CHAPTER 1

Fundamentals— Concepts and Logic

Die Welt ist alles, was der Fall ist.
Ludwig Wittgenstein

"The world is everything that is the case" — this is the first tractatus in Ludwig Wittgenstein's *Tractatus Logico-Philosophicus*.

In science, we want to know what is true, i.e., what is the case, and what is not. Propositions are the theorems of our language, they are to describe or denote what is the case. If they do, they are called true, otherwise they are called false. This sounds a bit clumsy, but actually it is pretty much what our common sense tells us about true and false statements. Some examples may help to clarify things:

"This sentence contains five words"
: This proposition describes something which is the case, therefore it is a *true* statement.

"Every human being has three heads"
: Since I myself have only one head (and I assume this is the case with you as well), this proposition describes a situation which is not the case, therefore it is *false*.

In order to handle propositions precisely, science makes use of two fundamental tools of thought:

- Propositional Logic
- Architecture of Concepts

These tools aid a scientist to construct accurate concepts and to formulate new true propositions from old ones.

The following sections may appear quite diffuse to the reader; some things will seem to be obviously true, other things will perhaps not make much sense to start with. The problem is that we have to use our natural language for the task of defining things in a precise way. It is only by using these tools that we can define in a clear way what a set is, what numbers are, etc.

1.1 Propositional Logic

Propositional logic helps us to navigate in a world painted in black and white, a world in which there is only truth or falsehood, but nothing in between. It is a boiled down version of common sense reasoning. It is the essence of Sherlock Holmes' way of deducing that Professor Moriarty was the mastermind behind a criminal organization ("Elementary, my dear Watson"). Propositional logic builds on the following two propositions, which are declared to be true as basic principles (and they seem to make sense...):

Principle of contradiction (principium contradictionis)
A proposition is never true and false at the same time.

Principle of the excluded third (tertium non datur)
A proposition is either true or false—there is no third possibility.

In other words, in propositional logic we work with statements that are either true (T) or false (F), no more and no less. Such a logic is also known as *absolute* logic.

In propositional logic there are also some operations which are used to create new propositions from old ones:

Logical Negation
The negation of a true proposition is a false proposition, the negation of a false proposition is a true proposition. This operation is also called 'NOT'.

Logical Conjunction
The conjunction of two propositions is true if and only if both propositions are true. In all other cases it is false. This operation is also called 'AND'.

Logical Disjunction
The disjunction of two propositions is true if at least one of the propositions is true. If both propositions are false, the disjunction is false, too. This operation is also known as 'OR'.

Logical Implication
A proposition $\mathcal{P}_1$ implies another proposition $\mathcal{P}_2$ if $\mathcal{P}_2$ is true whenever $\mathcal{P}_1$ is true. This operation is also known as 'IMPLIES'.

Often one uses so-called truth tables to show the workings of these operations. In these tables, A stands for the possible truth values of a proposition $\mathcal{A}$, and B stands for the possible truth values of a proposition $\mathcal{B}$. The rows labeled "A AND B" and "A OR B" contain the truth value of the conjunction and disjunction of the propositions.

A	NOT A
F	T
T	F

A	B	A AND B
F	F	F
F	T	F
T	F	F
T	T	T

A	B	A OR B
F	F	F
F	T	T
T	F	T
T	T	T

A	B	A IMPLIES B
F	F	T
F	T	T
T	F	F
T	T	T

Let us look at a few examples.

1. Let proposition $\mathcal{A}$ be "The ball is red". The negation of $\mathcal{A}$, (i.e., NOT $\mathcal{A}$) is "It is not the case that the ball is red". So, if the ball is actually green, that means that $\mathcal{A}$ is false and that NOT $\mathcal{A}$ is true.
2. Let proposition $\mathcal{A}$ be "All balls are round" and proposition $\mathcal{B}$ "All balls are green". Then the conjunction $\mathcal{A}$ AND $\mathcal{B}$ of $\mathcal{A}$ and $\mathcal{B}$ is false, because there are balls that are not green.
3. Using the same propositions, the disjunction of $\mathcal{A}$ and $\mathcal{B}$, $\mathcal{A}$ OR $\mathcal{B}$ is true.
4. For any proposition $\mathcal{A}$, $\mathcal{A}$ AND (NOT $\mathcal{A}$) is always false (principle of contradiction).
5. For any proposition $\mathcal{A}$, $\mathcal{A}$ OR (NOT $\mathcal{A}$) is always true (principle of excluded third).

