg B$ :: We have the following equivalences:

$$\begin{array}{rlll} T_\Gamma \models \neg B & \Leftrightarrow & T_\Gamma \not\models B & \text{definition of } \models,\ A \text{ closed} \\ & \Leftrightarrow & \Gamma \not\vdash_{\mathcal{N}} B & \text{IH} \\ & \Leftrightarrow & \Gamma \vdash_{\mathcal{N}} \neg B & \text{maximality of } \Gamma \end{array}$$

$B \to C$ :: Since $T_\Gamma \models A \Leftrightarrow T_\Gamma \not\models B$ or $T_\Gamma \models C$, we have two cases. Each one follows easily by IH and maximality of Γ.

$\exists xB$ :: Since A is closed and Γ is a Henkin-theory, we have $\Gamma \vdash_{\mathcal{N}} \exists xB \to B^x_c$ for some c. Now if $\Gamma \vdash_{\mathcal{N}} A$ then also $\Gamma \vdash_{\mathcal{N}} B^x_c$ and, by IH, $T_\Gamma \models B^x_c$, hence $T_\Gamma \models \exists xB$ by soundness of A4.
For the converse, assume that $T_\Gamma \models \exists xB$, i.e., there is a $t \in \underline{T_\Gamma}$ such that $T_\Gamma \models_{x\mapsto t} B$. But then $T_\Gamma \models B^x_t$ by lemmas 8.9 and 8.7, so by IH, $\Gamma \vdash_{\mathcal{N}} B^x_t$, and by A4 and MP, $\Gamma \vdash_{\mathcal{N}} A$.

QED (10.6)

The construction in the above proof is only guaranteed to work for complete Henkin-theories. The following examples illustrate why.

Example 10.7
For Γ in general, T_Γ may fail to satisfy some formulae from Γ (or $Th(\Gamma)$) because:

(1) Some (atomic) formulae are not provable from Γ:
Let Σ contain two constant symbols a and b and one binary relation R. Let Γ be a theory over Σ with one axiom $R(a,b) \vee R(b,a)$. Each model of Γ must satisfy at least one disjunct but, since $\Gamma \not\vdash_{\mathcal{N}} R(a,b)$ and $\Gamma \not\vdash_{\mathcal{N}} R(b,a)$, none of these relations will hold in T_Γ.
(2) The interpretation domain $\underline{T_\Gamma}$ has too few elements:
It may happen that $\Gamma \vdash_{\mathcal{N}} \exists xR(x)$ but $\Gamma \not\vdash_{\mathcal{N}} R(t)$ for any ground term t. Since only ground terms are in $\underline{T_\Gamma}$, this would again mean that $T_\Gamma \not\models \exists xR(x)$.

□

A general assumption to be made in the following is that the alphabet Σ under consideration is countable (i.e., has at most countably infinitely many symbols). Although not necessary, it is quite reasonable and, besides, makes the arguments clearer.

We now strengthen lemma 10.6 by successively removing the extra assumptions about the theory. First we show that the assumption about *maximal* consistency is superfluous; every consistent theory can be extended to a maximal consistent one. (We say that Γ' is an *extension* of Γ if $\Gamma \subseteq \Gamma'$.)

Lemma 10.8 Let Σ be a countable alphabet. Every consistent theory Γ over Σ has a maximal consistent extension $\widehat{\Gamma}$ over the same Σ.

Proof. Since $|\Sigma| \leq \aleph_0$, there are at most $\aleph_0$ Σ-formulae. Choose an enumeration $A_0, A_1, A_2, \ldots$ of all *closed* Σ-formulae. Construct an increasing sequence of theories as follows:

$$\begin{aligned} \text{Basis} &:: \Gamma_0 = \Gamma \\ \text{Ind.} &:: \Gamma_{n+1} = \begin{cases} \Gamma_n, A_n & \text{if it is consistent} \\ \Gamma_n, \neg A_n & \text{otherwise} \end{cases} \\ \text{Clsr.} &:: \widehat{\Gamma} = \bigcup_{n \in \mathbb{N}} \Gamma_n \end{aligned}$$

We show by induction on n that for any n, Γ_n is consistent.

Basis :: $\Gamma_0 = \Gamma$ is consistent by assumption.

Ind. :: Suppose Γ_n is consistent. If Γ_{n+1} is inconsistent, then from the definition of Γ_{n+1} we know that both Γ_n, A_n and $\Gamma_n, \neg A_n$ are inconsistent, hence by Deduction Theorem both $A_n \to \bot$ and $\neg A_n \to \bot$ are provable from Γ_n. By Exercise 4.1.4, Γ_n proves then both A_n and $\neg A_n$, which contradicts its consistency by Exercise 4.5.

Now by theorem 4.29 (which holds for FOL by the same argument as in PL), $\widehat{\Gamma}$ is consistent iff each of its finite subtheories is. Any finite subtheory of $\widehat{\Gamma}$ will be included in some Γ_n, so $\widehat{\Gamma}$ is consistent. From the definition of $\widehat{\Gamma}$ it now follows that $\widehat{\Gamma}$ is also *maximal* consistent. **QED** (10.8)

Corollary 10.9 Let Σ be a countable alphabet. Every consistent Henkin-theory over Σ is satisfiable.

Proof. If Γ is a consistent Henkin-theory, it has an extension $\widehat{\Gamma}$ which is maximal consistent. Now since $\Gamma \subseteq \widehat{\Gamma}$ and both are theories over the same alphabet Σ, it follows that $\widehat{\Gamma}$ is a Henkin-theory if Γ is. Hence $\widehat{\Gamma}$ is a complete Henkin-theory and so has a model, which is also a model of Γ. **QED** (10.9)

To bridge the gap between this result and theorem 10.15 we need to show that every consistent theory has a consistent Henkin extension – we shall make use of the following auxiliary notion.

Definition 10.10 Let Σ and Σ' be two alphabets, and assume $\Sigma \subseteq \Sigma'$. Moreover, let Γ be a Σ-theory and let Γ' be a Σ'-theory. Then Γ' is said to be a *conservative extension* of Γ, written $\Gamma \preceq \Gamma'$, if $\Gamma \subseteq \Gamma'$ and for all Σ-formulae A: if $\Gamma' \vdash_{\mathcal{N}} A$ then $\Gamma \vdash_{\mathcal{N}} A$.

Thus, a conservative extension Γ' of Γ may prove more formulae involving the symbols from the extended alphabet – but any formula over the alphabet of Γ which is provable from Γ' must be provable from Γ itself. The next lemma records a few useful facts about conservative extensions. The proof is left as an exercise.

Lemma 10.11 The conservative extension relation $\preceq$:

(1) preserves consistency: a conservative extension of a consistent theory is consistent.
(2) is transitive: if $\Gamma_1 \preceq \Gamma_2$ and $\Gamma_2 \preceq \Gamma_3$ then $\Gamma_1 \preceq \Gamma_3$.
(3) is preserved in limits: if $\Gamma_1 \preceq \Gamma_2 \preceq \Gamma_3 \preceq \ldots$ is an infinite sequence with each theory being a conservative extension of the previous, then $\bigcup_{n\in\mathbb{N}} \Gamma_n$ is a conservative extension of Γ_1.

We shall make use of these facts in a moment, but first we note the following important lemma.

Lemma 10.12 Let Γ be a Σ-theory and let A be a Σ-formula with at most x free. Moreover let c be an individual constant that does not occur in Σ and let Σ' be $\Sigma \cup \{c\}$. Then the Σ'-theory $\Gamma \cup \{\exists x A \to A^x_c\}$ is a conservative extension of Γ.

Proof. Let B be an arbitrary Σ-formula, which is provable from the extended theory, i.e.,

$1 : \Gamma, \exists x A \to A^x_c \vdash_{\mathcal{N}} B$

$2 : \Gamma \vdash_{\mathcal{N}} (\exists x A \to A^x_c) \to B$ $DT +$ at most x free in A

But this means that Γ, with axioms *not involving any* occurrences of c, proves the indicated formula which has such occurrences. Thus we may choose a new variable y (not occurring in this proof) and replace all occurrences of c in this proof by y. We will then obtain

$3 : \Gamma \vdash_{\mathcal{N}} (\exists x A \to A^x_y) \to B$
$4 : \Gamma \vdash_{\mathcal{N}} \exists y(\exists x A \to A^x_y) \to B$ $\exists$I $+y$ not free in B
$5 : \Gamma \vdash_{\mathcal{N}} \exists y(\exists x A \to A^x_y)$ Exc. 7.2.2 + 7.5
$6 : \Gamma \vdash_{\mathcal{N}} B$ $MP(5,4)$

QED (10.12)

Lemma 10.13 Let Σ be a countable alphabet, and let Γ be a Σ-theory. Then there exists a countable alphabet Σ_H and a Σ_H-theory Γ_H such that $\Sigma \subseteq \Sigma_H$, $\Gamma \preceq \Gamma_H$, and Γ_H is a Henkin-theory over Σ_H.

Proof. Let Γ be a Σ-theory. Extend the alphabet Σ to $H(\Sigma)$ by adding, for each Σ-formula A with exactly one variable free, a new constant c_A. Let $H(\Gamma)$ be the $H(\Sigma)$-theory obtained by adding to Γ, for each such A, the new axiom

$$\exists x A \to A^x_{c_A},$$

where x is the free variable of A.

In particular, $H(\Gamma)$ can be obtained by the following iterated construction: we enumerate all formulae A with exactly one variable free, getting $A_0, A_1, A_2, A_3, \ldots$. For any n, let x_n be the free variable of A_n. We take

$$\begin{array}{rl}
\textsc{Basis} :: & \Gamma_0 = \Gamma \text{ and } \Sigma_0 = \Sigma \\
\textsc{Ind.} :: & \Gamma_{n+1} = \Gamma_n, \exists x_n A_n \to (A_n)^{x_n}_{c_{A_n}}, \text{ and} \\
& \Sigma_{n+1} = \Sigma_n \cup \{c_{A_n}\} \\
\textsc{Clsr.} :: & H(\Gamma) = \bigcup_{n\in\mathbb{N}} \Gamma_n \text{ and } H(\Sigma) = \bigcup_{n\in\mathbb{N}} \Sigma_n.
\end{array} \qquad (10.14)$$

By lemma 10.12, each theory Γ_{n+1} is a conservative extension of Γ_n, and hence by lemma 10.11 $H(\Gamma)$ is a conservative extension of Γ. It is also clear that $H(\Sigma)$ is countable, since only countably many new constants are added.

