Weak type theory

Kuijpers, N.J.M.

Published: 01/07/2007

Document Version
Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:
• A submitted manuscript is the author's version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

Citation for published version (APA):

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal?

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 04. Jan. 2019
Weak Type Theory

by

N.J.M. Kuijpers

Eindhoven, July 2007
# Table of Contents

1 Introduction ................................................................................................................. 3  
1.1 The need for a language of mathematics ............................................................ 3  
1.2 What is WTT? ..................................................................................................... 4  
1.3 The moral nature of WTT ................................................................................... 5  
1.4 A brief history of WTT ....................................................................................... 5  
1.5 An overview of the course .................................................................................. 6  

2 Natural and mathematical language ............................................................................ 8  
2.1 Introduction ......................................................................................................... 8  
2.2 Hierarchy in language ......................................................................................... 8  
2.3 Phonemes and morphemes .................................................................................. 9  
2.4 Words ................................................................................................................ 
2.5 Phrases .............................................................................................................. 11  
2.6 Sentences ........................................................................................................... 12  
2.6.1 Sentences as word combinations .............................................................. 12  
2.6.2 Sentences as elementary text units ............................................................ 13  
2.7 Groups of sentences .......................................................................................... 14  
2.8 Texts.................................................................................................................. 15  
2.9 Summary ........................................................................................................... 15  

3 Formula basics .......................................................................................................... 16  
3.1 Introduction ....................................................................................................... 16  
3.2 Formula symbols ............................................................................................... 16  
3.3 Formula notations ............................................................................................. 18  
3.3.1 Prefix, infix, postfix .................................................................................. 18  
3.3.2 Exceptions ................................................................................................. 19  
3.3.3 Parentheses ................................................................................................ 20  
3.3.4 Polish and Reverse Polish Notation .......................................................... 21  
3.3.5 Tree representation .................................................................................... 22  
3.3.6 Symbol priorities ....................................................................................... 26  
3.4 Binding in formulas .......................................................................................... 27  
3.4.1 Types of binders ........................................................................................ 27  
3.4.2 Binders in trees ......................................................................................... 29  
3.4.3 Binding with restrictions ........................................................................... 33  
3.4.4 The @-sign ................................................................................................ 35  
3.4.5 The sum binder ......................................................................................... 37  
3.4.6 Substitutions .............................................................................................. 37  
3.5 Summary ........................................................................................................... 41  

4 Principles of Weak Type Theory .............................................................................. 42  
4.1 Introduction ....................................................................................................... 42  
4.2 Variables ........................................................................................................... 42  
4.3 Constants.......................................................................................................... 43
1 Introduction

1.1 The need for a language of mathematics

In textbooks, papers and other sources that are mathematical in nature, we encounter lots of mathematical ideas. There is a wide variety of ways in which these ideas are expressed, because no formal syntax exists for this purpose. In Example 1.1 we show a short text fragment, found in [12], which can be formulated in lots of different ways. This demonstrates the freedom and diversity we have when formulating mathematical ideas.

Example 1.1 The freedom to formulate mathematical content exposed.

Original text fragment:
“Let $a$ be the number $1 + \sqrt{2}$ and $n \in \mathbb{N}$. Then $a^{n+2} = 2a^{n+1} + a^n$ holds.”

A list (which is still incomplete) of the many variants to express the same mathematical idea:

- “Let $a$ be the number $1 + 1,4142$ and $n \in \mathbb{N}$. Then $a^{n+2} = 2a^{n+1} + a^n$ holds.”
- “Let $a$ be the real number $2,4142$ and let $n$ be a natural number. Then $a^{n+2} = 2a^{n+1} + a^n$ holds.”
- “Assume $a$ equals $1 + \sqrt{2}$ and $n \in \mathbb{N}$. Then $a^{n+2}$ equals $2a^{n+1} + a^n$.”
- “$a^{n+2} = 2a^{n+1} + a^n$ holds if $a = 1 + \sqrt{2}$ and if $n$ is a natural number.”
- “$a = 1 + \sqrt{2}$ and $n \in \mathbb{N}$ implies $a^{n+2} = 2a^{n+1} + a^n$ is true.”
- $(a = 1 + \sqrt{2} \land n \in \mathbb{N}) \implies a^{n+2} = 2a^{n+1} + a^n$.
- “Let $n \in \mathbb{N}$ and $a := 1 + \sqrt{2}$. Then $a^{n+2} - 2a^{n+1} + a^n = 0$.”
- “Assume $a$ equals $1 + \sqrt{2}$. Then $a^{n+2}$ equals $2a^{n+1} + a^n$ if $n$ is a natural number.”

In this example we can see that the mathematical parts of the original text fragment can be formulated differently and that natural language words can be interchanged which semantically equivalent natural language words. We can even vary the order (first something is said about $a$, then about $n$ and finally about an equivalence between two formulas containing $a$ and $n$) in which the original text is built up, as long as the semantic meaning of the alternative formulation remains the same. The freedom with which we can make variations that still mean the same is tremendous.
Both the original text fragment from Example 1.1 and the alternatives that we proposed are written in what we call *Common Mathematical Language* (CML). CML is a language used to express mathematical ideas. It is defined in [9] as "the linguistic machinery which mathematicians preferably use to express mathematical content and to communicate with their fellow mathematicians".

CML is a mixture of natural language, mathematical symbols and mathematical formulas. Since there is no prescribed standard to write down CML, we say it is *informal*. This means that it has no formal basis and therefore heavily depends on the understanding of the reader, which has to cope with the enormous freedom the author of a text has to write down his ideas. Moreover, ambiguity, imprecision and incompleteness are a few important disadvantages of natural language which also occur in CML. This grew the need for a *formal* language of mathematics, dealing with these disadvantages in a prescribed, structured way as much as possible.

A formal language that can act as a substitute for CML has been proposed by Kamareddine and Nederpelt in [9]. Their aim was to provide a formal language to express mathematical ideas, while keeping it as close to CML as possible. This formal language, named *Weak Type Theory* (WTT), is the main topic of this course.

### 1.2 What is WTT?

As we already briefly explained in section 1.1, WTT has been proposed to be a formal language for expressing mathematical content. It captures the exact same intention of a mathematical text written in CML, but uses a formalism that makes the text intuitively more clear and reduces ambiguity, imprecision and incompleteness in it.

The name WTT points out that it is a kind of type theory, but do we know what a ‘type theory’ is? A nice definition, found at Wikipedia, is the following:

> "At the broadest level, type theory is the branch of mathematics and logic that first creates a hierarchy of types, then assigns each mathematical (and possibly other) entity to a type."

Since WTT is a type theory and since we saw the definition above, it will be no surprise that in WTT there exists a hierarchy of types. These types in WTT are of linguistic nature, which makes them very comprehensive. Generally, a type in type theory can be considered to be a collection of objects that have something in common, for instance the natural numbers. An example of a type in WTT would be the type *noun*, which intuitively has the same meaning as the concept of noun has in natural language.

The ‘weakness’ of WTT is attributed to the fact that the linguistic types that exist in WTT, provide much less information than the types in ‘ordinary’ type theory. In WTT for instance, it doesn’t matter whether ‘2’ is a natural number, an integer or even a real
number. It is just a term. In ‘ordinary’ type theory however, there are significant differences between the 2’s of either one of these types.

But why did we want to have a language of mathematics like WTT at all? What have we gained from it? Of course, we have gained better insight and better understanding of mathematical texts by translating them into WTT. Moreover, WTT reduces the step for mathematicians towards the opportunity of getting computerized assistance in developing new mathematical ideas and verifying mathematical theories. A computer program might be used to translate a given WTT text into type theory and possibly verify its correctness. Another possible advantage we want to mention is that documentation of checked WTT texts can be combined into a widely accessible knowledgebase.

1.3 The moral nature of WTT

We want to stress that by no means WTT is demonstrably better than CML. We simply cannot decide upon this, because this is a moral judgment that each individual has to make for himself.

When we find that a text is incomplete, imprecise, ambiguous, or perhaps perfectly clear, someone else doesn’t have to agree with us at all. The crucial point here lies in the amount of foreknowledge one has at its disposal. So we emphasize that the usefulness of WTT heavily depends on the reader. When a text in CML is read by two different people, it may be perfectly clear to one of them and utterly confusing to the other, although the text is exactly the same! For the first reader WTT is superfluous, while for the second it might provide the necessary additional information in order to be able to better understand the intention of the text.

WTT is a means to provide better insight in and understanding of a text, but its moral nature plays an important role too. In the examples we will see throughout this course, we will repeatedly show that there are multiple ways to write down a text in WTT. There exists no such thing as a unique way to convert a text into WTT. For almost every decision we make in these examples, there is at least one alternative, which is just as good (or just as bad) as the one we choose. Therefore we will try to show as many alternatives as possible.

1.4 A brief history of WTT

Before starting out with the actual course material, we first give a short overview of the history of WTT. WTT is based on the principles of WOT, which is a Dutch acronym for Wiskundige OmgangsTaal, meaning mathematical everyday language. In 1979, the Dutch mathematician De Bruijn introduced WOT in [3] and its grammar/syntax in [4]. He defined WOT as the mathematical everyday language we speak and write, that is a mixture of formulas and natural language words.
Though De Bruijn’s ideas about WOT formed an important basis for WTT, the greatest influence stemmed from his *Mathematical Vernacular* (MV), which he introduced in 1987 and last revised in 1994. He introduced this MV to accommodate his ideas for a language of mathematics [7].

Since this MV is primarily about structuring mathematical texts, flags are used in MV as a graphical mechanism to present the context of MV books in a more striking way. Using flags not only provides a better insight in the structure of the text, but also a better understanding of what the author of the text intended to say.

WTT can be seen as a refinement of De Bruijn’s MV, with as main difference that mathematical and logical correctness are not part of WTT. This mean that for instance ‘0 is greater than 1’ is a correct statement in WTT, though it is logically incorrect. WTT is mainly about the formalization and structuring of mathematical texts. Therefore, WTT is more informal than MV and hence stays closer to CML than MV does. A more extensive comparison of WTT and MV can be found in [9].

1.5 An overview of the course

We gently start this course by discussing natural language and mathematical language and the differences between them in chapter 2. We show how a language is built up by showing the various existing linguistic levels as well as how they are related to each other.

Chapter 3 teaches us the principles of formulas. It is a quite general chapter, serving as a solid foundation on which we can elaborate in later chapters. Subsequently we discuss all kinds of formula symbols that exist, as well as the different notations that have been accepted as being ‘common’ in mathematics. Finally, we introduce the notion of trees in this chapter, which is essential for the further understanding of this course.

Our very first acquaintance with WTT will take place in chapter 4 where we discuss the roles of variables and constants in WTT first. Then we will move on to features that are specific for WTT. The main focus point of this chapter is to explain how WTT formulas are translated and interpreted in natural language. Since in lots of cases there is more than one correct translation for one formula, we try to be as exhaustive as possible in mentioning alternative translations as well. We also introduce the tree representation for WTT formulas in this chapter.

In the fifth chapter we will adopt a perspective contrary to the one in chapter 4: we discuss how to construct correct WTT formulas from a natural language expression. We give preference to specifying some general WTT formula formats rather than giving a complete and abstract formal syntax of WTT.
Again, there usually is more than one WTT formula which is a correct translation of a particular natural language expression. Whichever result we end up with mostly depends on one's own interpretation and preferences.