In practice it is cumbersome to say: "The proposition *It rains* is true". Instead, one just says: "It rains." Also, since the formal combination of propositions by the above operators is often an overhead, we mostly use the common language denotation, such as: "2 = 3 is false" instead of "NOT (2 = 3)" or: "It's raining or/and I am tired." instead of "It's raining OR/AND I am tired", or: "If it's raining, then I am tired" instead of "It's raining IMPLIES I am tired." Moreover, we use the mathematical abbreviation "*A* iff *B*" for "(*A* IMPLIES *B*) AND (*B* IMPLIES *A*)". Observe that brackets (...) are used in order to make the grouping of symbols clear if necessary.

The operations NOT, AND, OR, and IMPLIES have a number of properties which are very useful for simplifying complex combinations of these operations. Let *P*, *Q*, and *R* be truth values. Then the following properties hold:

Commutativity of AND
P AND *Q* is the same as *Q* AND *P*.

Commutativity of OR
P OR *Q* is the same as *Q* OR *P*.

Associativity of AND
(*P* AND *Q*) AND *R* is the same as *P* AND (*Q* AND *R*).
One usually omits the parentheses and writes *P* AND *Q* AND *R*.

Associativity of OR
(*P* OR *Q*) OR *R* is the same as *P* OR (*Q* OR *R*).
One usually omits the parentheses and writes *P* OR *Q* OR *R*.

De Morgan's Law for AND
NOT (*P* AND *Q*) is the same as (NOT *P*) OR (NOT *Q*).

De Morgan's Law for OR
NOT (*P* OR *Q*) is the same as (NOT *P*) AND (NOT *Q*).

Distributivity of AND over OR
P AND (*Q* OR *R*) is the same as (*P* AND *Q*) OR (*P* AND *R*).

Distributivity of OR over AND
P OR (*Q* AND *R*) is the same as (*P* OR *Q*) AND (*P* OR *R*).

Contraposition
P IMPLIES *Q* is the same as (NOT *Q*) IMPLIES (NOT *P*).

Idempotency of AND
P is the same as *P* AND *P*.

Idempotency of OR
P is the same as *P* OR *P*.

The validity of these properties can be verified by using the truth tables. We will show how this is done for the example of "Distributivity of OR over AND".

We want to show that *P* OR (*Q* AND *R*) is the same as (*P* OR *Q*) AND (*P* OR *R*), for every choice of *P*, *Q*, and *R*. To do so we first write a big truth table which shows the values for *P*, *Q*, and *R* as well as *Q* AND *R* and *P* OR (*Q* AND *R*) :

P	*Q*	*R*	*Q* AND *R*	***P* OR (*Q* AND *R*)**
F	F	F	F	**F**
F	F	T	F	**F**
F	T	F	F	**F**
F	T	T	T	**T**
T	F	F	F	**T**
T	F	T	F	**T**
T	T	F	F	**T**
T	T	T	T	**T**

Then we write a truth table which shows the values for *P*, *Q*, and *R* as well as *P* OR *Q*, *P* OR *R*, and (*P* OR *Q*) AND (*P* OR *R*):

P	*Q*	*R*	*P* OR *Q*	*P* OR *R*	**(*P* OR *Q*) AND (*P* OR *R*)**
F	F	F	F	F	**F**
F	F	T	F	T	**F**
F	T	F	T	F	**F**
F	T	T	T	T	**T**
T	F	F	T	T	**T**
T	F	T	T	T	**T**
T	T	F	T	T	**T**
T	T	T	T	T	**T**

The truth values of the two expressions we are interested in (shown in bold face) are indeed equal for every possible combination of *P*, *Q*, and *R*.

The verification of the remaining properties is left as an exercise for the reader.

1.2 Architecture of Concepts

In order to formulate unambiguous propositions, we need a way to describe the concepts we want to make statements about. An architecture of concepts deals with the question: "How does one build a concept?" Such an architecture defines ways to build new concepts from already existing concepts. Of course one has to deal with the question where to anchor the architecture, in other words, what are the basic concepts and how are they introduced. This can be achieved in two different ways. The first uses the classical approach of undefined primary concepts, the second avoids primary concepts by circular construction. This second approach is the one that is used for building the architecture of set theory in this book.

1. A concept has a *name*, for example, "Number" or "Set" are names of certain concepts.
2. Concepts have *components*, which are concepts, too.
   These components are used to construct a concept.
3. There are three fundamental principles of how to combine such components:
   - *Conceptual Selection*: requires one component
   - *Conceptual Conjunction*: requires one or two components
   - *Conceptual Disjunction*: requires two components
4. Concepts have *instances* (examples), which have the following properties:
   - Instances have a name
   - Instances have a value

The construction principles mentioned above are best described using instances:

The value of an instance of a concept constructed as a selection is the collection of the references to selected instances of the component.