$H(\Gamma)$ is however not a Henkin-theory, since we have not ensured the provability of appropriate formulae $\exists x A \to A^x_c$ for $H(\Sigma)$-formulae A that are not Σ-formulae. For instance, for $R \in \Sigma$

$$H(\Gamma) \vdash_{\mathcal{N}} \exists x \exists y R(x,y) \to \exists y R(c_{\exists y R(x,y)}, y),$$

but there may be no c such that

$$H(\Gamma) \vdash_{\mathcal{N}} \exists y R(c_{\exists y R(x,y)}, y) \to R(c_{\exists y R(x,y)}, c).$$

To obtain a Henkin-theory, the construction (10.14) has to be iterated, i.e, the sequence of theories $\Gamma, H(\Gamma), H^2(\Gamma), H^3(\Gamma), \ldots$ is constructed (where $H^{n+1}(\Gamma)$ is obtained by starting (10.14) with $\Gamma_0 = H^n(\Gamma)$), and Γ_H is defined as the union of them all $\bigcup_{n\in\mathbb{N}} H^n(\Gamma)$.
The sequence of corresponding alphabets $\Sigma, H(\Sigma), H^2(\Sigma), H^3(\Sigma), \ldots$ are collected into a corresponding union Σ_H. Γ_H is a Σ_H-theory and Σ_H, being the union of countably many countable sets, is itself countable.
Since each theory $H^{n+1}(\Gamma)$ is a conservative extension of $H^n(\Gamma)$, it follows by lemma 10.11 that Γ_H is a conservative extension of Γ.
Finally we check that Γ_H is a Henkin-theory over Σ_H: let A be any Σ_H-formula with exactly x free. A contains only finitely many symbols, so A is also a $H^n(\Sigma)$-formula for some n. But then $\exists x A \to A^x_{c_A}$ is contained in $H^{n+1}(\Gamma)$, and hence in Γ_H. By lemma 10.5, this proves that Γ_H is a Henkin-theory. **QED** (10.13)

Gathering all the pieces we thus obtain the main result.

Theorem 10.15 Let Σ be a countable alphabet. Every consistent theory over Σ is satisfiable.

Proof. Let Σ be countable. Suppose Γ is a consistent Σ-theory. Then there exist Γ_H and Σ_H with the properties described in lemma 10.13. Since Γ is consistent, Γ_H, being a conservative extension, must be consistent as well. By corollary 10.9, Γ_H (and hence Γ) has a Σ_H-model. This can be converted to a Σ-model by "forgetting" the interpretation of symbols in $\Sigma_H \setminus \Sigma$. **QED** (10.15)

This is the strongest version that we prove here. The assumption about countability is however unnecessary, and the following version is also true.

Theorem 10.16 Every consistent theory is satisfiable.

Lemma 10.3 and soundness yield then the final result:

Corollary 10.17 For any $\Gamma \subseteq \mathsf{WFF}_{\mathsf{FOL}}$, $A \in \mathsf{WFF}_{\mathsf{FOL}}$ we have the following.

(1) $\Gamma \models A$ iff $\Gamma \vdash_{\mathcal{N}} A$.
(2) $Mod(\Gamma) \neq \varnothing$ iff $\Gamma \nvdash_{\mathcal{N}} \bot$, i.e. Γ is satisfiable iff it is consistent.

2.1: Some Applications

We list here some typical questions, the answers to which may be significantly simplified by using soundness and completeness theorem. These are the same questions as we listed earlier in subsection 4.1 after the respective theorems for statement logic. The schemata of the arguments are also the same as before, because they are based exclusively on the soundness and completeness of the respective axiomatic system. The differences concern, of course, the semantic definitions which are more complicated for FOL, than they were for PL.

1. Is a formula provable?
If it is, it may be worth trying to construct a syntactic proof of it. Gentzen's system is easiest to use, so it can be most naturally used for this purpose. However, one should first try to make a "justified guess". To make a guess, we first try to see if we can easily construct a counter example, i.e., a structure which falsifies the formula. For instance, is it the case that:

$$\vdash_{\mathcal{N}} (\exists x P(x) \to \exists x Q(x)) \;\to\; \forall x(P(x) \to Q(x)) \;? \tag{10.18}$$

Instead of starting to look for a syntactic proof, we better think first. Can we falsify this formula, i.e., find a structure M such that

$$M \models \exists x P(x) \to \exists x Q(x) \tag{10.19}$$

and

$$M \not\models \forall x(P(x) \to Q(x)) \;? \tag{10.20}$$

More explicitly, (10.19) requires that either

$$\text{for all } m_1 \in \underline{M} : [\![P(x)]\!]^M_{x\mapsto m_1} = \mathbf{0} \;\text{ or }\; \text{for some } m_2 \in \underline{M} : [\![Q(x)]\!]^M_{x\mapsto m_2} = \mathbf{1} \tag{10.21}$$

while (10.20) that

$$\text{for some } m \in \underline{M} : [\![P(x)]\!]^M_{x\mapsto m} = \mathbf{1} \;\text{ and }\; [\![Q(x)]\!]^M_{x\mapsto m} = \mathbf{0}. \tag{10.22}$$

But this should be easy to do. Let $\underline{M} = \{m_1, m_2\}$ and $[\![P]\!]^M = \{m_1\}$ and $[\![Q]\!]^M = \{m_2\}$. This makes (10.21) true since $[\![Q(x)]\!]^M_{x\mapsto m_2} = \mathbf{1}$. On the otehr hand (10.22) holds for $m_1 : [\![P(x)]\!]^M_{x\mapsto m_1} = \mathbf{1}$ and $[\![Q(x)]\!]^M_{x\mapsto m_1} = \mathbf{0}$. Thus, the formula from (10.18) is not valid and, **by soundness** of $\vdash_{\mathcal{N}}$, is not provable.

This is, in fact, the only general means of showing that a formula is *not* provable in a sound system which is not decidable ($\vdash_{\mathcal{N}}$ is not) – to find a structure providing a counter example to validity of the formula.

If such an analysis fails, i.e., if we are unable to find a counter example, it may indicate that we should rather try to construct a proof of the formula in our system. **By completeness** of this system, such a proof will exist, if the formula is valid.

2. Is a formula valid?
For instance, is it the case that

$$\models \forall x(P(x) \to Q(x)) \to (\exists x P(x) \to \exists Q(x))? \tag{10.23}$$

We may first try to see if we can find a counter example. In this case, we need a structure M such that $M \models \forall x(P(x) \to Q(x))$ and $M \not\models \exists x P(x) \to \exists x Q(x)$ – since both (sub)formulae are closed we need not consider particular assignments. Thus, M should be such that

$$\text{for all } m \in \underline{M} : [\![P(x) \to Q(x)]\!]^M_{x \mapsto m} = \mathbf{1}. \tag{10.24}$$

To falsify the other formula we have to find an

$$m_1 \in \underline{M} \ \text{ such that } [\![P(x)]\!]^M_{x \mapsto m_1} = \mathbf{1} \tag{10.25}$$

and such that

$$\text{for all } m_2 \in \underline{M} : [\![Q(x)]\!]^M_{x \mapsto m_2} = \mathbf{0}. \tag{10.26}$$

Assume that m_1 is as required by (10.25). Then (10.24) implies that we also have $[\![Q(x)]\!]^M_{x \mapsto m_1} = \mathbf{1}$. But this means that (10.26) cannot be forced, m_1 being a witness contradicting this statement. Thus, the formula from (10.23) cannot be falsified in any structure, i.e., it is valid. This is sufficient argument – direct, semantic proof of validity of the formula.

However, such semantic arguments involve complicating subtelities which may easily confuse us when we are using them. If we have a strong conviction that the formula indeed is valid, we may instead attempt a syntactic proof. Below, we are doing it in Gentzen's system – soundness and completeness theorems hold for this system as well. (Notice that we first eliminate the quantifier from $\exists x P(x)$ since this requires a fresh variable y; the subsequent substitutions must be legal but need not introduce fresh variables.)

$$\begin{array}{l}
P(y) \vdash_{\mathcal{G}} Q(y), P(y) \quad ; \quad Q(y), P(y) \vdash_{\mathcal{G}} Q(y) \\ \hline
\qquad\quad P(y) \to Q(y), P(y) \vdash_{\mathcal{G}} Q(y) \\ \hline
\qquad \forall x(P(x) \to Q(x)), P(y) \vdash_{\mathcal{G}} \exists x Q(x) \\ \hline
\forall x(P(x) \to Q(x)), \exists x P(x) \vdash_{\mathcal{G}} \exists x Q(x) \\ \hline
\qquad\qquad \forall x(P(x) \to Q(x)) \vdash_{\mathcal{G}} \exists x P(x) \to \exists x Q(x) \\ \hline
\qquad\qquad\qquad\qquad\qquad\qquad \vdash_{\mathcal{G}} \forall x(P(x) \to Q(x)) \to (\exists x P(x) \to \exists x Q(x))
\end{array}$$

Having this proof we conclude, **by soundness** of $\vdash_{\mathcal{G}}$, that the formula is indeed valid.

Summarising these two points.
In most axiomatic systems the relation $X \vdash Y$ is semi-decidable: to establish that it holds, it is enough to generate all the proofs until we encounter one which proves Y from X. Therefore, if this actually holds, it may be natural to try to construct a syntactic proof (provided that the axiomatic system is easy to use, like $\vdash_{\mathcal{G}}$) – completeness of the system guarantees that there exists a proof of a valid formula. If, however, the relation does not hold, it is always easier to find a semantic counter example. If it is found, and the system is sound, it allows us to conclude that the relation $X \vdash Y$ does *not* hold. That is, in order to know what is easier to do, we have to know what the answer is! This is, indeed, a vicious circle, and the best one can do is to "guess" the right answer before proving it. The quality of such "guesses" increases only with exercise and work with the system itself and cannot be given in the form of a ready-made recipe.