Chapter 6 explains the notion of context, which is extremely important in both natural and mathematical language. We introduce a demarcation system, called flag notation, to structure and formalize mathematical texts and to indicate contexts more strikingly. The flag notation is an intuitively helpful instrument to emphasize the (implicit) structure of mathematical texts. We also discuss the book notation in this chapter, which compared to the flag notation, expresses the same but is more compact.

We conclude chapter 6 by showing that mathematical texts often accommodate lots of shortcomings like ambiguity or even the omission of information. We rigorously criticize some examples of larger mathematical texts after having done the translation into WTT flag notation first.

Finally, in chapter 7, we display a number of larger mathematical texts that we formalized using the flag notation. These texts are coping with numerous flaws, which we sum up. We try to indicate how these flaws can be eliminated by choosing another way of formulation.
2 Natural and mathematical language

2.1 Introduction

In everyday life, almost every day we are making contact with other people. During these moments of contact we are interacting and continuously communicating with each other. We have several means at our disposal to express ourselves. An important one is *body language*. Body language is a communication means that uses the human body to express for instance emotions and relations. Examples of body language are eye contact, posture, (arm) gestures and facial expression.

Opposite to body language, which is also referred to as nonverbal communication, we use *verbal communication* as well by using our ability to speak. To communicate verbally it is required to speak a *language*. Languages consist of an *alphabet* of characters, a *vocabulary* and a *grammar* to glue elements of the vocabulary together to well-formed expressions.

The English language for example, is a universal, basic language in which a number of ‘sublanguages’ exist. In various fields a jargon has evolved over the years. Jargon is an extension of natural language with words and terms that are invented to express things that natural language cannot. There exist medical, juridical, nautical, snooker and football jargons, to mention only a few. Mathematicians also felt the urge to invent new words to be able to express themselves better. This jargon, combined with a natural language, we will call *mathematical language* from now on.

In this chapter we zoom into several linguistic levels we distinguish in (all kinds of) languages. We do this in a hierarchical bottom-up approach, thus starting with the smallest linguistic level. In the remainder of this course we assume the English language as our basic language of expression. Our main subject of discourse will be the just mentioned mathematical language, as a sublanguage of the English language.

2.2 Hierarchy in language

When considering the English language, or in fact any possible language, we notice that everything we express with it is built from a number of smaller units. Each of these types of units that are present in a language is part of a so-called linguistic level. We can make distinctions between those linguistic levels by showing the hierarchy which exists among them. Figure 2.1 shows this hierarchical structure. It can be read from top to bottom and displays a kind of relation between the linguistic levels which is best described as an ‘is-built-from’ relation, which is displayed in Figure 2.1. So an element of the topmost level, i.e., a text, ‘is built from’ elements of one level below, i.e., sentence groups, of which an element in turn again ‘is built from’ sentences, etc.
2.3 Phonemes and morphemes

The smallest elements used in natural language are morphemes and phonemes. Phonemes are the vowels (like ‘a’, ‘e’ and ‘u’) and consonants (like ‘q’, ‘b’ and ‘z’) we use. Morphemes are fragments of words that have a meaning of their own as well. An example would be the word ‘construction’, with morphemes ‘con-’, ‘struct’ and ‘-ion’. The morpheme ‘con-’ is a prefix meaning ‘together’, ‘struct’ is a verb stem and ‘-ion’ is a postfix meaning ‘result of what the verb expresses’.

However, for our purpose, which is to teach a course in WTT, we can suffice with just the first five levels in Figure 2.1 and leave the morphemes and phonemes out of consideration for the rest of the course. This is indicated by the dashed horizontal line that is drawn in the figure.
2.4 Words

Though words are not the smallest elements present in natural language (as we noticed above), we consider this to be the case for mathematical language. There are a number of different categories of words used in mathematical language which are clearly distinct from each other. The two main categories are natural language words and mathematical words. They consist of several different subcategories as well, which are shown here:

1. Natural language words
   - Natural language words in their usual meaning.
   - Natural language words in a different, new meaning.

2. Mathematical words
   - Mathematical standard words and abbreviations.
   - Mathematical non-standard words.

Natural language words can be used in mathematical language in their normal meaning, but also in a meaning that is new and specific for the mathematical working field. In some cases this new meaning is closely related to its usual meaning, whereas in other cases the new meaning substantially differs from the usual one. This becomes clear when we look at Example 2.1 below.

Example 2.1 Examples of natural language words.

- ‘the’, ‘simplify’ and ‘equals’ are natural language words that have kept their original meaning in mathematical language.

- ‘origin’, ‘argument’, ‘relation’, ‘power’, ‘function’, ‘monotonous’ and ‘inequality’ all are natural language words that have adopted a different meaning in mathematical language.

When we think about words in natural language we consider them to be a concatenation of vowels and consonants. For mathematical words though, this does not always have to be the case. A mathematical word is a word, symbol or abbreviation, mainly occurring in a mathematical text, which can be considered as an independent element of that text.

These mathematical words can be either standard words or non-standard words. The former are often standard mathematical symbols and abbreviations with a predefined meaning, the latter are often single letters, combinations of letters, combinations of letters and numbers or self invented symbols without a fixed meaning.

We provide another example of the abovementioned mathematical standard words and mathematical non-standard words.
Example 2.2 Examples of mathematical words.

- \forall, \mathbb{N}, \Delta, \Sigma, \Rightarrow, \geq, \varepsilon, e, \lim, \pi, \cos and \cap are all mathematical standard words, symbols and abbreviations that have a universally known meaning.

- 'a', 'p', 'gr', 'n2' and '△' are mathematical non-standard words. There are infinitely many mathematical non-standard words. The last symbol is a self invented symbol and could for instance be used in a mathematical text about trapezoids.

We already mentioned that the mathematical standard words have a predefined meaning which has become so deeply rooted into the mathematical working field, that mathematicians 'blindly' associate them with that meaning every time they occur. Therefore it is not recommended to use these in another meaning. Example 2.3 will point out two examples of situations that must be avoided when mathematical standard words are concerned.

Example 2.3 Don'ts concerning mathematical standard words.

- 'length' should not be abbreviated to 'ln', because the latter is automatically associated with the natural logarithm function.

- The use of 'e' for anything else than the base of the natural logarithm may be dangerous as well.

Similar restrictions are often applied to mathematical non-standard words. To denote an element from the natural numbers or real numbers it is common to use \( n \) and \( x \), respectively. Defining \( n \) as a real number and \( x \) as a natural number for instance, would cause intuitive difficulties for most people, though choosing these names is still allowed.

2.5 Phrases

A phrase is actually built from words, as shown in Figure 2.1. Because a phrase is built from at least two words, phrases are also called word groups. More formally, a phrase is defined as a concatenation of words, separated with spaces, that has a clear meaning of its own. This implies that the words occurring in a phrase are related to each other in some way: they form a linguistic unit. An example of a natural language phrase would be 'the green door'. Such a phrase itself can be used to build sentences like "John opens the green door."

In mathematical language there are two general types of phrases between which we draw a distinction. In the first place the object names. In fact, object names are formulas that function as a phrase when we 'translate' them into natural language. An example of an
object name would be ‘$x - 5$’, defining the object obtained when subtracting 5 from a
given $x$. We will come back to formulas in chapter 3 extensively.

Predicate phrases, noun phrases and adjective phrases belong to the second type of
phrases. An example of a predicate phrase would be ‘Man($x$)’, which states that the
predicate ‘Man’ applies to input $x$, or, in other words, $x$ is a ‘Man’. Note that ‘Man()’ is
not a function here because it doesn’t convert given input to a certain output. It just
verifies the truth-value when applied to certain input. It only delivers true or false as a
result.

A noun phrase is a phrase that is built around a noun. An example of this would be ‘the
least common multiple’, which is a phrase built around the noun ‘multiple’.
Characteristic for a noun phrase is that the other words (other than the noun they are built
around) in it cannot be left out because they are indispensable, i.e., they provide
additional information about the central noun in the phrase.

Finally, an adjective phrase is a group of words doing the ‘work’ of an adjective. An
example of this notion in mathematical language would be the phrase ‘upper bounded’,
which can be a part of, for instance, “The upper bounded subset of a set $A$”.

2.6 Sentences

Sentences are actually the pivot category in language. They are the units of information in
every natural language. From words and phrases we can make sentences, which implies
that these sentences consist of word combinations. But we can also consider sentences as
being the elementary building blocks to construct whole texts. In this case, sentences can
be seen as text units themselves.

2.6.1 Sentences as word combinations

We will first discuss sentences as combinations of words. We distinguish two kinds of
these sentences, namely statements/propositions and definitions. Statements and
propositions are formulas which function as a sentence. They are either true or false, and
nothing else. Definitions serve a special goal, viz. to fix a certain notion and give it a
name. In section 6.4 we discuss definitions in detail.

A subkind of the statements/propositions are the typing sentences, which are an adequate
means to describe that an object belongs to a certain class or is of a certain type. In
section 5.7 more will be said about this topic. For now, we will confine ourselves to
giving a few examples.
Example 2.4 Two statements/propositions, two definitions and two typing sentences, exactly in this order.

- \( x^2 = 3x + 10 \) is a proposition that results in either true or false, depending on the value of \( x \) that is provided as input.

- \( \forall_{n \in \mathbb{N}} (n \geq 0) \) is a proposition as well, ranging over all natural numbers. This particular proposition is true.

- \( \beta := \frac{2}{3} \sqrt{a - b} \) is a definition of \( \beta \), which is equal to a specific value, depending on the values of \( a \) and \( b \).

- \( l := \lim_{n \to \infty} (a_n) \) is the definition of \( l \), being the limit of \( a_n \), as \( n \) approaches infinity.

- \( x \in \mathbb{R} \) and \( x \in V \) are two typing sentences stating that \( x \) is a real number and an element of some set \( V \), respectively.

- \( 3 : \text{ a natural number} \); \( 3 : \text{ a real number} \); \( 3 : \text{ a threefold} \) are typing sentences specifying that the number 3 belongs to a specific domain. We can easily verify that these three sentences are all true.

Note that specifying the number 3 being a natural number can be done in two different ways: \( 3 \in \mathbb{N} \) and \( 3 : \text{ a natural number} \). Obviously, these typing sentences mean exactly the same, though there is a slight difference between both notations. We will discuss this in more detail in chapter 5.

2.6.2 Sentences as elementary text units

When sentences function as the elementary building blocks of a text fragment or a complete text, they play a particular role within that larger text structure. In a text fragment or a complete text, sentences are related to each other, i.e., are interdependent. Causal, logical, sequential and temporal relations between sentences are well-known.

A special type of sentences is the type of context sentences because they are used in a broader, context dependent way which we have not seen before. There are lots of different roles context sentences can play. In this section we confine ourselves to mentioning the most frequently used ones: context sentences can be definitions, generations and assumptions. In chapter 6 we will come back to this extensively.
2.7 Groups of sentences

A text fragment consists of one or more groups of sentences that in a way are related with each other. Text fragments can have various functions themselves as well. Often these functions become clear if the text fragment has been labeled. Text labels indicate how a certain fragment should be read. These labels are also called mathematical text markers. We are not trying to be exhaustive by enumerating all the existing mathematical text markers, yet we want to list the most common markers here.