The value of an instance of a concept constructed as a conjunction is the sequence of the references to the instances of each component.

The value of an instance of a concept constructed as a disjunction is a reference to an instance of one of the components.

Perhaps some examples will clarify those three construction principles.

A selection is really a selection in its common sense meaning: You point at a thing and say, "I select this", you point at another thing and say "I select this, too" and so on.

One example for a conjunction are persons' names which (at least in the western world) always consists of a first name and a family name. Another example is given by the points in the plane: every point is defined by an x- and a y-coordinate.

A disjunction is a simple kind of "addition": An instance of the disjunction of all fruits and all animals is either a fruit or an animal.

Notation

If we want to write about concepts and instances, we need an expressive and precise notation.

Concept: `ConceptName.ConstructionPrinciple(Component(s))`
: This means that we first write the concept's name followed by a dot. After the dot we write the construction principle (Selection, Conjunction, or Disjunction) used to construct the concept. Finally we add the component or components which were used for the construction enclosed in brackets.

Instance: `InstanceName@ConceptName(Value)`
: In order to write down an instance, we write the instance's name followed by an '@'. After this, the name of the concept is added, followed by a value enclosed in brackets. In the case of a disjunction, a semicolon directly following the value denotes the first component, and a semicolon preceding the value denotes the second component.

Very often it is not possible to write down the entire information needed to define a concept. In most cases one cannot write down components and values explicitly. Therefore, instead of writing the concept or instance, one only writes its name. Of course, this presupposes that these

objects can be identified by a name, i.e., there are enough names to distinguish these objects from one another. Thus if two concepts have identical names, then they have identical construction principles and identical components. The same holds for instances: identicals name mean identical concepts and identical values.

By identifying names with objects one can say "let X be a concept" or "let z be an instance", meaning that X and z are the names of such objects that refer to those objects in a unique way.

Here are some simple examples for concepts and instances:

CitrusFruits.Disjunction(Lemons, Oranges)
: The concept **CitrusFruit** consists of the concepts **Lemons** and **Oranges**.

MyLemon@Citrusfruits(Lemon2;)
: **MyLemon** is an instance of the concept **CitrusFruit**, and has the value **Lemon2** (which is itself an instance of the concept **Lemons**).

YourOrange@Citrusfruits(; Orange7)
: **YourOrange** is an instance of the concept **CitrusFruits**, and has the value **Orange7** (which is itself an instance of the concept **Oranges**).

CompleteNames.Conjunction(FirstNames, FamilyNames)
: The concept **CompleteNames** is a conjunction of the concept **FirstNames** and **FamilyNames**.

MyName@CompleteNames(John; Doe)
: **MyName** is an instance of the concept **CompleteNames**, and has the value **John; Doe**.

SmallAnimals.Selection(Animals)
: The concept **SmallAnimals** is a selection of the concept **Animals**.

SomeInsects@SmallAnimals(Ant, Ladybug, Grasshopper)
: **SomeInsects** is an instance of the concept **SmallAnimals** and has the value **Ant, Ladybug, Grasshopper**.

Mathematics

The environment in which this large variety of concepts and propositions is handled is Mathematics.

With the aid of set theory Mathematics is made conceptually precise and becomes the foundation for all formal tools. Especially formal logic is only possible on this foundation.

In Mathematics the existence of a concept means that it is conceivable without any contradiction. For instance, a set exists if it is conceivable without contradiction. Most of the useful sets exist (i.e., are conceivable without contradiction), but one may conceive sets which don't exist. An example of such a set is the subject of the famous paradox advanced by Bertrand Russell: the set containing all sets that do not contain themselves—does this set contain itself, or not?

Set theory must be constructed successively to form an edifice of concepts which is conceivable without any contradictions.

In this section we will first show how one defines natural numbers using concepts and instances. After that, we go on to create set theory from "nothing".

Naive Natural Numbers

The natural numbers can be conceptualized as follows:

Number.Disjunction(Number, Terminator)
Terminator.Conjunction(Terminator)

The concept **Number** is defined as a disjunction of itself with a concept **Terminator**, the concept **Terminator** is defined as a conjunction of itself (and nothing else). The basic idea is to define a specific natural number as the successor of another natural number. This works out for 34, which is the successor of 33, and also for 786657, which is the successor of 786656. But what about 0? The number zero is not the successor of any other natural number. So in a way we use the **Terminator** concept as a starting point, and successively define each number (apart from 0) as the successor of the preceding number. The fact that the concept **Terminator** is defined as a conjunction of itself simply means: "**Terminator** is a thing". This is a first example of a circular construction used as an artifice to ground the definition of natural numbers.