3. Is a rule admissible?
Suppose that we have an axiomatic system $\vdash$and a rule $R: \dfrac{\Gamma \vdash A_1 \ldots \Gamma \vdash A_n}{\Gamma \vdash C}$. The question whether R is admissible in R can be answered by trying to verify by purely proof theoretic means that any given proofs for the premises entitle the existence of a proof for the conclusion C. This, however, is typically a cumbersome task.

If the system is sound and complete, there is a much better way to do that. The schema of the proof is as follows. For the first, we verify if the rule is sound. If it isn't, we can immediately conclude, **by soundness** of our system, that it is not admissible. If, on the otehr hand, the rule is sound, the following schematic argument allows us to conclude that it is admissible:

$$\frac{\Gamma \vdash A_1 \ldots \Gamma \vdash A_n}{\Gamma \vdash C} \quad \begin{array}{c} \overset{\text{soundness}}{\Longrightarrow} \\ \underset{\text{completeness}}{\Longleftarrow} \end{array} \quad \frac{\Gamma \models A_1 \ldots \Gamma \models A_n}{\Gamma \models C} \Downarrow \text{soundness of } R$$

For instance, are the following rules admissible in $\vdash_{\mathcal{N}}$:

$$i)\ \frac{\vdash_{\mathcal{N}} \exists x(P(x) \to Q(x)) \quad ; \quad \vdash_{\mathcal{N}} \forall x P(x)}{\vdash_{\mathcal{N}} \exists x Q(x)} \qquad ii)\ \frac{\vdash_{\mathcal{N}} \exists x(P(x) \to Q(x)) \quad ; \quad \vdash_{\mathcal{N}} \exists x P(x)}{\vdash_{\mathcal{N}} \exists x Q(x)}\ ?$$

The first thing to check is whether the rules are sound, that is, assume that M is an arbitrary structure which satisfies the premises. For the rule i), this means

$$\begin{array}{ll} M \models \exists x(P(x) \to Q(x)) & \textit{and } M \models \forall x P(x) \\ \textit{i.e.} \text{ for some } m \in \underline{M}: & \textit{and} \text{ for all } n \in \underline{M}: \\ \quad [\![P(x) \to Q(x)]\!]^M_{x \mapsto m} = \mathbf{1} & \quad [\![P(x)]\!]^M_{x \mapsto n} = \mathbf{1} \end{array} \tag{10.27}$$

Will M satisfy the conclusion? Let m be the witness making the first assumption true, i.e., either $[\![P(x)]\!]^M_{x \mapsto m} = \mathbf{0}$ or $[\![Q(x)]\!]^M_{x \mapsto m} = \mathbf{1}$. But from the second premise, we know that for all n, in particular for the chosen $m : [\![P(x)]\!]^M_{x \mapsto m} = \mathbf{1}$. Thus, it must be the case that $[\![Q(x)]\!]^M_{x \mapsto m} = \mathbf{1}$. But then m is also a witness to the fact that $[\![\exists x Q(x)]\!]^M = \mathbf{1}$, i.e., the rule is sound. By the above argument, i.e., **by soundness and completeness** of $\vdash_{\mathcal{N}}$, the rule i) is admissible.

For the second rule ii), we check first its soundness. Let M be an arbitrary structure satisfying the premises, i.e.:

$$\begin{array}{ll} M \models \exists x(P(x) \to Q(x)) & \textit{and } M \models \exists x P(x) \\ \textit{i.e.} \text{ for some } m \in \underline{M}: & \textit{and} \text{ for some } n \in \underline{M}: \\ \quad [\![P(x) \to Q(x)]\!]^M_{x \mapsto m} = \mathbf{1} & \quad [\![P(x)]\!]^M_{x \mapsto n} = \mathbf{1} \end{array} \tag{10.28}$$

Here it is possible that $m \neq n$ and we can utilize this fact to construct an M which does not satisfy the conlcusion. Let $\underline{M} = \{m, n\}$ with $[\![P]\!]^M = \{n\}$ and $[\![Q]\!]^M = \varnothing$. Both assumtptions from (10.28) are now satisfied. However, $[\![Q]\!]^M = \varnothing$, and so $M \not\models \exists x Q(x)$. Thus the rule is not sound and, **by soundness** of $\vdash_{\mathcal{N}}$, can not be admissible there.

2.2: COMPLETENESS OF GENTZEN'S SYSTEM .... [optional]

Recall exercise 6.9 and the discussion just after the introduction of Gentzen's system for FOL in Section 7.4.

(A) We assume that the new rules added in the FOL system are invertible in the same sense as are the propositional rules (exercise 6.9). More precisely, for any rule $\dfrac{\Gamma_i \vdash_{\mathcal{G}} \Delta_i}{\Gamma \vdash_{\mathcal{G}} \Delta}$, if the conclusion is valid, $\bigwedge \Gamma \Rightarrow \bigvee \Delta$, then so are all the assumptions, $\bigwedge \Gamma_i \Rightarrow \bigvee \Delta_i$. We can strengthen this, since this implication holds also when validity is replaced by truth in an arbitrary structure. I.e., for any structure M, if $M \models \bigwedge \Gamma \to \bigvee \Delta$ then likewise $M \models \bigwedge \Gamma_i \to \bigvee \Delta_i$ for each assumption i. Verification of this fact is left as exercise 10.6.

(B) We proceed as we did in exercise 6.9, constructing a counter-model for any unprovable seuquent. But now we have to handle the additional complications of possibly non-terminating derivations. To do this, we specify the following strategy for an exhaustive bottom-up proof search.

(B.1) First, we have to ensure that even if a branch of a proof does not terminate, all formulae in the sequent are processed. Let us therefore view a sequent as a pair of (finite) sequences $\Gamma = G_1, ..., G_a$ and $\Delta = D_1, ..., D_c$. Such a sequent is processed by applying bottom-up the appropriate rule first to G_1, then to G_2, to G_3, etc. until G_a, and followed by D_1 through D_c. If no rule is applicable to a formula, it is skipped and one continues with the next formula. New formulae arising from rule applications are always placed at the start of the appropriate sequence (on the left or on the right of $\vdash_{\mathcal{G}}$). These restrictions are particularly important for the quantifier rules which introduce new formulae, i.e.:

$$6.\ \vdash\exists\ \frac{\Gamma \vdash_{\mathcal{G}} A^x_t \ldots \exists x A \ldots}{\Gamma \vdash_{\mathcal{G}} \ldots \exists x A \ldots}\ A^x_t\ legal \qquad 6'.\ \forall\vdash\ \frac{A^x_t \ldots \forall x A \ldots \vdash_{\mathcal{G}} \Delta}{\ldots \forall x A \ldots \vdash_{\mathcal{G}} \Delta}\ A^x_t\ legal$$

These two rules might start a non-terminating, repetitive process introducing new substitution instances of A without ever considering the remaining formulae. Processing formulae from left to right, and placing new formulae to the left of the actually processed ones, makes sure that all formulae in a sequent will be processed, before starting the processing of the newly introduced formulae.

(B.2) We must also ensure that all possible substitution instances of quantified formulae are attempted in search for axioms. To do this, it suffices to require that the formula A^x_t, introduced by an application of rule 6 or 6', does not already occur in the sequence to which it is introduced.

(C) Let now $\Gamma \vdash_{\mathcal{G}} \Delta$ be an arbitrary sequent for which the above strategy does not yield a proof. There are two cases.

(C.1) If every branch in the obtained proof tree is finite, then all its leafs contain irreducible sequents, some of which are non-axiomatic. Select such a non-axiomatic leaf, say, with $\Phi \vdash_{\mathcal{G}} \Psi$. Irreducibility means that no rule can be applied to this sequent, i.e., all its formulae are atomic. That it is non-axiomatic means that $\Phi \cap \Psi = \varnothing$. Construct a counter-model M by taking all terms occurring in the atoms of Φ, Ψ as the intepretation domain $\underline{M}$. (In particular, if there are open atomic formulae, their variables are treated as elements on line with ground terms.) Interpret the predicates over these elements by making all atoms in Φ true and all atoms in Ψ false. This is possible since $\Phi \cap \Psi = \varnothing$. (E.g., a leaf $P(x) \vdash_{\mathcal{G}} P(t)$ gives the structure with $\underline{M} = \{x, t\}$ where $t \notin [\![P]\!]^M = \{x\}$.)

Invertibility of the rules implies now that this is also a counter-model to the initial sequent, i.e., $\bigwedge \Gamma \not\Rightarrow \bigvee \Delta$.

(C.2) In the other case, the resulting tree has an infinite branch. We then select an arbitrary infinite branch B and construct a counter-model M from all the terms occurring in the atomic formulae on B. (Again, variables occurring in such formulae are taken as elements on line with ground terms.) The predicates are defined by

$$\bar{t} \in [\![P]\!]^M \text{ iff there is a node on } B \text{ with } P(\bar{t}) \text{ on the left of } \vdash_{\mathcal{G}} \quad (*)$$

The claim is now that M is a counter-model to every sequent on the whole B. We show, by induction on the complexity of the formulae, that all those occurring on B on the left of $\vdash_{\mathcal{G}}$ are true and all those on the right false. The claim is obvious for the atomic formulae by the definition (*). In particular, since atomic formulae remain unchanged once they appear on B, and since no node is axiomatic, no formula occurring on the left of $\vdash_{\mathcal{G}}$ in B occurs also on the right. The claim is easily verified for the propositional connectives by the invertibility of the propositional rules. Consider now any quantified formula occurring on the left. Due to fairness strategy (B.1), it has been processed. If it is universal, it has been processed by the rule:

$$6'.\ \forall\vdash\ \frac{A^x_t \ldots \forall x A \ldots \vdash_{\mathcal{G}} \Delta}{\ldots \forall x A \ldots \vdash_{\mathcal{G}} \Delta}\ \ A^x_t\ legal.$$

Then $[\![A^x_t]\!]^M = \mathbf{1}$ by IH and, moreover, since by (B.2) all such substitution instances are tried on every infinite branch, so $[\![A^x_{t_i}]\!]^M = \mathbf{1}$ for all terms t_i. But this means that $M \models \forall x A$. If the formula is existential, it is processed by the rule:

$$7'.\ \exists\vdash\ \frac{\Gamma, A^x_{x'} \vdash_{\mathcal{G}} \Delta}{\Gamma, \exists x A \vdash_{\mathcal{G}} \Delta}\ \ x'\ fresh.$$

Then x' occurs in the atomic subformula(e) of A and is an element of $\underline{M}$. By IH, $[\![A^x_{x'}]\!]^M = \mathbf{1}$ and hence also $[\![\exists x A]\!]^M = \mathbf{1}$.