- proof
- lemma
- definition
- corollary
- example
- conclusion
- theorem
- exercise
- etc.

It needs no further explanation that a fragment of text that is marked as an example should be read (and interpreted) in a totally different manner than for instance a fragment of text that has been marked as proof, theorem or corollary. Furthermore, it is worth noting that some of these mathematical text markers can be related to each other as well. However, this exceeds the scope of this particular section. In section 2.8 we will come back to relations between various mathematical text markers briefly.

Relations between sentences cannot always be seen very easily. Sentences that seem to be independent of each other can nevertheless be related to each other. Imagine the two sentences representing a text fragment in Example 2.5.

Example 2.5 Text fragment example.

"Hello Mary, how are you doing?"
"Come in, John."

At first sight these sentences have nothing to do with each other. But humans are very skillful in inventing or imagining a context such that these sentences can be related to each other indeed. In this example, a logical context we can come up with to link the two sentences would be a situation where Mary is standing in her hallway after just having opened the front door for her good friend John who came to visit her.
2.8 Texts

A text is built from either sentence groups and/or sentences (also see Figure 2.1). Just as sentences in text fragments are related to each other, text fragments in texts can be related too. When examining mathematical texts, we see that a fragment labeled as theorem is often followed by a proof fragment and lemmas or theorems can very well be followed by corollaries.

More about mathematical texts and how these can be read, interpreted and analyzed properly, will be the main focus point of chapter 6.

2.9 Summary

In this chapter, we have introduced the way in which we consider language to be built up. We have developed a feeling for the linguistic levels that exist and the hierarchy in which they are related to each other.

We also explained the distinction between natural language and mathematical language. Though we made a clear distinction between these two, we see that both, as a matter of fact, have the same structure, i.e., both can be classified into the same sublevels.

The sentences-level is the linguistic level that functions as the main level, since sentences are built up from words and/or phrases, but as a whole also are the elementary building blocks to construct sentence groups and texts. When we consider sentences in this last way, we often see that there are interdependent relations between sentences that directly follow upon each other in a sentence group or text. These relations between sentences tell us how to read and interpret a certain text, because, for example, a text spelling out a proof should be read differently than a text that formulates a conclusion.
3 Formula basics

3.1 Introduction

When we open an arbitrary textbook on mathematics at a randomly selected page, it is highly probable that we will encounter formulas. Formulas are extremely important in all kinds of fields, including mathematics of course. They occur in all kinds of different forms and notations and consist of a number of concatenated symbols that all have a meaning in the formula in which they occur. In this chapter we discuss all the basics of formulas that we consider to be an important foundation to build our WTT course upon.

First we explain the notion of formula symbols, followed by two extensive sections. The first of these sections discusses formula notations; especially some frequently used notational styles, parentheses, trees and priorities. The second one introduces binding in formulas, in particular the different types of binders, binding in trees, restrictions and substitutions. Finally, a brief résumé summarizes what we actually should have learnt from this entire chapter.

3.2 Formula symbols

We have seen several categories of phrases in mathematical language in section 2.5. We briefly mentioned a special category of phrases there, namely the category of formulas. Formulas play a characteristic role in mathematical language. In this course about WTT, we will interpret the word 'formula' in a broad sense. We consider a formula to be a phrase that consists of more than one mathematical word. We will divide the formulas that we are dealing with into the following two groups:

1. Formulas that denote an object name and therefore can be considered as being a mathematical word or phrase, such as:

\[ \forall W', \quad \frac{3}{4}(x - 2y^3), \quad \sin(2\pi), \quad \{ x \in \mathbb{R} | x > 5 \}. \]

2. Formulas that specify a statement and therefore can be considered as being a sentence, such as:

\[ a^2 - 3b^2 = c, \quad \forall x \in \mathbb{R} (\sqrt{x^2} > 0), \quad p \land q \Rightarrow r \lor s. \]

The formulas we see here are all built from letters, numbers and other common mathematical symbols. Some of these symbols are especially used to create new formulas, i.e., they function as a means to 'glue' mathematical words together into a formula. We distinguish four groups of these symbols, which we call operators, relation
symbols, junction symbols and nominators. Collectively we will call them formula symbols.

Each formula symbol can be applied to either object names or statements and will result in new object names or new statements again. In Table 3.1 we orderly explain what each category of formula symbols uses and results in. This table is followed by some examples to indicate how these formula symbols are used in practice.

<table>
<thead>
<tr>
<th>Category</th>
<th>Uses</th>
<th>Yields</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operator</td>
<td>Object name</td>
<td>Object name</td>
</tr>
<tr>
<td>Relation symbol</td>
<td>Object name</td>
<td>Statement</td>
</tr>
<tr>
<td>Junction symbol</td>
<td>Statement</td>
<td>Statement</td>
</tr>
<tr>
<td>Nominator</td>
<td>Statement</td>
<td>Object name</td>
</tr>
</tbody>
</table>

Table 3.1 Characterization of formula symbols

**Example 3.1** Operators, relation symbols, junction symbols and nominators.

- The operator ‘+’ creates the new object name ‘x + y’ when applied to object names ‘x’ and ‘y’. Some other operators are ‘-’, ‘\’ and ‘\-’.

- The relation symbol ‘>’ creates the statement ‘t > u’ when applied to object names ‘t’ and ‘u’. Other relation symbols are ‘\=’, ‘\leq’ and ‘\neq’.

- The junction symbol ‘\\exists_{n \in N}(.....)’ creates the new statement ‘\\exists_{n \in N}(n > 3)’ when applied to the statement ‘n > 3’. Other junction symbols are ‘\\wedge’, ‘\\neg’ and ‘\\forall_{n \in N}(.....)’.

- The nominator ‘ {...}’ creates, when choosing the real numbers as its domain and applied to statement ‘A(x)’, the set of all real numbers for which statement A(x) holds. The domain is placed left of the separating bar, whereas the statement is placed on the right of it.

Every formula symbol is applied to a fixed number of operands. The operator ‘+’ and the relation symbol ‘>’ are applied to two operands, whereas for instance the junction symbol ‘\neg’ is applied to only one operand. The number of operands a formula symbol takes is called the arity of the formula symbol. A formula symbol that takes 1, 2, 3, etc. arguments is called unary, binary, ternary, etc.

For the sake of completeness we want to mention here that WTT is equipped with two special constants. These are denoted by \uparrow and \downarrow and are called generalization and specification, respectively. Both generalization and specification are unary constants. The former typically takes a noun as input, whereas the latter takes a set, class or collection as
its argument. For now, we will disregard their meaning and merely mention them as being present. They will be discussed in more detail in section 4.5.

3.3 Formula notations

In section 3.2 we got ourselves acquainted with formulas and formula symbols. We have written down a number of formulas in that section without asking ourselves how they are written down and why they should be written down in that way. Maybe there are more ways to write down the same formula? This question can be answered affirmatively. We will discuss the most common notational conventions to write down a formula in this section. They all are based on the position of the formula symbols that occur in a formula.

3.3.1 Prefix, infix, postfix

When a formula symbol occurs in front of its operand(s), we speak of prefix notation. If the formula symbol is placed in between its operand(s), it is known as infix notation. It is noteworthy to point out that infix notation is only used when dealing with binary formula symbols, like ‘+’, ‘<’ and ‘∩’. Finally, postfix notation is the notational style where the formula symbol appears behind its operand(s). To illustrate these notions we will now give an example where we classify formula symbols as being prefix, infix or postfix symbols.

Example 3.2 Common prefix, infix and postfix formula symbols.

- ‘√’, ‘¬’ and g.c.d. (the binary operator ‘greatest common divisor’) are commonly used prefix symbols.

- ‘+’, ‘∪’, ‘∨’ and ‘≤’ are generally used as infix symbols. Note that they all have an arity of 2, i.e., they are all binary.

- ‘!’ (the factorial operator), ‘‘ and ‘‘ are a few of the formula symbols that mostly occur as postfix formula symbols.

However, we are not forced to always write down ‘+’ as an infix symbol or ‘2’ as a postfix symbol, like we did in Example 3.2. The formula ‘3 + 8’ in infix notation can also be written as ‘+(3, 8)’ or even as ‘(3, 8)+’, whereas ‘x2’ can be written as ‘x ↑ 2’ (infix notation) and ‘↑(x, 2)’ (prefix notation), where ‘↑’ should not be confused with the special generalization constant we briefly mentioned earlier.

Traditionally, most formula symbols are used in the way people are most acquainted with. You can see the most common notational styles for frequently occurring formula symbols in Example 3.2. So in general, ‘¬’ will be used as a unary prefix symbol, ‘+’ as
a binary infix symbol and ‘!’ as a unary postfix symbol. Because of these habits which we have learned ourselves, most formulas containing multiple formula symbols are written in a style that is a mixture of various notational styles. We will give an example of a formula in which prefix, infix and postfix formula symbols are used. Hence this formula is written in a mixed notational style here.

**Example 3.3** ‘\( n + 2 \cdot \sqrt{n}! \)’ is a formula subsequently containing the binary infix symbol ‘+’, the binary infix symbol ‘\( \cdot \)’, the unary prefix symbol ‘\( \sqrt{} \)’ and the unary postfix symbol ‘!’.

### 3.3.2 Exceptions

Not all existing formula symbols qualify for one of these three categories. There are some exceptions that cannot be characterized as being common prefix, infix or postfix formula symbols. The first of these exceptions we want to mention is ‘\( |x| \)’, to denote the absolute value of a real number \( x \). The mathematical word \( x \) is in fact enclosed by this formula symbol, so neither prefix, nor infix, nor postfix would be a suitable denomination for this formula symbol. The exact same argument applies to the operators ‘(, )’, ‘[, ]’ and ‘{ }’, which are used for vector notation, interval notation and the enumeration of elements of a set, respectively. These enclosing operators are called *circumfix* formula symbols sometimes. The round brackets used to denote a matrix can also be mentioned in this context.

We find it useful to make some more remarks concerning rather strange notational habits which yet have become widely accepted. Though there are tremendously many details we could mention here, it is not our goal to be exhaustive on this area, hence we will stick to the most important ones.

For efficiency reasons a shorthand notation for formulas containing binary relation symbols has become quite popular. For example, the notation ‘\( i < j < k \)’ has been introduced instead of writing ‘\( i < j \land j < k \)’ and similarly, ‘\( x^2 - 4x + 4 = (x - 2)^2 \geq 0 \)’ is a shorthand for ‘\( x^2 - 4x + 4 = (x - 2)^2 \land (x - 2)^2 \geq 0 \)’.

Multiplication is an operation for which various formula symbols may be used, such as ‘\( \times \)’, ‘\( * \)’ and ‘\( \cdot \)’. But when leaving out any of those three formula symbols, still a correct multiplication can be written down. In the formula ‘\( x(x^2 - 4) = x^3 - 4x \)’ the reader should add his preferred multiplication symbol in his own mind, namely between the ‘\( x \)’ and ‘\( \cdot \)’ on the left hand side and between the ‘\( 4 \)’ and the ‘\( x \)’ on the right hand side. Placing the ‘\( 4 \)’ and the ‘\( x \)’ next to each other without using a formula symbol in between is called *juxtaposition*.