Now let us look at some instances of these concepts:

t@Terminator(t)
In natural language: the value of the instance **t** of **Terminator** is itself. This is a second application of circularity.

0@Number(; t)
The instance of **Number** which we call **0** has the value **t;**.

1@Number(0;)
The instance of **Number** which we call **1** has the value **0;**.

2@Number(1;)
The instance of **Number** which we call **2** has the value **1;**.

If we expand the values of the numbers which are neither **0** nor **t**, we get

- the value of **1** is **0;**
- the value of **2** is **1;** which is **0;;**
- the value of **3** is **2;** which is **0;;;**
- etc.

This could be interpreted by letting the semicolon stand for the operation "successor of", thus **3** is the successor of the successor of the successor of **0**.

Pure Sets

The pure sets are defined in the following circular way:

Set.Selection(Set)

Here, we say that a set is a selection of sets. Since one is allowed to select nothing in a conceptual selection, there is a starting point for this circularity. Let us look at some instances again:

∅@Set()
Here we select nothing from the concept **Set**. We therefore end up with the empty set.

***1*@Set(∅)**
Since ∅ is a set we can select it from the concept **Set**. The value of ***1*** is a set consisting of one set.

***2*@Set(∅, *1*)**

Here we select the two sets we have previously defined. The value of *2* is a set consisting of two sets.

Elements of the Mathematical Prose

In Mathematics, there is a "catechism" of true statements, which are named after their relevance in the development of the theory.

An *axiom* is a statement which is not proved to be true, but supposed to be so. In a second understanding, a theory is called *axiomatic* if its concepts are abstractions from examples which are put into generic definitions in order to develop a theory from a given type of concepts.

A *definition* is used for building—and mostly also for introducing a symbolic notation for—a concept which is described using already defined concepts and building rules.

A *lemma* is an auxiliary statement which is proved as a truth preliminary to some more important subsequent true statement. A *corollary* is a true statement which follows without significant effort from an already proved statement. Ideally, a corollary should be a straightforward consequence of a more difficult statement. A *sorite* is a true statement, which follows without significant effort from a given definition. A *proposition* is an important true statement, but less important than a *theorem*, which is the top spot in this nomenclature.

A mathematical *proof* is the logical deduction of a true statement $\mathcal{B}$ from another true statement C. Logical deduction means that the theorems of absolute logic are applied to establish the truth of $\mathcal{B}$, knowing the truth of C. The most frequent procedure is to use as the true statement C the truth of $\mathcal{A}$ and the truth of $\mathcal{A}$ IMPLIES $\mathcal{B}$, in short, the truth of $\mathcal{A}$ AND ($\mathcal{A}$ IMPLIES $\mathcal{B}$). Then $\mathcal{B}$ is true since the truth of the implication with the true antecedent $\mathcal{A}$ can only hold with $\mathcal{B}$ also being true. This is the so-called *modus ponens*. This scheme is also applied for *indirect proofs*, i.e., we use the true fact (NOT $\mathcal{B}$) IMPLIES (NOT $\mathcal{A}$), which is equivalent to $\mathcal{A}$ IMPLIES $\mathcal{B}$ (contraposition, see also properties on page 6). Now, by the principle of the excluded third and the principle of contradiction, either $\mathcal{B}$ or NOT $\mathcal{B}$ will be true, but not both at the same time. Then the truth of NOT $\mathcal{B}$ enforces the truth of NOT $\mathcal{A}$. But by the principle of contradiction, $\mathcal{A}$ and NOT $\mathcal{A}$ cannot be both true, and since $\mathcal{A}$ is true, NOT $\mathcal{B}$ cannot hold, and therefore, by the principles of the excluded

third and of contradiction, $\mathcal{B}$ is true. There are also more technical proof techniques, such as the proof by induction, but logically speaking, they are all special cases of the general scheme just described.

In this book, the end of a proof is marked by the symbol □.

http://www.springer.com/978-3-540-36873-1

Comprehensive Mathematics for Computer Scientists 1
Sets and Numbers, Graphs and Algebra, Logic and Machines, Linear Geometry
Mazzola, G.; Milmeister, G.; Weissmann, J.
2006, XIV, 388 p., Softcover
ISBN: 978-3-540-36873-1