Similarly, by (B.1) every quantified formula on the right of $\vdash_{\mathcal{G}}$ has been processed. Universal one was processed by the rule

$$7.\ \ \vdash\forall\ \frac{\Gamma \vdash_{\mathcal{G}} A^x_{x'}, \Delta}{\Gamma, \vdash_{\mathcal{G}} \forall x A, \Delta}\ \ x'\ fresh$$

Then $x' \in \underline{M}$ and, by IH, $[\![A^x_{x'}]\!]^M = \mathbf{0}$. Hence also $[\![\forall x A]\!]^M = \mathbf{0}$. An existential formula on the right is processd by the rule

$$6.\ \ \vdash\exists\ \frac{\Gamma \vdash_{\mathcal{G}} A^x_t \ldots \exists x A \ldots}{\Gamma \vdash_{\mathcal{G}} \ldots \exists x A \ldots}\ \ A^x_t\ legal$$

and, by the exhaustive fairness strategy (B.2), all such substitution instances $A^x_{t_i}$ appear on the right of $\vdash_{\mathcal{G}}$ in B. By IH, $[\![A^x_{t_i}]\!]^M = \mathbf{0}$ for all t_i, which means that $[\![\exists x A]\!]^M = \mathbf{0}$.

(D) Since the sequent $\Gamma \vdash_{\mathcal{G}} \Delta$ is on B, (C.1) and (C.2) together show that $M \not\models \bigwedge\Gamma \to \bigvee\Delta$., i.e., that unprovability of a sequent, $\Gamma \not\vdash_{\mathcal{G}} \Delta$, implies the existence of a counter-model for it, $\bigwedge\Gamma \not\Rightarrow \bigvee\Delta$. Formulated contrapositively, if a sequent is valid (has no counter-model), $\bigwedge\Gamma \Rightarrow \bigvee\Delta$, then it is provable, $\Gamma \vdash_{\mathcal{G}} \Delta$.

.... [end optional]

Exercises 10.

EXERCISE 10.1 Show the inductive step for the case $B \to C$, which was omitted in the proof of Lemma 10.6.

EXERCISE 10.2 Let Σ contain one constant $\odot$ and one unary function s. Let T_Σ denote its term structure. Show that

(1) T_Σ is set-isomorphic to the set $\mathbb{N}$ of natural numbers,

(2) T_Σ with the ordering of terms induced by their inductive definition is order-isomorphic (definition 1.17) to $\mathbb{N}$ with $<$.

EXERCISE 10.3 Show that the formula 1. from lemma 7.25 is provable, i.e., that $\Gamma \vdash_{\mathcal{N}} \forall x A \rightarrow A$, without constructing the actual syntactic proof.

EXERCISE 10.4 Show that the rules 2., 3. and 4. from lemma 7.25 are admissible in $\mathcal{N}$ without constructing any syntactic proofs (like in the proof of the lemma).

EXERCISE 10.5 Prove the three statements of lemma 10.11.

(Hint: In the proof of the last point, you will need (a form of) compactness, i.e., if $\Gamma \vdash_{\mathcal{N}} A$, then there is a finite subtheory $\Delta \subseteq \Gamma$ such that $\Delta \vdash_{\mathcal{N}} A$.)

EXERCISE 10.6 Show that the quantifer rules of Gentzen's system for FOL, 6, 6', 7 and 7' are invertible, i.e., that every structure M in which the conclusion of the rule is satisfied, then so is its premise.

EXERCISE 10.7 Suppose the definition of a maximal consistent theory (definition 10.4) is strengthened to the requirement that $\Gamma \vdash_{\mathcal{N}} A$ or $\Gamma \vdash_{\mathcal{N}} \neg A$ for *all* formulae, not only the closed ones. In this case lemma 10.8 would no longer be true. Explain why.

(Hint: Let P be a unary predicate, a, b two constants and $\Gamma = \{P(a), \neg P(b)\}$. Γ is consistent, but what if you add to it open formula $P(x)$, resp. $\neg P(x)$? Recall discussion from remark 8.17, in particular, the fact (8.20).)

optional

EXERCISE 10.8 Prove that every consistent theory over a countable alphabet has a countable model.

(Hint: You need only find the relevant lemmata. Essentially, you repeat the proof of completeness verifying that each step preserves countability and, finally, that this leads to a countable model (in the proof of lemma 10.6). Specify only the places which need adjustments – and, of course, which adjustments.)

Chapter 11
Identity and Some Consequences

- Identity
- A few more bits of Model Theory
  - Compactness
  - Skolem-Löwenheim
- Semi- and Undecidability of FOL
- Second-Order Logic: essential Incompleteness

♢——————a Background Story——————♢

Identity is the concept which has been in the center of philosophical discussions for centuries. The principle of identity, namely, the statement that for any $x : x = x$, has been postulated as the ultimate law of the universe or, in any case, of our thinking. But, in a sense, identity seems to be an abvious relation which should not cause too much problems. Any thing is itself and any two things are not identical. What is all the fuzz about?

To appreciate that the situation may not be so simple, we should carefully distinguish between the *semantical* identity relation on the one hand, and the identity which is claimed in a language, let us say, the *syntactic* identity. The former is a relation between "things in the world", while the latter is a linguistic statement of such a relation.

It does not make much sense to talk about identity of things – two things are two precisely because they are not identical, while any thing is identical to itself, and there is not much more to say about that. However, from the syntactic, or linguistic, point of view, the situation is less simple. If I state "The Morning Star is the same as the Evening Star" I am not stating any triviality of the kind $x = x$. I claim that two different terms, "the Morning Star" and "the Evening Star" refer to the same thing. Historically, this claim had been, for some time, considered as false and only later discovered to be true. We are using language to describe and communicate the world; in this language there are many words, terms, phrases which are intended to denote various things. And the fact that two different phrases, two *different meanings, happen to denote the same thing, i.e., to have the same extension, is often a highly non-trivial fact.*

In our context of logical systems, this is the issue which will be at stake. Given a language with some intended semantics, and given a way of stating equality of terms from this language, how can we justify, that is, prove, that different terms actually have to denote the same thing? As we will see, FOL *provides such a means, but we will have to augment it with some additional definitions in order to capture the nature of identity.*

1: FOL with Identity

So far we have no means to reason about identities within the logic, or to assert such claims as

$$\forall x \forall y\ (\ R(x,y) \wedge R(y,x) \rightarrow x \equiv y\ )$$
$$\forall x\ \neg(\ mother(x)\ \equiv\ father(x)\ )$$
$$\forall x \forall y\ \neg(\ mother(x)\ \equiv\ father(y)\ )$$

i.e., to express that R is antisymmetric, to say that anybody's mother is distinct from his or her father, or (stronger) that the sets of mothers and fathers are disjoint. It is very easy to make such an extension; just add the logical symbol $\equiv$ for identity, with a corresponding, *fixed* interpretation-rule: definition 7.5 of $\mathsf{WFF}_{\mathsf{FOL}}$ is extended with the clause

$$\text{If } t_1, t_2 \text{ are terms over } \Sigma \text{ then } t_1 \equiv t_2 \in \mathsf{WFF}^{\Sigma}_{\mathsf{FOL}}. \tag{11.1}$$

and definition 8.4 of semantics is extended with the clause

$$[\![t_1 \equiv t_2]\!]^M_v = \mathbf{1} \ \text{ iff } \ [\![t_1]\!]^M_v = [\![t_2]\!]^M_v. \tag{11.2}$$

Note that this is equivalent to treating $\equiv$ as a binary relation symbol that should *always* be interpreted as identity, i.e., $[\![\equiv]\!]^M = id_{\underline{M}}$ for any structure M.

Remark 11.3 **[Identity is not axiomatizable]**
The symbol $\equiv$ is "*logical*" because it has a *fixed* meaning – its interpretation as identity is built into any system using it. Other (relational) symbols, whose axiomatization as well as interpretation in various structures may vary are called "non-logical". The reason for introducing identity in this way is that ... it cannot be introduced otherwise! If we introduce a non-logical binary relation symbol, say

$\sim$, then no matter how we axiomatize it, there will always be structures satisfying these axioms where $\sim$ is *not* interpreted as identity. We now show this claim.

Assume that we have an alphabet Σ with $\sim$ and some axiom set Γ which is supposed to force $\sim$ to be interpreted as identity, i.e., as in definition (11.2). Let $M \models \Gamma$. Pick any element $m \in \underline{M}$, and form a new Σ-structure N as follows:

(1) $\underline{N} = \underline{M} \cup \{n\}$, where $n \notin \underline{M}$ is a new element, and define:

(2) for all function symbols f : $[\![f]\!]^N(\widehat{m}) \stackrel{\text{def}}{=} [\![f]\!]^M(\widehat{m})$, when all $\widehat{m} \in \underline{M}$, and $[\![f]\!]^N(...n...) \stackrel{\text{def}}{=} [\![f]\!]^M(...m...)$

(3) for all relation symbols R, let $\langle ...n... \rangle \in [\![R]\!]^N \Leftrightarrow \langle ...m... \rangle \in [\![R]\!]^M$, in particular $\langle n, m_i \rangle \in [\![\sim]\!]^N \Leftrightarrow \langle m, m_i \rangle \in [\![\sim]\!]^M$, and $\langle m, n \rangle \in [\![\sim]\!]^N$ (since $\langle m, m \rangle \in [\![\sim]\!]^M$)

Lemma 11.4 Let $v_m, v_n : \mathcal{V} \to \underline{N}$ be arbitrary assignments which are identical except that, for some $x : v_m(x) = m \Leftrightarrow v_n(x) = n$. For any Σ-formula A, we have $[\![A]\!]^N_{v_n} = [\![A]\!]^M_{v_m}$.