Finally, there exist a few embedded conventions about the order of the operands. For example, we habitually write ‘\( 3x \)’ instead of ‘\( x3 \)’, ‘\( 2 \cos x \)’ instead of ‘\( (\cos x) 2 \)’ and
'\sqrt{3}' instead of '3\sqrt{3}'. Considering this last notational habit, we point out that there is a somewhat confusing and contradictory habit to write '2\frac{1}{2}' as a shorthand for '2+\frac{1}{2}'.

3.3.3 Parentheses

In a formula a lot of different formula symbols can occur. A special kind of formula symbols are parentheses, which are often called brackets as well. Parentheses always appear in pairs. Such a pair starts with a left parenthesis and ends with an accompanying right parenthesis. Pairs of parentheses are used to indicate the range of a formula symbol. This closely relates to the priorities of formula symbols, which we will get back to in section 3.3.6. For now, we will only focus on formulas containing (a number of) pairs of parentheses and the correct use of them. This can be done best by looking at some examples, starting with a rather simple one.

Example 3.4 \((x+5)\cdot 2-(y/2)+(x)\) is a formula with three pairs of parentheses.

In this example, it is obvious which parentheses are associated with each other, namely each right parenthesis accompanies the directly preceding left parenthesis. When this is the case, the pairs of parentheses are said to be unnested, meaning that they don't interfere with each other, i.e., no two left parentheses are observed in succession. We also note that the last pair of parentheses is abundant. Though the formula as displayed is indeed correct, readability is increased when parentheses around a single character are omitted.

Now we will give an example of a formula containing parentheses in a nested fashion, i.e., more than one left parenthesis occurs in succession.

Example 3.5 \((3x^2-(2\sqrt{6x+(-8)})+(3x/2))\) is a more complex formula with nested parentheses.

In this example, five pairs of parentheses occur in a nested fashion in total. It is more difficult to see which right parenthesis accompanies which left parenthesis than in Example 3.4. Therefore, we used different colors to indicate the five separate pairs.

It can be easily checked whether the structure of the parentheses in a formula is correct or not. This can be done by adding sequence numbers to the occurring parentheses in the following way:

- Add a zero in front of the first left parenthesis.
- Then, go rightwards, one position at a time. Increase the number by 1 if you encounter another left parenthesis; decrease it by 1 if you come across a right parenthesis. Ignore other formula symbols, variables and constants in the numbering.
After having gone through the whole formula, the first and last number should be both zeroes whereas all the numbers in between should be greater than or equal to zero.

For the above example, this would yield the following numbering (we have grayed out all other parts of the formula to mainly focus on the parentheses and their numbering):

\[ (1, 3^2 - (2 \cdot \sqrt{(3, 6x + (4, -8)2)} / (3, 3x / 2))_0 ) \]

When concatenating two formulas that both have their parentheses correctly structured, the numbering will contain another zero, exactly between the two formulas that are concatenated. By applying the described numbering technique to, say, \( n \) concatenated formulas, we will encounter exactly \( n + 1 \) zeroes in the numbering. If this is the case, then this concatenation of formulas is correctly structured, as far as parentheses are concerned.

3.3.4 Polish and Reverse Polish Notation

In order to achieve a consistent way of representing formulas, we can require entire formulas to be expressed in only one specific notational format. For instance, we can require a formula to be entirely written in prefix notation or in postfix notation. Prefix and postfix versions for a given formula are shown in the example below:

**Example 3.6** The formula ‘((15 / (7 - (1 + 1))) \cdot 3 - (2 + (1 + 1)))’ has the following equivalent prefix notation:

\[ - \cdot / 15 - 7 + 1 1 3 + 2 + 1 1 \]

It can also be written down in postfix notation:

\[ '15 7 1 1 + - / 3 \cdot 2 1 1 + + -' \]

Note that ‘15’ in this example must be read as the number fifteen, whereas the rest of the occurring numbers all are 1-digit numbers.

A prefix notation of a formula as displayed above, without any parentheses, is said to be in Polish notation. Accordingly, the postfix notation of a formula, without parentheses, is also known as Reverse Polish Notation (RPN).

A question we could ask ourselves now is why someone would want to write formulas down in prefix or postfix notation. By looking carefully at Example 3.6, the answer to this question can be found. In the original formula that was given we used parentheses,
but in the equivalent prefix and postfix notations we did not. That's because they have become superfluous.

The reason why we used parentheses in the original formula was to override priorities of formula symbols, i.e., to determine to which operand(s) a formula symbol is applied. We will pay more attention to priorities of formula symbols in section 3.3.6.

When we go through a given prefix formula from left to right and we encounter a formula symbol, we know that this formula symbol is applied to the directly succeeding operand(s). The number of operands to which the formula symbol is applied corresponds to the arity of the formula symbol we encounter. When traversing a given postfix formula, also from left to right, a formula symbol that we come across is applied to the operand(s) directly preceding it. So in prefix and postfix formulas it is always unambiguously clear, without having to use parentheses, to which operand(s) every formula symbol is applied. Again, the arity determines the number of operands to be associated with the formula symbol.

3.3.5 Tree representation

Formulas can be represented in various notational styles. We already mentioned the prefix, infix and postfix notation, as well as a mixture of these. A second possibility to present formulas, other than writing down its building blocks in a specific format, is to present them graphically in the form of a tree.

In nature, trees have roots, branches and leaves. In mathematical trees these are called the root, the edges and the nodes. A bit counterintuitive is the fact that in mathematics trees are usually not standing up straight as they do in nature but are hanging upside down with the root being the only node at the topmost level. The bottom nodes are called leaves. A tree contains exactly one path from the root node to every arbitrary other node. Moreover, no cycles exist in it.

But how can trees be related to formulas? That is, how can a textual representation on the one hand and a graphical representation on the other hand, provide us with the same information? How are formula symbols and mathematical words captured in tree representation? All of these questions will be answered in this section.

The transformation of a formula into a tree structure which represents that exact same formula can be split up into a number of small steps. The first step in this process is to identify the most important formula symbol(s) in a formula, called main formula symbol(s). A main formula symbol is a symbol that applies to the formula as a whole, and not just to a part of the formula. With this we mean that a main formula symbol, together with its operands, forms the entire original formula.
There is always at least one main formula symbol present in a formula, but there could also be more than one. To this we will come back in section 3.3.6. In the following example the main formula symbols of three different formulas are identified. These formulas all have only one main formula symbol.

**Example 3.7** A couple of formulas with their main formula symbols.

- \(\sqrt{3c}/b^2\) is a formula with binary main formula symbol '÷'
- \((A \land B) \land D\)' is a formula with unary main formula symbol '\(\land\)'
- \((P \Rightarrow Q) \lor \neg R\)' is a formula with binary main formula symbol '\(\lor\)'

The three formulas in Example 3.7 all have one main formula symbol that applies to two subformulas. These subformulas itself can be analyzed again by identifying a main formula symbol for each subformula. Actually, we should say main subformula symbol instead of main formula symbol, because we are looking one level deeper now.

The second step now is to draw the root node of the tree we are going to construct and to attach the previously identified main formula symbol to this root node. However, we do not write the main formula symbol just anywhere near the root node. If it is a prefix symbol, we place it left of the root node, if it is an infix symbol we place it below the root node and if it is a postfix symbol, we place it right of the root node. This convention actually not only applies to the root node. We will use it for all other nodes in the tree as well.

When the root node has been drawn and its accompanying main formula symbol has been attached to it, we need to draw edges from it. How many edges we draw depends on the arity of the formula symbol: if it is unary, then we draw 1 edge; if it is binary we draw 2 edges, etc. At the bottom of every edge we draw a new node, which we label with the main formula symbol of the corresponding subformula.

Now this process repeats itself for each subformula until we no longer encounter formula symbols but only variables and/or constants. These variables and constants will be the leaves of the tree and are always placed below the corresponding node in the tree. We adopt the general rule here that variables can only occur at the leaves of the tree, whereas constants may occur both at leaves and at higher levels.

In Figure 3.1 we show how the three formulas that were given in the previous example can be represented by trees. Notice that we use both multiplication and raising to a certain power as a binary infix symbol in these trees. We will keep on doing this silently for the remainder of this course.
When a tree representation is given, there is a really straightforward way to retrieve the corresponding formula. In order to be able to do this it is required that every symbol appearing at a node in the tree is written down at the correct side of the node, either at the left side of it, below it or at the right side of it. If this requirement is met, we can start leftwards of the root node and traverse along the edges as if we are encircling the tree. Every time we pass a label we write it down, building up a list in this way. When we are back at the (right hand side of the) root again, we got our formula when reading the result of what we have written down from left to right.

The attentive reader must have noticed that we silently left out all parentheses in the trees. We did this on purpose because they are of no avail here. Actually we could have placed parentheses around every node in the tree that has at least one outgoing edge. These parentheses would then indicate for each formula symbol which its corresponding operands are. However, the hierarchy in a tree is such an apparent one that we can omit parentheses and still easily see the structure of the formula.

A special case in tree notations is the commonly used shorthand notation to describe the range of a certain variable. As an example of this we already mentioned \( i < j < k \) in section 3.3.2, which denoted the fact that both \( j \) is greater than \( i \) and \( j \) is smaller than \( k \): \( i < j \land j < k \). In a tree we will depict this, as you may already expect, as follows:

---

**Figure 3.1 Trees of the formulas given in Example 3.7**
Figure 3.2 A commonly used shorthand in mathematics

The opposite of using shorthand notations, which boils down to saying things in as few words as possible, is adding words or altering notations to improve readability. Being more descriptive in what we want to say can reduce the chance of misconception. This principle we will call sugaring. The added words or altered notations are the so-called ‘sugar’ which makes things ‘sweeter’ for the reader. In this context, ‘sweeter’ means easier to read and understand. In Example 3.8 below, a list of both sugared and unsugared versions of various phrases is displayed in a table to point out how sugaring is used in WTT.

Example 3.8 Phrases and sentences with and without sugaring. The ‘sugar’ is indicated in bold face.

<table>
<thead>
<tr>
<th>Unsugared</th>
<th>Sugared</th>
</tr>
</thead>
<tbody>
<tr>
<td>increasing ([a, b))</td>
<td>increasing on [a, b]</td>
</tr>
<tr>
<td>an edge (ΔABC)</td>
<td>an edge of ΔABC</td>
</tr>
<tr>
<td>the average (x, y)</td>
<td>the average of (x, y)</td>
</tr>
<tr>
<td>divisible (m)</td>
<td>divisible by (m)</td>
</tr>
<tr>
<td>lies between (n, k, l)</td>
<td>n lies between k and l</td>
</tr>
<tr>
<td>goes through (l, S, T)</td>
<td>(line) l goes through (points) S and T</td>
</tr>
</tbody>
</table>

Table 3.2 Sugaring applied to various phrases and sentences

The first four entries in Table 3.2 are phrases, whereas the last two entries are sentences. We can see a difference in the sugaring between those two. For phrases, sugaring is merely done by adding an appropriate preposition. For sentences however, we see that also the structure of the sentence can change in order for them to be better readable.
3.3.6 Symbol priorities

When in a formula a number or expression is both preceded as well as followed by an operator, a relation symbol or a junction symbol, we need a rule to determine which of these should be applied first, i.e., which priority they have. Especially when a formula contains a number of binary infix formula symbols, such a rule would be convenient.