Proof. By induction on the complexity of A. The basis case follows trivially from the construction of N (point 3.), and induction passes trivially through the propositional connectives. For $A = \exists x B$, we have $[\![A]\!]^N_{v_n} = \mathbf{1} \Leftrightarrow [\![B]\!]^N_{v_n[x\mapsto b]} = \mathbf{1}$ for some $b \in \underline{N}$. If $b \neq n$, this means that $v_n[x \mapsto b]$ and $v_m[x \mapsto b]$ satisfy the conditions of the lemma, and so by IH, we get $[\![B]\!]^N_{v_n[x\mapsto b]} = \mathbf{1} \Leftrightarrow [\![B]\!]^M_{v_m[x\mapsto b]} = \mathbf{1} \Leftrightarrow [\![A]\!]^M_{v_m} = \mathbf{1}$. If $b = n$ then, similarly, $v_n[x \mapsto n]$ and $v_m[x \mapsto m]$ satisfy the condition and IH yields the conclusion. **QED** (11.4)

Entirely analogous, but even simpler, induction shows (Exercise 11.1)

Lemma 11.5 For any assignment $v : \mathcal{V} \to \underline{M} \subset \underline{N}$, and any formula A we have $[\![A]\!]^M_v = [\![A]\!]^N_v$.

Corollary 11.6 For any formula $A : M \models A \Leftrightarrow N \models A$ (in particular, $N \models \Gamma$).

Proof. Let $v : \mathcal{V} \to \underline{M}$, $w : \mathcal{V} \to \underline{N}$ range over arbitrary assignments. Then: $M \models A \Leftrightarrow$ for all $v : [\![A]\!]^M_v = \mathbf{1} \stackrel{11.5}{\Longleftrightarrow}$ for all $v : [\![A]\!]^N_v = \mathbf{1} \stackrel{11.4}{\Longleftrightarrow}$ for all $w : [\![A]\!]^N_w = \mathbf{1} \Leftrightarrow N \models A$. **QED** (11.6)

Summarizing: starting with a theory Γ and its model M in which a non-logical symbol $\sim$ was interpreted as identity, we constructed another model $N \models \Gamma$, but where $[\![\sim]\!]^N$ is not identity since $m \neq n$ while $\langle m, n \rangle \in [\![\sim]\!]^N$.

1.1: Axioms for Identity

Although we cannot force any relation symbol to be interpreted as identity we may, nevertheless, reason about structures which happen to interpret some relation symbol as identity. Fixing the interpretation of $\equiv$, definition (11.2) forces the validity of a number of formulae, such as

$x \equiv y \to (R(f(x)) \to R(f(y)))$, that would not come out as valid if $\equiv$ were just another binary relation symbol. To obtain a complete proof system we need more axiom schemata. We extend the system $\vdash_{\mathcal{N}}$ for FOLwith the following:

A5 $\Gamma \vdash_{\mathcal{N}} \forall x : x \equiv x$
A6 $\Gamma \vdash_{\mathcal{N}} \forall x \forall y : x \equiv y \to y \equiv x$
A7 $\Gamma \vdash_{\mathcal{N}} \forall x \forall y \forall z : (x \equiv y \wedge y \equiv z) \to x \equiv z$
A8 $\Gamma \vdash_{\mathcal{N}} (s_1 \equiv t_1 \wedge \ldots \wedge s_n \equiv t_n) \;\to\; f(s_1, \ldots, s_n) \equiv f(t_1, \ldots, t_n)$
for all terms s_i and t_i and function symbols f
A9 $\Gamma \vdash_{\mathcal{N}} (s_1 \equiv t_1 \wedge \ldots \wedge s_n \equiv t_n \wedge R(s_1, \ldots, s_n)) \;\to\; R(t_1, \ldots, t_n)$
for all terms s_i and t_i and relation symbols R

The system FOL extended with (11.1), (11.2) and the above axioms will be denoted $\mathsf{FOL}^{=}$ and the provability relation by $\vdash_{\mathcal{N}^=}$. We will show that these axioms are sufficient to reason about identity. However, becausee identity is not axiomatizable, remark 11.3 implies that the obtained results will apply not only to identity but also to some other relations – namely those satisfying the above axioms. The first three axioms make $\equiv$ an equivalence relation. The last two make an equivalence into a *congruence* relation – notice that they are relative to a given Σ, since they involve explicit reference to the terms. Essentially, they make congruent terms "indistinguishable" (as we did with m and n in remark 11.3): applying a function f to two congruent (indistinguishable) terms $s \equiv t$ must yield indistinguishable results $f(s) \equiv f(t)$ – axiom A8. Similarly, if a formula (predicate) is true about s, it must be true about t (and vice versa, since $\equiv$ is symmetric).

One of the main means simplifying the reasoning with identity is the derived rule admitting "replacement of equals by equals"

$$\text{(REP)} \quad \frac{\Gamma \vdash_{\mathcal{N}^=} A^x_s \quad ; \quad \Gamma \vdash_{\mathcal{N}^=} s \equiv t}{\Gamma \vdash_{\mathcal{N}^=} A^x_t} \qquad \text{all substitutions are legal}$$

Another very common rule is *substitution* or "specialization of universal quantifiers"

$$\text{(SUB)} \quad \frac{\Gamma \vdash_{\mathcal{N}^=} \forall x A}{\Gamma \vdash A^x_t} \qquad A^x_t \text{ is legal}$$

These rules are easily derivable, the second one using lemma 7.25.1 and 7.25.5 (i.e., it is admissible already in $\vdash_{\mathcal{N}}$), and the first one using also some of the additional axioms for identity (Exercise 11.2).

1.2: Some examples

Most classical theories of mathematics are axiomatized in FOL with identity. We have seen (in chapter 5, subsection 4.3) the example of equational theory of Boolean algebras. Here we give two more examples.

Example 11.7
Groups are structures over the signature Σ_G with $\mathcal{I} = \{e\}$, $\mathcal{F}_1 = \{\_^-\}$, $\mathcal{F}_2 = \{\_ \cdot \_\}$ and satisfying the axioms

$$\begin{array}{lll}
\text{G1} & \forall x, y, z : & x \cdot (y \cdot z) \equiv (x \cdot y) \cdot z \\
\text{G2} & \forall x : & x \cdot e \equiv x \\
 & \forall x : & e \cdot x \equiv x \\
\text{G3} & \forall x : & x \cdot x^- \equiv e \\
 & \forall x : & x^- \cdot x \equiv e
\end{array}$$

Integers with addition for $\cdot$, 0 for e and unary minus is an example of a group, so are the real numbers (without 0) with multiplication for $\cdot$, 1 for e, and $1/x$ for x^-.

The axioms imply all the properties of the groups. For instance, we show that e is its own inverse, i.e., $e \equiv e^-$:

1. $\forall x : x \cdot x^- \equiv x$ $G3$
2. $e \cdot e^- \equiv e$ (SUB) 1.
3. $\forall x : e \cdot x \equiv x$ $G2$
4. $e \cdot e^- \equiv e^-$ (SUB) 3.
5. $e \equiv e \cdot e^-$ $0. + A6 + (SUB)$
6. $e \equiv e^-$ $3., 2. + A7 + (SUB)$

The last two lines in the above proof are actually standard abbreviations where the idenity axioms are applied implicitly. Applications of the axioms A6, A7 are (implicitly) preceded by applications of the substitution-rule (SUB). Written in full $\vdash_{\mathcal{N}^=}$, the proof after line 4. would look as follows:

5. $\forall xy : x \equiv y \to y \equiv x$ $A6$
6. $e \cdot e^- \equiv e \to e \equiv e \cdot e^-$ (SUB) 5.
7. $e \equiv e \cdot e^-$ $MP(2, 6)$
8. $\forall xyz : x \equiv y \land y \equiv z \to z \equiv z$ $A7$
9. $e \equiv e \cdot e^- \land e \cdot e^- \equiv e^- \to e \equiv e^-$ (SUB) 8.
10. $e \equiv e^-$ $MP(7 + 4, 9)$

Other properties, like uniqueness of the identity element e (i.e., $\forall z : (\forall x : x \cdot x^- \equiv z \land x^- \cdot x \equiv z) \to z = e$) follow by more elaborate proofs. In any case, when working with identity, it is usual to allow for abbreviated proofs (like the first proof above), where transitions due to the identity axioms are taken implicitly for granted.

The axioms are also sufficient to *prove* the fact which is assumed by the definition of the alphabet, namely that $\_^-$ is a function (i.e., $\forall x, x_1, x_2 : x \cdot x_1 \equiv e \land x_1 \cdot x \equiv e \land x \cdot x_2 \equiv e \land x_2 \cdot x \equiv e \to x_1 \equiv x_2$). $\square$

Example 11.8
An important example of a theory in $\mathsf{FOL}^=$ is the theory RCF of *real closed fields.* It has the alphabet $\{0, 1, +, \cdot\}$, where the two former are constants and the two latter are binary function symbols, and it contains the infinite set of axioms below. ($x \not\equiv y$ abbreviates $\neg\, x \equiv y$ and y^2 is an abbreviation for $y \cdot y$. In general y^n is $y \cdot y \cdot \ldots \cdot y$, with n occurrences of y.)

RCF1 $\forall x \forall y\ x + y \equiv y + x$
RCF2 $\forall x \forall y\ x \cdot y \equiv y \cdot x$
RCF3 $\forall x \forall y \forall z\ x + (y + z) \equiv (x + y) + z$
RCF4 $\forall x \forall y \forall z\ x \cdot (y \cdot z) \equiv (x \cdot y) \cdot z$
RCF5 $\forall x \forall y \forall z\ x \cdot (y + z) \equiv (x \cdot y) + (x \cdot z)$
RCF6 $\forall x\ x + 0 \equiv x$
RCF7 $0 \not\equiv 1$
RCF8 $\forall x\ x \cdot 1 \equiv x$
RCF9 $\forall x \exists y\ x + y \equiv 0$
RCF10 $\forall x \forall y\ (x \cdot y \equiv 0 \to x \equiv 0 \vee y \ \equiv 0)$
RCF11 $\forall x \forall y \forall z (x^2 + y^2 + z^2 \equiv 0 \to x \equiv 0 \wedge y \equiv 0 \wedge z \equiv 0)$
RCF12 $\forall x \exists y (y^2 \equiv x \vee y^2 + x \equiv 0)$
RCF13$_n$ $\forall x_0 \forall x_1 \ldots \forall x_n (x_n \not\equiv 0 \to \exists y\ x_n \cdot y^n + x_{n-1} \cdot y^{n-1} + \ldots + x_1 \cdot y + x_0 = 0)$ for any odd number n

Note that these axioms are all true of the real numbers. In fact, it can be shown that RCF in a sense captures everything that can be said about the real number system in the language of $\mathsf{FOL}^=$: let $\mathbb{R}$ be the $\{0, 1, +, \cdot\}$-model with the set of real numbers as interpretation domain, with 0 and 1 interpreted as zero and one respectively, and with + and $\cdot$ interpreted as addition and multiplication. Then for any sentence A over the given alphabet, $RCF \vdash_{\mathcal{N}^=} A$ is provable in $\mathsf{FOL}^=$ iff $\mathbb{R} \models A$. □

1.3: Soundness and Completeness of $\mathsf{FOL}^=$

Theorem 11.9 Let Γ be a set of $\mathsf{FOL}^=$-formulae, and let A be a formula of $\mathsf{FOL}^=$. Then
(1) $\Gamma \vdash_{\mathcal{N}^=} A$ iff $\Gamma \models A$.
(2) Γ is consistent iff it is satisfiable.

Proof. We only give a rough outline of the proof. As before, 1. and 2. are easily derived from each other. The soundness part is proved in the same way as it was for ordinary FOL. (The additional axioms A5–A9 are easily seen to be valid.)

Now let ID be the set of all instances of A5–A9; note that $\Gamma \vdash_{\mathcal{N}^=} A$ is provable in $\mathsf{FOL}^=$ iff $\Gamma \cup ID \vdash_{\mathcal{N}} A$ is provable in ordinary FOL – this is merely a matter of whether A5–A9 are presented as explicit assumptions, or axioms of the underlying logic.

To prove completeness, suppose Γ is consistent in $\mathsf{FOL}^=$. From the above remarks it follows that $\Gamma \cup ID$ is consistent in ordinary FOL, where now $\equiv$ is just another binary relation symbol. By the completeness theorem of ordinary FOL, $\Gamma \cup ID$ has a model M.

The proof does not end here, however, since ordinary FOL puts no restrictions on $[\![\equiv]\!]^M$ beyond that of being a subset of $\underline{M} \times \underline{M}$ satisfyings some axioms. Let us write $\equiv^M$ for $[\![\equiv]\!]^M$. We would like $\equiv^M$ to be the identity relation $id_{\underline{M}}$ on $\underline{M}$, for in that case the proof would be complete. There is no guarantee that this is the case, but since $M \models ID$ we do know that $\equiv^M$ is a *congruence relation* on $\underline{M}$, i.e.,

- $\equiv^M$ is an equivalence relation on $\underline{M}$,
- if $\underline{a}_1 \equiv^M \underline{b}_1$ and ... and $\underline{a}_n \equiv^M \underline{b}_n$, then $[\![f]\!]^M(\underline{a}_1, \ldots, \underline{a}_n) \equiv^M [\![f]\!]^M(\underline{b}_1, \ldots, \underline{b}_n)$,
- if $\underline{a}_1 \equiv^M \underline{b}_1$ and ... and $\underline{a}_n \equiv^M \underline{b}_n$ and $\langle \underline{a}_1, \ldots, \underline{a}_n \rangle \in [\![R]\!]^M$ then $\langle \underline{b}_1, \ldots, \underline{b}_n \rangle \in [\![R]\!]^M$.

Now define the *quotient structure* $M/{\equiv^M}$, such that $\underline{M/{\equiv^M}} = \{[\underline{a}] : \underline{a} \in \underline{M}\}$, i.e. the set of equivalence classes of $\equiv^M$ (for each $\underline{a} \in \underline{M} : [\underline{a}] = \{\underline{b} \in \underline{M} : \underline{b} \equiv^M \underline{a}\}$.) Moreover,

- $[\![c]\!]^{M/\equiv^M} = [[\![c]\!]^M]$ for any constant c,
- $[\![f]\!]^{M/\equiv^M}([\underline{a}_1], \ldots, [\underline{a}_n]) = [[\![f]\!]^M(\underline{a}_1, \ldots, \underline{a}_n)]$ for any function symbol f,
- $[\![R]\!]^{M/\equiv^M} = \{\langle [\underline{a}_1], \ldots, [\underline{a}_n] \rangle : \langle \underline{a}_1, \ldots, \underline{a}_n \rangle \in [\![R]\!]^M\}$ for any relation symbol R.

In particular, $[\![\equiv]\!]^{M/\equiv^M} = id_{\underline{M/\equiv^M}} = \{\langle [m], [m] \rangle : [m] \in \underline{M/{\equiv^M}}\}$. It remains to show that

$$M/{\equiv^M} \models \Gamma. \tag{11.10}$$

This is rather cumbersome, but we write this proof for the sake of completeness.

Since we do not know anything particular about the axioms in Γ, we show a stronger statement, namely, that M and $M/{\equiv^M}$ are *logically indistinguishable*, i.e.:

$$\text{for any formula } A : M \models A \Leftrightarrow M/{\equiv^M} \models A. \tag{11.11}$$

This obviously implies (11.10). To show (11.11), we first observe that forming the congruence classes can be viewed as a function $[\_] : \underline{M} \to \underline{M/{\equiv^M}}$. For any assignment $v : X \to \underline{M}$, resp., $[v] : X \to \underline{M/{\equiv^M}}$, we get the following diagram, satisfying points i), ii) below:

$$\begin{tikzcd} & X \arrow[dl, "v"'] \arrow[dr, "{[v]}"] & \\ \underline{M} \arrow[rr, "{[\_]}"] & & \underline{M/{\equiv^M}} \end{tikzcd}$$

i) for any assignment $[v] : X \to \underline{M/{\equiv^M}}$, there is an assignment $v : X \to \underline{M} : [v](x) = [v(x)]$,
ii) for any assignment $v : X \to \underline{M}$, there is an assignment $[v] : X \to \underline{M/{\equiv^M}} : [v(x)] = [v](x)$.

To establish (11.11), we show first that

for any $v, [v]$ satisfying $[v](x) = [v(x)]$ and any term $t : [[\![t]\!]^M_v] = [\![t]\!]^{M/\equiv^M}_{[v]}$, (11.12)

by induction on the complexity of terms:

$x \in \mathcal{V}$ :: By definition 8.2 and assumption: $[[\![x]\!]^M_v] = [v(x)] = [v](x) = [\![x]\!]^{M/\equiv^M}_{[v]}$.

$c \in \mathcal{I}$ :: By definition of $M/{\equiv^M}$: $[[\![c]\!]^M] = [\![c]\!]^{M/\equiv^M}$.

$f(t_1...t_n)$ ::

$$\begin{aligned}
[[\![f(t_1...t_n)]\!]^M_v] &= [[\![f]\!]^M([\![t_1]\!]^M_v...[\![t_n]\!]^M_v)] && \text{by definition of } [\![f]\!]^M \\
&= [\![f]\!]^{M/\equiv^M}([[\![t_1]\!]^M_v]...[[\![t_n]\!]^M_v]) && \text{by definition of } [_] \\
&= [\![f]\!]^{M/\equiv^M}([\![t_1]\!]^M_{[v]}...[\![t_n]\!]^M_{[v]}) && \text{by IH} \\
&= [\![f(t_1...t_n)]\!]^{M/\equiv^M}_{[v]} && \text{by definition of } [\![f]\!]^{M/\equiv^M}
\end{aligned}$$

The claim (11.11) will be now obtained from a stronger statement, namely:

for any $v, [v]$ satisfying $[v(x)] = [v](x)$ and any formula $A : [\![A]\!]^M_v = [\![A]\!]^{M/\equiv^M}_{[v]}$ (11.13)

We show it by induction on the complexity of formulae.

<u>A IS :</u>

$s \equiv t$ :: By (11.12) we get the equalities $[\![s]\!]^{M/\equiv^M}_{[v]} = [[\![s]\!]^M_v]$ and $[[\![t]\!]^M_v] = [\![t]\!]^{M/\equiv^M}_{[v]}$. By definition of $M/{\equiv^M}$, $[[\![s]\!]^M_v] = [[\![t]\!]^M_v] \Leftrightarrow [\![s]\!]^M_v \equiv^M [\![t]\!]^M_v$ which yields the claim.

$R(t_1...t_n)$ ::

$$\langle [\![t_1]\!]^M_v ... [\![t_n]\!]^M_v \rangle \in [\![R]\!]^M \Rightarrow \langle [[\![t_1]\!]^M_v] ... [[\![t_n]\!]^M_v] \rangle \in [\![R]\!]^{M/\equiv^M} \quad \text{def. of } M/\equiv^M$$

$$\Rightarrow \langle [\![t_1]\!]^{M/\equiv^M}_{[v]} ... [\![t_n]\!]^{M/\equiv^M}_{[v]} \rangle \in [\![R]\!]^{M/\equiv^M} \quad (11.12)$$

On the other hand,

$\langle [\![t_1]\!]^M_v ... [\![t_n]\!]^M_v \rangle \notin [\![R]\!]^M \Rightarrow$

$$\text{for any } [\![s_i]\!]^M_v \equiv^M [\![t_i]\!]^M_v : \langle [\![s_1]\!]^M_v ... [\![s_n]\!]^M_v \rangle \notin [\![R]\!]^M \quad \text{congr. of } \equiv^M$$

$$\Rightarrow \langle [[\![t_1]\!]^M_v] ... [[\![t_n]\!]^M_v] \rangle \notin [\![R]\!]^{M/\equiv^M} \quad \text{def. of } M/\equiv^M$$

$$\Rightarrow \langle [\![t_1]\!]^{M/\equiv^M}_{[v]} ... [\![t_n]\!]^{M/\equiv^M}_{[v]} \rangle \notin [\![R]\!]^{M/\equiv^M} \quad (11.12)$$

$\neg B$ :: $[\![\neg B]\!]^M_v = \mathbf{1} \Leftrightarrow [\![B]\!]^M_v = \mathbf{0} \overset{\text{IH}}{\Leftrightarrow} [\![B]\!]^{M/\equiv^M}_{[v]} = \mathbf{0} \Leftrightarrow [\![\neg B]\!]^{M/\equiv^M}_{[v]} = \mathbf{1}$.