The generally known order of operations is often described by the phrase “Please Excuse My Dear Aunt Sally”, frequently abbreviated to PEMDAS. It stands for Parentheses, Exponents, Multiplication, Division, Addition and Subtraction. So, first expressions between parentheses should be evaluated, then exponentiation, then multiplication and division and finally addition and subtraction. When we also include relation symbols and junction symbols, which all have lower priority than the PEMDAS-symbols, we get the hierarchy of formula symbols as shown in Table 3.3. The higher in the hierarchy the formula symbol, the higher its priority.

<table>
<thead>
<tr>
<th>Formula symbols</th>
<th>Names</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>(</td>
<td>parentheses</td>
<td>-</td>
</tr>
<tr>
<td>^</td>
<td>exponentiation</td>
<td>operators</td>
</tr>
<tr>
<td>. /</td>
<td>multiplication, division</td>
<td>operators</td>
</tr>
<tr>
<td>+ -</td>
<td>addition, subtraction</td>
<td>operators</td>
</tr>
<tr>
<td>&gt; &lt; = ≥ ≤</td>
<td>greater than, less than, equal to, at least, at most</td>
<td>relation symbols</td>
</tr>
<tr>
<td>¬</td>
<td>not</td>
<td>junction symbols</td>
</tr>
<tr>
<td>∧ ∨</td>
<td>and, or</td>
<td>junction symbols</td>
</tr>
<tr>
<td>⇒</td>
<td>implication</td>
<td>junction symbols</td>
</tr>
<tr>
<td>⇔</td>
<td>bi-implication</td>
<td>junction symbols</td>
</tr>
</tbody>
</table>

Table 3.3 Hierarchy of frequently used formula symbols

Remarkable is that some formula symbols have equal priority. Then the question arises which of two formula symbols to apply first when they occur in the same formula and have mutual operands. To avoid ambiguity about this, there exist rules called the left-associativity and right-associativity rules. These rules state that formula symbols of equal priority should be applied from left to right or from right to left, respectively. So the formula ‘a + b - c’ must be read as ‘(a + b) - c’ when applying the left-associativity rule. In the remainder of this discourse we choose to explicitly use parentheses to prevent multiple interpretations of formulas.
Usually, parentheses, which have the highest priority as we have just seen, are used as a means to indicate the range of formula symbols. The range of a formula symbol is often referred to as its scope. By using parentheses the priority of formula symbols can be temporarily overruled. In Example 3.9 below, we use parentheses to force subtraction to precede multiplication.

**Example 3.9** The order of operations can be changed arbitrarily by using parentheses. When writing down the formula ‘3 + 6 · 2’ according to the hierarchy above, we mean ‘3 + (6 · 2)’, which is equal to 15. When we write ‘(3 + 6) · 2’, we are giving addition a higher priority than multiplication in this formula. The result, obviously, is equal to 18.

### 3.4 Binding in formulas

Up to this point all the formulas we have seen were quite simple. The formulas we saw were concatenations of formula symbols with their operands and consisted of one (or sometimes more than one) main formula symbol. Every variable appearing in these formulas was free. But variables in formulas can also occur as bound variables. For now we will only briefly mention these two types of variable occurrences. We will come back to variables in more detail in section 4.2. In this section we will take a look at a more complex kind of formulas, namely the kind with binders occurring in them.

#### 3.4.1 Types of binders

Binders, or variable-binding operators, are in fact operators that ‘tie up’ a variable within a formula. There are two types between which we make a clear distinction in this course, although they ‘work’ in the same way. We distinguish between the so-called ‘common mathematical binders’ and the ‘WTT binders’. The latter will be discussed in chapter 4, the former will form the basis of this chapter.

Common mathematical binders are embedded and used in all of the different mathematical working fields, like algebra, logical reasoning and set theory. Some of the most frequently used common mathematical binders are listed here (though there are even more): ∀, ∃, ∪, ∩, Σ, Π, ∫, MIN, MAX, #, lim and {...}. We will not elaborate on their exact mathematical meaning here but we will mainly focus on how they are used. In Example 3.10 we can see some formulas containing these common mathematical binders. After this example we will discuss the structure of these formulas.

**Example 3.10** A list of the most frequently used common mathematical binders, for each of them providing its name and an example of how they are used to bind a variable. Also, the bound and free variables, if any, are indicated.
<table>
<thead>
<tr>
<th>Binder symbol</th>
<th>Binder name</th>
<th>Example</th>
<th>Bound variable(s)</th>
<th>Free variable(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>∀</td>
<td>Universal quantification</td>
<td>$\forall_{x \in \mathbb{R}} (x + y &gt; 0)$</td>
<td>$x$</td>
<td>$y$</td>
</tr>
<tr>
<td>∃</td>
<td>Existential quantification</td>
<td>$\exists_{n \in \mathbb{N}} (Q(n))$</td>
<td>$n$</td>
<td>-</td>
</tr>
<tr>
<td>/</td>
<td>Uniqueness quantification</td>
<td>$\exists!_{n \in \mathbb{N}} (Q(n))$</td>
<td>$n$</td>
<td>-</td>
</tr>
<tr>
<td>∪</td>
<td>Union</td>
<td>$\bigcup_{i \in I} (A_i)$</td>
<td>$i$</td>
<td>-</td>
</tr>
<tr>
<td>∩</td>
<td>Intersection</td>
<td>$\bigcap_{i \in I} (\bigcap_{j \in J} (B_{i,j}))$</td>
<td>$i, j$</td>
<td>-</td>
</tr>
<tr>
<td>∑</td>
<td>Sum quantification</td>
<td>$\sum_{n=1}^{10} (n^2 + k)$</td>
<td>$n$</td>
<td>$k$</td>
</tr>
<tr>
<td>∏</td>
<td>Product quantification</td>
<td>$\prod_{n=1}^{N} (m \cdot n)$</td>
<td>$n$</td>
<td>$m, N$</td>
</tr>
<tr>
<td>∫</td>
<td>Integration</td>
<td>$\int (x^{-1} e^{-x}) dx$</td>
<td>$x$</td>
<td>$y$</td>
</tr>
<tr>
<td>MIN</td>
<td>Minimum quantification</td>
<td>$\min_{n \in \mathbb{N}} (\sqrt{n} &gt; 5)$</td>
<td>$n$</td>
<td>-</td>
</tr>
<tr>
<td>MAX</td>
<td>Maximum quantification</td>
<td>$\max_{i \in \mathbb{Z}} (i^2 &lt; 5 \cdot i)$</td>
<td>$i$</td>
<td>-</td>
</tr>
<tr>
<td>#</td>
<td>&quot;Number of&quot; quantification (or: count quantification)</td>
<td>$#_{n \in \mathbb{N}} (0 \leq n &lt; m \cdot \pi)$</td>
<td>$n$</td>
<td>$m$</td>
</tr>
<tr>
<td>{...}</td>
<td>Set notation</td>
<td>${ n \in \mathbb{N}</td>
<td>n^2 &lt; 25 }$</td>
<td>$n$</td>
</tr>
<tr>
<td>lim</td>
<td>Limit</td>
<td>$\lim_{x \to 0} \left( \frac{\sin(x)}{x} \right)$</td>
<td>$x$</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 3.4 Examples of the most common mathematical binders

The examples in Table 3.4 clearly show the structure of formulas containing the most common mathematical binders. We see that usually the binder is in front and the actual formula comes after the binder and is enclosed in parentheses. The variable that is bound is placed in subscript, either below or slightly right of the binder. Furthermore, all kinds of free variables, constants and formula symbols are occurring in the formulas.

In most of the binders the bound variables are tied to a specific domain, except for the sum and product binders. In those two binders a specific domain is given by specifying a starting value and a maximum value for the bound variable. It is silently assumed that the bound variable $n$ in both formulas comes from the domain $\mathbb{N}$ and thus takes any natural number value within the specified range.

Exceptions to the standard notation for formulas with binders are the integration, limit and set notations from Table 3.4. We will describe them in more detail here, because they
require special attention. We also show how they can be written in the standard notation for formulas with binders, as described above.

- The integration binder is a special one. The bound variable does not appear together with the binding symbol but it is placed after the formula, behind the $d$, denoting the variable of integration. As a domain here implicitly the real numbers are assumed.

This is not the only way is which integrals appear though. Often the domain is restricted by a given lower bound and upper bound, which are placed below and above the integral symbol. There are even more ways to write down an integral formula but we will not discuss those here.

When making an attempt to write the integral formula from Example 3.10 in the standard way, we will get something like this: $\int_{x_{\in \mathbb{R}}^{x^{-1}} e^{-x}}$.

- The set notation differs the most from the standard notation of a formula with a binder. The entire formula is enclosed in curly braces. Inside those curly braces we will find a separator ('bar'), left of which the bound variable is bound to a domain and right of which the actual predicate is written down. When trying to rewrite this to a standard form, we get the following: $\text{Set}_{n \in \mathbb{N}} (n^2 < 25)$.

- The limit binder binds the $x$, which approaches the value 0 in this case. Again, no explicit interval or domain is specified. The domain for limit binders is the real numbers (or a subset thereof), because the bound variable approaches the specified value. Any attempt to rewrite the limit binder into a standard representation will lead to a fairly awkward result. Therefore we will not give it a shot at all.

### 3.4.2 Binders in trees

To incorporate binders in the tree representation we saw in section 3.3.5, we introduce a new notation to be able to distinguish them from other parts of formulas. Instead of drawing closed nodes (see Figure 3.1 and Figure 3.2) we draw open nodes for binders. These binder nodes are drawn as open nodes deliberately, because we can now place the variable that is bound inside of them.

To emphasize that an occurrence of a certain variable in a formula is bound, we will draw a dashed line back to the open node where that variable was bound originally. Furthermore, we will write the binder itself in front of, i.e., to the left of, this open node.

Binder nodes are always binary. The left branch of a binder node specifies the domain of the binder whereas the right branch contains the formula that is actually bound by it.
Schematically this looks as depicted in Figure 3.3, where ‘B’ stands for ‘Binder’, ‘bvar’ stands for ‘bound variable’, ‘D’ stands for ‘Domain’ and ‘F’ stands for ‘Formula’. Notice the fact that a ‘Formula’ itself may contain other binders as well, which of course have their own domain and formula again.

The triangular shapes representing the domain and formula parts indicate that these parts themselves are entire trees as well, consisting of at least one node. In those subtrees bound variables can also appear again.

![Figure 3.3 Schematic structure of WTT binders](image)

In this course, we chose not to define an abstract formal syntax for WTT and its binders, but to keep things as simple as possible. For those who want to have a look at this abstract syntax anyway, we recommend [9], where Kamareddine and Nederpelt account for a complete syntax description which is more formal and abstract than we show here.