Induction passes equally trivially through the case $B \to C$

$\exists x B$ :: $[\![\exists x B]\!]^M_v = \mathbf{1} \Leftrightarrow$ for some $m \in \underline{M}$: $[\![B]\!]^M_{v[x \mapsto m]} = \mathbf{1} \overset{\text{IH}}{\Leftrightarrow} [\![B]\!]^{M/\equiv^M}_{[v][x \mapsto [m]]} = \mathbf{1} \Leftrightarrow [\![\exists x B]\!]^{M/\equiv^M}_{[v]} = \mathbf{1}$.

(11.11) follows: $M \models A \Leftrightarrow$ for all v : $[\![A]\!]^M_v = \mathbf{1} \overset{(11.13)}{\Leftrightarrow}$ for all $[v]$: $[\![A]\!]^{M/\equiv^M}_{[v]} = \mathbf{1} \Leftrightarrow M/\equiv^M \models A$. **QED** (11.9)

Remark 11.14 **[Logical indistinguishability of identity and congruence]**

Thus, axioms ID for identity (A5-A9 from subsection 1.1) allow us to deduce all statements about identity following logically from some theory Γ. But observe that they work equally well for arbitrary congruence relations. As we have said in the above proof, taking $\Gamma \cup ID$ and using $\vdash_{\mathcal{N}}$, we can deduce all logical conseuences of this theory (by completeness of $\vdash_{\mathcal{N}}$), i.e., ones which hold in arbitrary model of $\Gamma \cup ID$. Obviously, these two sets are identical $\{A : \Gamma \vdash_{\mathcal{N}^=} A\} = \{A : \Gamma \cup ID \vdash_{\mathcal{N}} A\}$! This means that, logically, the identity relation is *indistinguishable* from congruence relation – there is no formula which will hold about the one but not about the other.

This is a further aspect of non-axiomatizability of identity which we pointed out in remark 11.3. In terms of model classes, the proof of completeness we have given above says that:

- Any model of Γ where $\equiv$ is interpreted as identity, i.e., any $\mathsf{FOL}^=$ model of Γ is also a standard FOL model of $\Gamma \cup ID$. I.e. $Mod(\Gamma \cup ID) \models A \Rightarrow Mod^=(\Gamma) \models A$.
- The quotient construction shows the opposite – any FOL model of $\Gamma \cup ID$ can be transformed into a *logically equivalent* $\mathsf{FOL}^=$ model of Γ. I.e. $Mod^=(\Gamma) \models A \Rightarrow Mod(\Gamma \cup ID) \models A$.

Thus, although $Mod^=(\Gamma) \subset Mod(\Gamma \cup ID)$ – and the subset relation is proper! –

the two classes are logically indistinguishable, i.e., $\{A : Mod^{=}(\Gamma) \models A\} = \{A : Mod(\Gamma \cup ID) \models A\}$.

2: A few more bits of Model Theory

2.1: Compactness

In Theorem 4.29 we have shown that a theory Γ is consistent iff each finite subtheory of Γ is. This result (and the notion of consistency) depends on the proof system. The completeness theorem 10.17 allows us to express this fact in a way independent from the proof system.

Theorem 11.15 A FOL theory Γ has a model iff each finite subtheory Δ of Γ has a model.

Proof. $Mod(\Gamma) \neq \varnothing \overset{10.17}{\iff} \Gamma \not\vdash_{\mathcal{N}} \bot \overset{4.29}{\iff}$ for all $\Delta : \Delta \not\vdash_{\mathcal{N}} \bot \overset{10.17}{\iff}$ for all $\Delta : Mod(\Delta) \neq \varnothing$ **QED** (11.15)

Using theorem 11.9 rather than 10.17, we may obtain the analogous result for $\mathsf{FOL}^{=}$.

An interesting consequence of the compactness theorem for $\mathsf{FOL}^{=}$ is that the concept of *finitude* is not axiomatizable in $\mathsf{FOL}^{=}$. 'Finitude' here means that something is finite but we do not care how may elements it contains. Of course, we can force all models of a theory to have, say, exactly one element by adding an axiom $\forall x \forall y(x \equiv y)$. Similarly, we can force all models to have exactly two elements by an axiom $\exists x_1 \exists x_2(x_1 \not\equiv x_2 \wedge \forall y(y \equiv x_1 \vee y \equiv x_2))$. But we have no means to ensure that all models are finite with unspecified cardinality, i.e., that a theory has models with *arbitrary large but finite* cardinalities.

Theorem 11.16 If a $\mathsf{FOL}^{=}$ theory Γ has arbitrarily large finite models, then it has also infinite model.

Proof. We first consider special kinds of formulae which are satisfied by structures having at least some prescribed, finite cardinality.

- $F2 :\ \exists x_1 \exists x_2 : x_1 \not\equiv x_2$ – is satisfied by structures having at least two elements
- $F3 :\ \exists x_1, x_2, x_3 : x_1 \not\equiv x_2 \wedge x_1 \not\equiv x_3 \wedge x_2 \not\equiv x_3$ – is satisfied by structures having at least three elements
- $Fn :\ \exists x_1, x_2 \ldots x_n : x_1 \not\equiv x_2 \ldots \wedge x_1 \not\equiv x_n \wedge x_2 \not\equiv x_3 \ldots \wedge x_2 \not\equiv x_n \ldots \wedge x_{n-1} \not\equiv x_n$ – is satisfied by structures having at least n

elements. This formula can also be written as

$$Fn = \quad \exists x_1, x_2 \ldots x_n \bigwedge_{1 \leq i < j \leq n} x_i \not\equiv x_j$$

where $\bigwedge$ denotes the conjunction of all the inequalities.

Define $\Gamma_n \stackrel{\text{def}}{=} \Gamma \cup \{Fn\}$, $F_\omega \stackrel{\text{def}}{=} \{Fn : n \in \mathbb{N}\}$, and $\Gamma_\omega \stackrel{\text{def}}{=} \Gamma \cup \{F_\omega\}$.
If Γ has arbitrary large finite models, i.e., for any $1 < n \in \mathbb{N}$, Γ has a model M_n containing at least n elements, then each such $M_n \models \Gamma_n$. By the semantic compactness theorem 11.15, this means that also Γ_ω has a model – since any finite subtheory of Γ_ω is contained in some Γ_n, and each Γ_n has a model M_n. But the model of Γ_ω has to satisfy F_ω, so it has infinitely many elements. Thus $\Gamma \subset \Gamma_\omega$ has an inifinite model. **QED** (11.16)

2.2: Skolem-Löwenheim

Theorem 11.17 Let Σ be a countable alphabet and let A be a closed formula over Σ. If A is satisfiable then there is a countable model of A.

Proof. You have shown this theorem in exercise 10.8. If A is consistent, then by the completeness theorem it is satisfiable. In the proof of this theorem it was shown that in this case A has a term model over a countable alphabet. Since there are only countably many ground terms over a countable alphabet, such a model will be countable. **QED** (11.17)

The result above was proved for ordinary FOL, but is easily extended to $\mathsf{FOL}^=$. The point is that the model $M/\equiv^M$ that was constructed from M in the proof of theorem 11.9 has a domain with cardinality no larger than the cardinality of $\underline{M}$.

One striking consequence of this latter version of the theorem concerns the theory RCF used to describe the real numbers. Since RCF has a model, it also has a countable model. But the set of real numbers itself is not countable, so this model is different from the "intended" one in a very important respect. Note carefully that this inability to "say all there is to be said" about the real numbers is a weakness not only of RCF but of FOL itself: everything said about the real numbers within FOL would be true also in some countable model.

3: Semi-Decidability and Undecidability of FOL

Theorem 11.18 The relation $\Gamma \models A$ for FOL over a countable alphabet is semi-decidable.

Proof. We only sketch the main idea without considering the details. By soundness and completeness, $\Gamma \models A$ holds iff $\Gamma \vdash_{\mathcal{N}} A$ holds, so we may check the former while in fact checking the latter: let Σ be countable; we can then enumerate all well-formed formulae over Σ and also all finite lists of such formulae. Now recall that a proof is just a finite list of formulae with some further properties. Hence the (infinite) list of all finite lists of formulae will contain every possible proof; the next step is to filter out those elements that are not proofs. This is an easy matter that can be carried out by a simple proof-checking machine which given any finite list $A_1, \ldots, A_n$ of formulae, for each $i \leq n$ will determine if the formula A_i is a logical axiom, belongs to Γ or else follows from the earlier formulae A_j for $j < i$, by an application of some proof rule. (Some stringent bookkeeping would be needed here.) Thus we can enumerate all possible proofs from Γ.
It only remains to look through this whole list to see if any of these proofs is in fact a proof of $\Gamma \vdash_{\mathcal{N}} A$, i.e., has A as the final formula. If $\Gamma \vdash_{\mathcal{N}} A$ is provable we will eventually discover this, otherwise we will keep on looking forever. **QED** (11.18)

Theorem 11.19 The relation $\Gamma \models A$ is undecidable.

Proof. You are asked to show this in exercise 11.4 by showing that if this relation were decidable, then the Halting Problem 3.16 would be decidable, thus contradicting theorem 3.17. **QED** (11.19)

4: Why is First-Order Logic "First-Order"?

The logic that we have called "predicate logic" and denoted FOL (with or without identity) is often called "first-order predicate logic" or just "first-order logic." What is the reason for this name?

Answer
Variables range *only over individual elements from the interpretation*

domain. In other words, if $A \in \mathsf{WFF}_{\mathsf{FOL}}$ then $\exists x A \in \mathsf{WFF}_{\mathsf{FOL}}$ only when *x is an individual variable $x \in \mathcal{V}$*.

Consider the formula $\forall x\ P(x) \to P(x)$. Obviously, no matter which unary predicate $P \in \mathcal{R}$ we choose, this formula will be valid. Thus, we might be tempted to write $\forall P\ \forall x\ P(x) \to P(x)$, i.e., try to quantify a variable P which ranges over predicates. This would not be a first-order – but second-order – formula. Second-order logic increases tremendously the expressive power and allows us to "describe" structures which are not fully "describable" in first-order logic. The most obvious example of such a structure are the natural numbers $\mathbb{N}$.

Example 11.20

The following definitions and axioms for $\mathbb{N}$ were introduced by Italian mathematician Giuseppe Peano in the years 1894-1908.
The language of natural numbers is given by the terms $\mathcal{T}_{\mathbb{N}}$ over the alphabet $\Sigma_{\mathbb{N}} = \{0, s\}$. We are trying here to define the property "*n is a natural number*", so we write $\mathbb{N}(n)$:

1: $\mathbb{N}(0)$
0 is a natural number.

2: $\mathbb{N}(n) \to \mathbb{N}(s(n))$
s(uccessor) of a natural number is a natural number.

We want all these terms to denote different numbers, for instance, we should not consider the structures satisfying $s(0) \equiv 0$. Thus we have to add the axioms:

3: $s(n) \not\equiv 0$
0 is not a successor of any natural number.

4: $s(n) \equiv s(m) \to n \equiv m$
If two numbers have the same successor then they are equal – the s function is injective.

This is fine but it does not guarantee that the models we obtain will contain *only* elements interpreting the ground terms: any model will have to contain distinct elements interpreting $\{0, s(0), s(s(0))...\}$ but we also obtain unreachable models with additional elements! We would like to pick only the reachable models, but reachability itself is a semantic notion which is not expressible in the syntax of FOL. In order to ensure that no such "junk" elements appear in any model it is sufficient to require that

5: For any property P, if
- $P(0)$, and
- $\forall n\ (P(n) \to P(s(n)))$

then $\forall n\ \mathbb{N}(n) \to P(n)$.

This last axiom is actually the induction property – if 0 satisfies P and whenever an n satisfies P then its successor satisfies P, then all natural numbers satisfy P. In other words, what is true about all terms generated by 1. and 2. is true about all natural numbers, because there are no more natural numbers than what can be obtained by applying s an arbitrary number of times to 0.

It can be shown that these five axioms actually describe the natural numbers up to isomorphism. (Any model of the five axioms is isomorphic to the $\{0, s, \mathbb{N}\}$-model with the natural numbers as interpretation domain and interpretation of $\mathbb{N}$, and with 0 and s interpreted as zero and the successor function.) Unfortunately, the fifth axiom is not first-order. Written as a single formula it reads:

$$\forall P\ \big(\big(P(0) \wedge \forall n\ (P(n) \to P(s(n)))\big) \to\ \forall n\ (\mathbb{N}(n) \to P(n))\big)$$

i.e., we have to introduce the quantification over predicates, thus moving to second-order logic. □

It can also be shown that the fifth axiom is necessary to obtain the natural numbers. In an exercise we have shown by induction that the binary + operation defined in example 2.27, is commutative. This is certainly something we would expect for the natural numbers. However, if we take only the first four axioms above, and define + as in 2.27, we won't be able to prove commutativity. In exercise 11.3 you are asked to design a structure satisfying these four axioms where + is not commutative

One of the most dramatic consequences of moving to second-order logic was pointed out by Kurt Gödel in 1931, to a big surprise of everybody, and in particular, of the so-called formalists and logical positivists. We only quote a reformulated version of the famous

Theorem 11.21 [Gödel's incompleteness theorem] There is no axiomatic system which is both sound and complete for second-order logic.

A related result says that no decidable set of axioms can do for the natural numbers what RCF does for the reals. (By a decidable set of axioms we mean a set of formulae for which there exists a decisions algorithm – a Turing machine that given any formula eventually halts with a correct verdict YES or NO regarding membership. Note that RCF, although infinite, is clearly decidable in this sense.) There is no decidable set X of axioms such that for every $\{0, 1, +, \cdot\}$-sentence A, $X \vdash_{\mathcal{N}^=} A$ is provable in $\mathsf{FOL}^=$ iff $\mathbb{N} \models A$.

We can thus set up the following comparison between the three logical systems (SOL stands for Second Order Logic) with respect to their provability relations:

$\vdash$	sound	complete	decidable
SOL	+	−	−
FOL	+	+	−
PL	+	+	+

Moving upwards, i.e., increasing the expressive power of the logical system, we lose some desirable properties. This is a common phenomenon that increase of expressive power implies increased complexity where nice and clean properties may no longer hold. Although SOL allows us to axiomatize natural numbers, the price we pay for this is that we no longer can prove all valid formulae. The choice of adequate logical system for a given application involves often the need to strike a balance between the expressive power, on the one hand, and other desirable properties, on the other.

Exercises 11.

EXERCISE 11.1 Prove lemma 11.5.

EXERCISE 11.2 Show admissibility of the rule (REP) in FOL with identity. (Hint: One may try to proceed by induction on the complexity of the formula A. However, given theorem 11.9, one may avoid proof-theoretic argument and, instead, reach the conclusion verifying that the rule is sound!

(1) $\Gamma \vdash s \equiv t$ is one of the assumptions of the rule
(2) by theorem 11.9, $\Gamma \models s \equiv t$
(3) by lemma 8.9, we may conclude that $\Gamma \models A^x_s \leftrightarrow A^x_t$

Complete this argument and fill in the details.)

EXERCISE 11.3 Recalling Example 11.20, convince yourself that the first four axioms are not sufficient to characterize the natural numbers by designing a structure (which does not satisfy the induction axiom 5.) where the + operation defined in Example 2.27 is not commutative.

<u>EXERCISE 11.4</u> Recall (from chapter 3, subsection 2.2) the representation of a TM as a set of transitions between situations. We will use this representation to encode a TM as a FOL theory. Without loss of generality, we will assume that the TM works on a binary alphabet $\Sigma = \{\#, 1\}$.
For each state q of TM, let R_q be a new relation symbol of arity 3, and let $R_q(l, a, r)$ mean that TM is in situation $\langle l, q, a, r\rangle$. Rewriting all the transitions as implications is a trivial task.

(1) Write the transitions of the machine M from example 3.8 as implications. Notice that $l1$, $1r$, $l\#$, resp. $\#r$ are now taken as functions on strings. The set of these implications is called the *general theory* of M.
(2) Let TM be an arbitrary machine with general theory Γ. Use exercise 3.8 to show that if at some point of computation TM is in the situation described by the premise of one of the implications from Γ, then its conclusion describes the situation of TM at the next step.
(3) Let $I = R_{q_0}(\epsilon, a, r)$ for some a and r be an initial situation, i.e., TM starts in the situation:

$$\begin{array}{c|c|c|c} \hline \cdots\ \# & a & \cdots r \cdots & \#\ \cdots \\ \hline & \uparrow & & \\ & q_0 & & \end{array}$$

Adding this statement to the general theory of TM yields an *input theory* of TM, Γ_I, for the particular initial situation I. Let us assume that TM has a distinguished halting state q_H in which it stops any terminating computation.
Show that: TM halts from the initial situation $\langle \epsilon, q_0, a, r\rangle \Leftrightarrow \Gamma_I, \neg\exists x\exists y\exists z R_{q_H}(x, y, z)$ is inconsistent. ($\Rightarrow$ is simple; for $\Leftarrow$ assume that TM does not halt and use this computation and point 2. to construct a model showing thus that the theory is consistent.)
(4) Use this result and the undecidability of the halting problem (Theorem 3.17) to show that, given an arbitrary FOL theory Γ and formula A, the question whether $\Gamma \vdash_{\mathcal{N}} A$ (or equivalently $\Gamma \models A$) is undecidable.
[Recall that we have shown undecidability of FOL in (9.40). But the result there depended on some claims which were merely assumed while here we have verified it directly.]

<u>EXERCISE 11.5</u> A *ground instance* of a formula A is a sentence obtained from A by substitution of ground terms for the free variables. Suppose M is reachable, and let A be a formula. Show that

$M \models A$ iff ($M \models B$ for all ground instances B of A).

Hint: If $x_1, \ldots, x_n$ are the free variables of A, then B is a ground instance of A iff there exist ground terms $t_1, \ldots, t_n$ such that B equals $A^{x_1,\ldots,x_n}_{t_1,\ldots,t_n}$, i.e., the result of a simultaneous substitution of each t_i for the corresponding x_i.

optional

EXERCISE 11.6 [Herbrand's theorem]
To distinguish between the versions of the proof system $\vdash_{\mathcal{N}}$ for statement logic and predicate logic (without identity) we write $\vdash^{\mathsf{PL}}_{\mathcal{N}}$ and $\vdash^{\mathsf{FOL}}_{\mathcal{N}}$. For any set Γ of predicate logic formulae over a given alphabet Σ, let $GI(\Gamma)$ be the set of ground instances of formulae in Γ, i.e., the ground formulae obtained from formulae of Γ by substitution of members of $\mathcal{GT}_\Sigma$ for the free variables. Now prove *Herbrand's Theorem:*

> Suppose Σ contains at least one constant, and let Γ be a set of quantifier free Σ-formulae. Then $\Gamma \vdash^{\mathsf{FOL}}_{\mathcal{N}} \bot$ iff $GI(\Gamma) \vdash^{\mathsf{PL}}_{\mathcal{N}} \bot$.

Hint: The direction towards the left is easiest: use the fact that everything provable in $\vdash^{\mathsf{PL}}_{\mathcal{N}}$ is also provable in $\vdash^{\mathsf{FOL}}_{\mathcal{N}}$, also use Deduction Theorem and prove that $A \vdash^{\mathsf{FOL}}_{\mathcal{N}} B$ whenever B is a ground instance of A.
For the other direction use the completeness theorem for PL to deduce the existence of a valuation model for $GI(\Gamma)$ whenever $GI(\Gamma) \not\vdash^{\mathsf{PL}}_{\mathcal{N}} \bot$ and show how to convert this into a term structure. Finally apply the result from exercise 11.5 and soundness theorem for FOL.