We can ask ourselves now whether drawing these dashed lines every time, to indicate the binding place of a variable, is absolutely necessary. The answer to this question is no, because we can discover this in another manner as well, as we will describe here:

- Start at the leaf node to which the variable belongs.
- Backtrack towards the root of the tree, taking one branch at a time.
- At every encountered binding node, check whether the variable within it is the same variable as we are checking at the moment. If so, then this is the place where that variable got bound. If not, backtrack along the branch ending in the binding node as described in the previous step, until a binding node containing a matching variable is found.
- If you reach the root and still have no match, the variable you are checking is a free variable in this tree.

So, even though we are not obliged to drawing these dashed lines every time we want to indicate that a variable is bound, we will persist in doing this for the sake of convenience. The dashed lines immediately and unambiguously tell us whether a variable is bound or
not. They will be present in each example that concerns trees in the remainder of this course.

When we draw trees of formulas containing bound variables, we adopt the convention that in each formula/tree representation a specific variable name may occur only once, either in a bound or free fashion. In other words, both free and bound variables must be drawn from disjoint sets and need to be mutually distinct. In [1], Barendregt introduced this convention as the Variable Convention. In literature, this convention is frequently referred to as the ‘Barendregt Variable Convention’, or ‘BVC’ in short.

In the examples 3.11 to 3.13 we will now give a number of formulas, together with their tree representation. For each example we also give the formula the tree represents.

**Example 3.11** \( \exists p: \text{a prime number} \exists q: \text{a prime number} \) \( (p \neq q \land 19 = p \cdot q + 1) \)

![Tree representation of a formula with \( \exists \) binders](image-url)
Example 3.12  \( \{ x \in \mathbb{R} \mid \forall p \in \mathbb{Z} (|x - p| > \frac{1}{4}) \} \)

Figure 3.5 Tree representation of a formula with a Set binder

Example 3.13  \( \text{MIN}_{i \in \mathbb{N}} (\sqrt{2 \cdot i} > 5) \)

Figure 3.6 Tree containing the MIN binder
3.4.3 Binding with restrictions

We have seen numerous examples of binders in section 3.4.1. In all these examples, a variable is bound to a specific domain, for instance the domain of the real numbers, denoted by $\mathbb{R}$. But sometimes it might be the case that we want to exclude a value, or a range of values, from the domain the variable is bound to. In other words, we don’t want the bound variable to be an element of the entire domain, but of a particular subset of this domain.

There are several ways to denote such a subset. We could do this by means of a nested domain using the set binder for instance, or by giving the restricted domain a name first and then using this name in the actual binder. We will show both of these notations in Example 3.14.

**Example 3.14** Two correct ways to specify subsets of domains.

- $\forall x \in \{y \in \mathbb{R} | y \geq 3\} \ (x^2 \geq 0)$
- $D = \{y \in \mathbb{R} | y \geq 3\}$
  
  $\forall x \in D \ (x^2 \geq 0)$

The latter notation is quite laborious and therefore also impractical. The former notation will prove itself very useful later in this section, when we will convert formulas containing binders into tree representation. However, for convenience’s sake we introduce a shorthand notation for it that we will use when we are only dealing with formulas rather than trees. Our preferred abbreviated notation for ‘$\forall x \in \{y \in \mathbb{R} | y \geq 3\} \ (x^2 \geq 0)$’ is ‘$\forall x \in \mathbb{R} | x \geq 3 \ (x^2 \geq 0)$’.

By abbreviating the first formula of Example 3.14, we applied so-called binding with type restriction which is much more elegant than the abovementioned notations. It is somewhat similar to the ‘normal’ set notation ‘{ ... | ... }’, except for the curly brackets that are left out. The restriction can be specified using one bound variable now instead of two.

The formula ‘$\forall x \in \mathbb{R} | x \geq 3 \ (x^2 \geq 0)$’ actually is true for any real number, so the restriction doesn’t have an added value here. However, there are situations where we are more or less compelled to make certain restrictions due to mathematical considerations. In Example 3.15 we will list a few of these cases and explain why domain restrictions must explicitly be made there.
Example 3.15 Compelled domain restrictions.

- In the formula ‘$\exists_{x \in \mathbb{R} \mid x > 1} (\ln (\ln x) < 0)$’ we require $x$ to be strictly greater than 1, because the natural logarithm function is undefined for values smaller than 1. In this case, $x$ may not even be equal to 1, because taking the natural logarithm of subexpression $\ln(1)$ has 0 as a result, of which subsequently the $\ln$ is taken again, leading to an incorrect use of the $\ln$ function.

- In the formula ‘$\prod_{n \in \mathbb{Z} \mid n > 0 \land n \text{ is even} \land n \leq 10} (\sqrt{n})$’ we see three restrictions that all must hold because of the conjunction symbols in between. Integer $n$ may not be negative, because extraction of a root is undefined for negative numbers.

- In the formula ‘$\forall_{p \in \mathbb{Z}} \exists_{q \in \mathbb{N} \setminus \{0\}} (\frac{p}{q} > 10)$’, $q$ can not just be any arbitrary natural number. The value 0 for $q$ is explicitly excluded from this domain, because division by 0 of course is never allowed.

In the middle case of Example 3.15 we see that one of the restrictions is that the integer $n$ must be greater than 0. A more compact and also widely accepted shorthand notation to express this is to say directly that $n$ is an element of the positive integers, denoted by ‘$n \in \mathbb{Z}^+$’. For the same purpose, also the symbols $\mathbb{R}^+$, $\mathbb{N}^+$ and $\mathbb{Q}^+$ are used often to specify the domains of the positive real numbers, positive natural numbers and positive rational numbers, respectively.

As indicated earlier in this section, the nested domain notation mentioned in Example 3.14 is excellently suitable to incorporate a restrictive binding into tree representation. This will become clear when we look at Example 3.16, where this technique is applied.
Example 3.16  \( \exists_{x \in \mathbb{R} \mid |x| < \pi} (\sin(\frac{1}{2}x) = -\frac{1}{2}) \)

We see here that \( x \) is defined as an element of a set of real numbers \( y \), which fulfill the restriction specified in the binder, namely \( |y| < \pi \). Also worth noticing is, that in this formula the first bar ('\(|\)') is the restrictive bar, whereas the second and third bar must be seen as the combination denoting the absolute value operator applied to, in this case, \( y \).

3.4.4 The \( @ \)-sign

A special symbol which we haven’t mentioned before is the \( @ \)-symbol. This symbol is used in tree representation when one variable must be applied to another one. This typically is the case when we are dealing with functions or relations.

The \( @ \)-sign has been introduced in WTT to be able to stick to the general convention that variables in trees can only occur at leaf nodes and absolutely nowhere else. Without the introduction of the \( @ \)-sign, we would be forced to place a variable to a node that is higher in the tree, which is contrary to the convention we just stated.

The use of the \( @ \)-symbol can be best explained by looking at an example formula. When we examine the formula \( \exists_{f : \mathbb{R} \to \mathbb{R}} \forall_{x \in \mathbb{R}} (x > 0 \Rightarrow f(x) < 0) \), we see a bound variable \( f \), which is a function on the real numbers, and a bound variable \( x \), which is a real number. In the formula part, behind the \( \forall \) quantification, we see that \( f \) takes \( x \) as an argument. To state it differently, the function \( f \) is applied to the real number \( x \), which is enclosed in parentheses.
We will consider this *application* of $f$ to $x$ as an operation, although in formulas it is not explicitly denoted by a formula symbol as is common for most operations. When drawing the tree representation of a particular formula, we will denote such an application with the @-symbol from now on, which we will use as a binary prefix symbol. Thus ‘@$(f,x)$’ means ‘the application of $f$ to $x$’.

To illustrate the use of the @-symbol in trees, we will draw the tree corresponding to the example formula ‘$\exists f: \mathbb{R} \to \mathbb{R} \ \forall x \in \mathbb{R} \ (x > 0 \Rightarrow f(x) < 0)$’, that we stated earlier in this section, in Example 3.17.

It is important to notice that the @-symbol is used in accordance with the principles and rules to construct trees as stated in sections 3.3.5 and 3.4.2. In particular we want to mention that through the @-symbol we still have the guarantee that variables can only occur at the leaves of the tree.

**Example 3.17**

$\exists f: \mathbb{R} \to \mathbb{R} \ \forall x \in \mathbb{R} \ (x > 0 \Rightarrow f(x) < 0)$

![Tree representation](image)

**Figure 3.8** @-symbol appearing in tree representation

When talking about functions, we often see another function symbol appear, namely the $\to$ symbol (see Figure 3.8). This arrow should not be confused with the implication arrow $\Rightarrow$ because it has a totally different meaning, though it also is a binary infix symbol. The
symbol specifies the input domain for a function and maps it onto the output domain. The latter is commonly called co-domain. In this case, both the input domain and the co-domain are the real numbers.

### 3.4.5 The sum binder

The sum binder or sum quantification sums up values over a certain continuous range. This range is explicitly determined by initializing the bound variable and specifying the maximal value that the bound variable may be equal to.

The formula \( \sum_{n=1}^{m} x^n \) is an example of how the sum binder is used. This formula must be read as: "The sum of \( x^n \) over all integer values \( n \) from 1 to \( m \)." We will now give an example of how a more difficult formula with two sum binders should be transformed into tree representation. Surprisingly enough we see the word 'to' coming back in the tree representation and as you can see it is treated as a binary infix symbol.

**Example 3.18** \( \sum_{n=1}^{m} (x^n) \cdot \sum_{j=1}^{n} \left(\frac{n}{j}\right) \)

![Figure 3.9 Sum binder in tree representation](image)

### 3.4.6 Substitutions

An often used notion in mathematics is the notion of substitution. For instance, substitution can be done within formulas. We will consider substitution to be a ternary operation, requiring three arguments as its input. It is an operation intuitively analogous
to making a substitution in for instance a football match. The coach of a team can suddenly decide to bring on one player as a substitute for another player that is already on the pitch.

We define a ternary operator 'Subst' now, subsequently taking the following three arguments as its input:

- A formula that is the substitute (the player coming on the pitch)
- A formula for which the previous argument will be substituted (the player coming off the pitch). This formula usually is a single variable.
- A formula in which the substitution takes place (the team in which the substitution takes place)

Though substitution may seem a straightforward process of replacing formulas by other formulas, it should be done with great caution. In Example 3.19 we will show how easily it can go wrong by showing a substitution that has been wrongly executed.

**Example 3.19** Subst ('2x', 'x', 'x^3 - x + 3') does not result in '2x^3 - 2x + 3' as you might expect. The correct result of this substitution however is '(2x)^3 - 2x + 3', which is an essentially different formula. The parentheses are absolutely necessary. They indicate that the multiplication in this case has highest priority, as opposed to the hierarchy of formula symbols we presented in Table 3.3.

To avoid problems like the one in Example 3.19, we also apply the Subst operator to trees. Obviously, the Subst operator now takes three trees as its arguments rather than formulas, but they serve the same purpose as previously explained.

The substitution from Example 3.19 can now be represented as:

```
Substitute 2 for x in - + ,
2               3
```

```
               +
               /
         -     3
        /
 1   -
  /
x 3
  /
 x 3
```
When doing a substitution, it is important to keep in mind that whenever we substitute something for ‘x’ in a certain formula or tree, we need to carry it out simultaneously for every x in that formula / tree. Of course new x’s, i.e., x’s occurring as a result of the substitution, should be left alone and not be substituted again and again, creating an infinite substitution sequence.

The Subst operator can also occur as an argument of an encapsulating Subst operator. This nested occurrence of Subst operators is called repetitive substitution. When considering repetitive substitution we have to distinguish two different kinds, namely sequential substitution and simultaneous substitution. Sequential substitutions are substitutions that are carried out consecutively in the order in which they appear. Simultaneous substitutions are carried out all at the same time, i.e., simultaneously. We will now give an example of a substitution which yields a different result depending on whether it is sequentially or simultaneously executed. Similarly, we also can apply these techniques to substitute within trees.

Example 3.20  Sequential and simultaneous substitutions.

- In Subst (‘a’, ‘y’, Subst (‘y + x’, ‘x’, ‘x’)) the second Subst must be executed first, resulting in ‘(y + x)’’. Now we can carry out the first Subst operator in the formula by executing Subst (‘a’, ‘y’, ‘(y + x)’). The final result of this sequential substitution will be ‘(a + x)’.

- In Subst (‘a’, ‘y + x’), (‘y’, ‘x’), (‘x’)) we have to execute the replacement of ‘y’ by ‘a’ and ‘x’ by ‘y + x’ at the same time, resulting in ‘(y + x)’.

In Example 3.20 we see that the results (and also the notations) of sequential and simultaneous substitutions differ from each other.

So far we only looked (deliberately) at substitution in formulas and trees without binders. Of course, substitutions can also be carried out in formulas and trees that do contain binders. The same rules hold when doing this, though even more caution is required as far
as bound variables and their names are concerned. It seems obvious that we should take extra precautionary measures when we want to substitute a formula containing a free variable, say \( x \), within a formula containing \( x \) also as a bound variable. In such cases we better avoid variable conflicts and replace the bound variable by a fresh variable first, before we carry out the actual substitution. We will now provide an example, which is directly taken from [12], in which we will clarify this method.

**Example 3.21** In Subst \( (k + 1', 'a', \sum_{k=1}^{15}(k^2 + ak + 1)') \) we will first replace the bound variable \( k \) by a variable that does not yet occur, say \( l \). Now the actual substitution can be carried out, resulting in \( \sum_{i=1}^{15}(l^2 + (k + 1) \cdot l + 1)' \).

For the sake of completeness we also provide the tree representation of this substitution in the same way we demonstrated previously in this section.
3.5 Summary

In this chapter, our goal was to learn the basics of formulas. Because formulas are extremely important in mathematics, we have exposed a lot of facets about formulas of which we thought that they would be important. There exist many kinds of notions, conventions, notations and operations for formulas, of which we explained the most important ones in this chapter.

One of the two key parts of this chapter was section 3.3.5, where we introduced the tree representation style to denote a formula, because converting a formula into tree representation offers a lot more insight in the way the formula is built up. We can, for instance, easily deduce arities of formula symbols, see the commonly used style of notation for formula symbols (pre-, in-, post-, circumfix or perhaps otherwise), see which variables are bound where by which binder, etc. This brings us to the second key section in chapter 3: the section about binding in formulas.

Variables in formulas can occur both in free and bound fashion. A binder is a formula symbol that can attach a variable in a formula. All binders have a domain, which is written in subscript and tells us whether the formula following it specifies something about real numbers, natural numbers, rational numbers, etc. In fact, any other domain that we can come up with can be part of the binder.

We pointed out the ins and outs about binding in formulas in detail, such as the tree representation of a formula containing at least one binder, the @-symbol, binding with restrictions and the sum binder. Finally, we have also explained the notion of substitution within formulas and within trees.
4 Principles of Weak Type Theory

4.1 Introduction

Up till now we merely focused on common mathematical aspects related to natural and mathematical language. We explained the differences between natural language and mathematical language and we presented basic notions about formulas, such as formula symbols, formula notations, parentheses, priorities and trees. Moreover, we elaborately discussed how binding in formulas works.

The previous chapters form a perfect prelude, especially for this chapter, because the notions, rules and techniques, as explained mainly in chapter 3, can in most cases be applied in an identical way when WTT is concerned.

So, in this chapter we are going to add the 'WTT flavor' to what we have already learnt. We start in a slow pace by explaining variables and constants in WTT. Subsequently we let you get acquainted with the specific WTT binders that exist and explain how they are used. An important aspect that will be discussed is how to read formulas containing WTT binders and how to translate them into an equivalent natural language expression containing no bound variables and as few mathematical symbols as possible. At the end of each subsection we give the tree representation of one of the formulas that occur in the various examples throughout this chapter.

4.2 Variables

In mathematics the word variable is a frequently used term. A variable usually occurs in the form of a single letter of the Roman or Greek alphabet. In mathematics, variables often represent a quantity that is yet unknown and tends to change, i.e., is variable. Because of the commonality of the term variable, WTT also adopts this term. In WTT, a variable is a mathematical word that occurs as a 'placeholder', filling up 'open places' in a formula. To have a closer look at what variables are, we first look at an example of a formula in which they occur.

Example 4.1 ‘\(x^2 - x\)’ is a formula, in which \(x\) occurs as a variable twice.

In fact, this formula should be read as “The square of some \(x\) minus that same \(x\)”. When we assume that the \(x\) isn’t bound (we will come back to this term later in this section) somewhere before, it doesn’t really matter whatever we pick as a variable in this formula, whether it is the \(x\) as used in the example, or \(y\), or \(\beta\), or even a symbol such as \(\diamond\). The meaning of the formula will not change when another variable is chosen instead of \(x\). So the given formula would provide the same information as, for example, the formulas ‘\(\diamond^2 - \diamond\)’, ‘\(\Omega^2 - \Omega\)’, and so on.
Whatever variable we fill in for this $x$, we will never obtain more specific information about it. This is because the occurrences of $x$ in this formula are free. In other words, $x$ is a free variable here, of which we don't know anything more. As opposed to free variables, bound variables are variables of which a little bit more information is known, namely to which class they belong. They are bound to a particular class (sometimes also a particular domain), excluding elements of other classes (and elements outside the specified domain) to be filled in for it. More on bound variables can be found in [5], which is a discourse in Dutch, containing all kinds of habitual and notational conventions concerning bound variables.

The word 'some' in the textual representation of the formula in Example 4.1 already indicates that we don't know exactly to which class the $x$ belongs. It could be 'some natural number', 'some real number', 'some triangle', etc. Now we give an example of a formula with a bound variable of which we do know to which class it belongs.

**Example 4.2** $\forall_{n \in \mathbb{N}} (n^2 - n \geq 0)$ is a formula in which the occurrences of $n$ are bound to the class $\mathbb{N}$ (the natural numbers).

This formula should be read as “For all natural numbers $n$, $n^2 - n$ is greater than or equal to 0.” The bound variable $n$ here is a natural number, but in fact all kinds of classes could be used here, like $\mathbb{R}$ (the real numbers) or even (if operations such as ‘$-$’ and ‘$^2$’ would be meaningfully defined on classes other than numerical ones) colors, as being the collection of all colors.

In both Example 4.1 and Example 4.2 the same variable occurs twice. It is important to mention that variable names within formulas can best be chosen unique. This uniqueness requires variables, like $x$ and $n$ in the examples 4.1 and 4.2, to be the exact same variables every time they occur in a certain formula. We say these variables are co-referential.

### 4.3 Constants

Variables are often contrasted with constants, which too are very important in mathematical language. Constants are also mathematical words, but are the opposite of variables because they are completely known, as well as unchanging. A completely known constant for instance is $\pi$, of which we know its domain (the real numbers), its meaning (the ratio of a circle's circumference to its diameter) and that it is a well-defined number.

Constants are always followed by a, possibly empty, parameter list, denoted as a list of entities, separated by commas and enclosed in parentheses. The fixed length of this parameter list, which is greater than or equal to 0, indicates the arity of a particular constant, just like we already discussed for the formula symbols in section 3.2.
In WTT, we distinguish five types of constants, namely constants for terms, sets, nouns, adjectives and statements. Let us now look at examples of constants with parameter lists that will clarify these notions.

**Example 4.3** A couple of constants with their parameter list.

- **Constants for terms:** \( \pi, 2 + 7, \sin(x, y, z) \)
  - The constants are: \( \pi, + \) and \( \sin \)
  - The parameter lists are: ( ), (2, 7) and (x, y, z)

- **Constants for sets:** \( Z, A \subseteq B, V \cap W, X^c \)
  - The constants are: \( Z, \subseteq, \cap \) and \( c \)
  - The parameter lists are: ( ), (A, B), (V, W) and (X)

- **Constants for nouns:** an edge of \( \Delta ABC \)
  - The constants are: a triangle and an edge
  - The parameter lists are: ( ) and \( \Delta ABC \)

- **Constants for adjectives:** prime, convex, increasing on \( [a, b] \), injective
  - The constants are: prime, convex, increasing and injective
  - The parameter lists are: ( ), ( ), ([a, b]) and ( )

- **Constants for statements:** \( p \Rightarrow q, 10 < x, l \) goes through \( P \) and \( T \)
  - The constants are: \( \Rightarrow, < \) and goes through
  - The parameter lists are: (p, q), (10, x) and (l, P, T)

Two remarks about this example are worth to make explicitly:

- The numbers '2', '7' and '10' are constant numerical values here. On the places where they occur, variables, like \( a, b \), etc. could also have occurred.

- Prepositions like on, by, with, etc. are used as forms of sugaring which are often omitted in the actual constants. This mainly is the case for constants for nouns and statements. We see that in this example the preposition on in 'increasing on \( [a, b] \)' is omitted here, as well as the preposition of in 'an edge of \( \Delta ABC \)'.

**4.4 WTT binders**

Besides the mathematical binders we discussed in section 3.4.1, there are some specific WTT binders as well. We will mention those and give an example of how to use each of them just like we did with the mathematical binders. We will elaborate more on when to use which WTT binder in chapter 5, where we talk about how to construct formulas. Here
we discuss how to go in the opposite direction, i.e., from formulas, containing WTT binders, to natural language expressing the same as the given formula expresses. The natural language expressions we construct should consist of natural language words as much as possible. They may not contain any bound variables and as few mathematical symbols as possible.

The binders that are specifically defined in WTT are Noun, Adj, Abst, \( \iota \) (the Greek letter ‘iota’), and \( \lambda \) (the Greek letter ‘lambda’). They are used in similar fashion as the mathematical binders we saw earlier, as far as binding is concerned. This means that the conventions about binding, trees with binders and substitution are adopted in WTT without any adaptations.

For each of these binders we provide a short and simple example in Table 4.1 that gives an indication of what they look like and how they are used properly.

**Example 4.4** Overview of the five WTT binders and their use.

<table>
<thead>
<tr>
<th>Binder symbol</th>
<th>Example</th>
<th>Bound variable(s)</th>
<th>Free variable(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Noun</td>
<td>( Noun_{x \in \mathbb{R}}(x &gt; m) )</td>
<td>( x )</td>
<td>( m )</td>
</tr>
<tr>
<td>( \iota )</td>
<td>( \iota_{n \in \mathbb{N}}(n \geq 0 \land n \leq 0) )</td>
<td>( n )</td>
<td>-</td>
</tr>
<tr>
<td>Adj</td>
<td>( Adj_{n \in \mathbb{N}}(n &lt; 0) )</td>
<td>( n )</td>
<td>-</td>
</tr>
<tr>
<td>Abst</td>
<td>( Abst_{y \in \mathbb{R}}(y^4 + 2) )</td>
<td>( y )</td>
<td>-</td>
</tr>
<tr>
<td>( \lambda )</td>
<td>( \lambda_{x \in \mathbb{Z}}(x^3) )</td>
<td>( x )</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 4.1 WTT binders binding variables in formulas

We will devote a separate section to each kind of binder in order to elaborate on their use and meaning in a structured and more detailed way.

### 4.4.1 The Noun binder

The Noun binder is used, as its name already states, to describe nouns. As we recall the Noun formula in Example 4.4, a real number with a certain property is described. The property in this case is "being bigger than \( m \)."

We have to be careful with nouns though, because not just any noun is described by the noun binder. It only specifies *generic nouns*, which are also called *indefinite nouns*. Generic nouns are nouns that describe an arbitrary object of a certain class. For instance, we can think of the phrase ‘*a circle*’ as a generic noun, because it describes an arbitrary object as a representative of the class of all circles. Other examples would be ‘*a book*’, ‘*a real number*’ and ‘*a natural number of which the square is at least 100*’. 

- 45 -
These generic noun examples 'generate' arbitrary objects of various classes. Those generated objects have the standard properties of the class they belong to. For instance, the phrase 'a circle' generates an object of the class of all circles, of which the circumference is equal to $2\pi \cdot r$, where $r$ is the radius of the circle. This holds for any circle. On the other hand, the arbitrariness of the object also means that we cannot say anything specific about it. For instance, we cannot say anything about what its radius exactly is.

In mathematical language there is a need to assign a name to the arbitrary objects that are generated. Such a generic name actually is the variable that is bound in the formula. Assigning a name to a generated object of a certain class offers a convenience: it makes the co-references more explicit. Example 4.5 will point this out.

**Example 4.5** Assigning a generic name $t$ to the generic noun appearing in the sentence.

"If a triangle is an isosceles triangle, then it has two equally sized angles; moreover, then there exists a median in that triangle which also is the bisector and perpendicular of that triangle."

leads to the following simplified sentence, in which the co-references are more clear than before:

"If a triangle, say $t$, is an isosceles triangle, then $t$ has two equally sized angles; moreover, then there exists a median in $t$ which also is the bisector and perpendicular of $t"."

Translating a formula containing a Noun binder to an equivalent natural language expression with no bound variables and as few mathematical symbols as possible can best be done by using the indefinite article 'a', as in "a ... such that ..". Instead of the 'such that' part we can also choose 'of which', or 'for which', or some other similar construction, whichever suits best to express a particular formula in natural language in a correct way.

The first sequence of dots in the above introduced way of constructing sentences is replaced by the class of the generated object; instead of the second sequence of dots, appearing after 'such that', the actual formula will appear in natural language. We will now give examples of how to translate a few formulas in which the Noun binder occurs.
Example 4.6  Translations of formulas with Noun binders. All of them start with the indefinite article ‘a’ to specify that it describes a generic noun.

<table>
<thead>
<tr>
<th>Noun formula</th>
<th>Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>( r := \text{Noun}_{x \in \mathbb{R}}(x &lt; x^2) )</td>
<td>“( r ) is defined as a real number that is smaller than its square.”</td>
</tr>
<tr>
<td>( \text{Noun}_{n \in \mathbb{N}}(n^2 \geq 100) )</td>
<td>“A natural number of which the square is at least 100.”</td>
</tr>
<tr>
<td>( \text{Noun}_{x \in (0, 2\pi)}(\sin(x) &lt; \cos(x)) )</td>
<td>“A real number between 0 and 2( \pi ), of which the sine is smaller than the cosine.”</td>
</tr>
<tr>
<td>( \text{Noun}_{n \in \mathbb{N}}(\exists k \in \mathbb{Z} \mid 1 &lt; k &lt; n \quad (n \text{ is divisible by } k)) )</td>
<td>“A natural number for which there exists an integer between 1 and that natural number, such that that same natural number is divisible by that integer.”</td>
</tr>
</tbody>
</table>

Table 4.2 Noun formulas translated into natural language equivalents

The first formula we see here is assigned a generic name, namely \( r \). We will encounter this way of using formulas more often throughout the remainder of this course. The middle two entries in Table 4.2 are quite straightforward ones, but the last one shows a nested formula starting with a Noun binder, which in turn has an existential quantification as its body. These nested constructions appear very often in WTT, where all kinds of mathematical and WTT binders can be nested within each other.

To show how the Noun binder appears in the tree representation of a formula, we provide the tree representation of the second formula in Example 4.7.

Example 4.7  \( \text{Noun}_{n \in \mathbb{N}}(n^2 \geq 100) \)

![Figure 4.1 Tree representation of a formula containing a Noun binder](image)
4.4.2 The $t$ binder

In section 4.4.1 we got familiar with the Noun binder and we know now that it is used to describe generic nouns. The $t$ binder also describes nouns, just as the Noun binder. The main difference is that it describes specific nouns rather than generic nouns.

When compared to generic nouns, specific nouns are in fact their counterpart. A specific noun describes a specific, unique object of a certain class. For example, if we assume $C$ to be a predefined circle, the phrase ‘the center of $C$’ describes a unique object, because $C$ only has one centre, just as any circle. Another example would be the specific description “The natural number between 2 and $\pi$”. We know that this can only be the natural number 3, because 3 is the only existing natural number within the given boundaries. We know that 3 is an odd number, a prime number and a threefold as well, i.e., of a specific noun we know specific properties.

The definite article ‘the’ is perfectly suitable to translate formulas containing the $t$ binder into natural language. It distinguishes between ‘some’ object of a class and ‘that specific’ object of a class. Therefore a formula containing the $t$ binder can be translated into an equivalent natural language expression using the construction “the … such that …”. The gaps in this construction are filled in in the exact same way as described in section 4.4.1.

We will now give a few examples of formulas in which the $t$ binder appears. We will also give an appropriate translation of all these formulas.

Example 4.8 Translations of formulas with $t$ binders into natural language equivalents. All of these translations start with the definite article ‘the’ to make explicit that they describe a specific noun.

<table>
<thead>
<tr>
<th>$t$ formula</th>
<th>Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_{n \in \mathbb{N}} (n^3 - 2n^2 = \frac{1}{4} n - \frac{1}{2})$</td>
<td>“The natural number of which its third power, subtracted by 2 times its square, is equal to one fourth that natural number, subtracted by one half.”</td>
</tr>
<tr>
<td>$t_{p: a \text{ point}} (\text{dist}(A, P) = \text{dist}(B, P) = \text{dist}(C, P))$</td>
<td>“The point which has a distance to point $A$ that is equal to its distance to point $B$ and to its distance to point $C$.”</td>
</tr>
<tr>
<td>$t_{f: \mathbb{N} \rightarrow \mathbb{N}} (\forall n \in \mathbb{N} (f(n) = 2n + 1))$</td>
<td>“The function on the natural numbers for which holds that for all natural numbers the result of that function applied to such a natural number is equal to 2 times this natural number plus 1.”</td>
</tr>
</tbody>
</table>
| $t_{x \in (0, 2\pi)} (\sin(x) + \cos(x) = \sqrt{2}) < \pi$ | “The real number between 0 and $2\pi$, for which holds that the sum of its sine and
its cosine is equal to the square root of 2, is smaller than \( \pi \).

Table 4.3 \( t \) formulas and their translations into natural language

The first formula in Table 4.3 is quite straightforward. In the second formula we assume that the binary function \( \text{dist} \) is predefined as a function computing the distance between two points that are given as arguments. Moreover, we assume that the points \( A, B \) and \( C \) are predefined points that are all different from each other. Finally, we see that in the fourth formula a specific noun is defined, which we immediately compare to the concrete value \( \pi \). This can be done because the specified noun is a specific, and thus unique, real number, viz. \( \frac{1}{2} \pi \).

Furthermore, we see that the translations into natural language become more laborious and cumbersome when the formula increases in length and difficulty. This cannot be prevented, yet we can emphasize the way the formula should be read by allowing parentheses in natural language to serve the same purpose as they serve in formulas, namely to stress priorities and to group words that belong together. However, it depends on one’s personal preferences whether this is experienced as clarifying or not. We could for instance write the third translation in Table 4.3 as follows: “The function on the natural numbers for which holds that for all natural numbers, (the result of that function (applied to such a natural number)) is equal to ((2 times this natural number) plus 1).” Therefore, we leave it to the reader to experiment with the use of parentheses in natural language himself.

Example 4.9 shows the tree representation of the second formula of Table 4.3 that we have seen already. To represent this formula with a tree in a correct way, we have to be aware of the fact that this formula is a shorthand notation. When we write it out we get: 
\[ t_r. \text{a point} \left( (\text{dist}(A, P) = \text{dist}(B, P)) \land (\text{dist}(B, P) = \text{dist}(C, P)) \right), \] of which the tree is shown here.
Example 4.9  \( lp:apoint ((dist(A, P) = dist(B, P)) \land (dist(B, P) = dist(C, P))) \)

![Tree representation of a formula containing the \( t \) binder](image)

**Figure 4.2 A formula containing the \( t \) binder in tree representation**

### 4.4.3 The Adj binder

The binder \( Adj \) is an adjectival binder, i.e., it gives a WTT description of an adjective. There are three important categories of adjectives which we will look into in more detail. These are *coordinating adjectives*, *subordinating adjectives* and *fixed 'adjective-noun' combinations*. We will explain these categories one by one and give examples of them.

Coordinating adjectives are two or more adjectives that all refer to the same noun but are independent of each other, which justifies their name. The adjectives are in front of the noun they are referring to and often appear in the form of a list, separated with commas.

To check whether an adjective is a coordinating adjective, those commas can be replaced by the coordinating conjunction *and*. If the result of this replacement is meaningful, i.e., if the two adjectives are still independently referring to the noun, then the adjectives are said to be coordinating adjectives. In Example 4.10 we can see some examples of coordinating adjectives.

**Example 4.10** Coordinating adjectives, independently referring to the same noun.

- In the phrase “even, positive number” both adjectives refer to the noun
'number' and they are independent of each other, as indicated by the arrows.

- In the phrase "an old, neglected, shabby house" all adjectives refer to the noun 'house'.

The category of subordinating adjectives is slightly different, as the following example will show:

**Example 4.11** Subordinating adjectives and their structure of referral.

- In the phrase "least common multiple" we see that the adjective 'common' refers to the noun 'multiple' and the adjective 'least' refers to 'common multiple' as a whole.

Generally, when two subordinating adjectives appear in front of a noun, then the adjective closest to the noun refers to this noun, whereas the other adjective refers to the combination of the previous two.

Finally, the category of the fixed adjective-noun combinations contains adjective-noun pairs which cannot be separated, because the adjective has become very strongly associated with the noun it accompanies. So, as a pair they have become a discernible notion.

**Example 4.12** Adjective-noun pairs that have become common mathematical notions of their own.

- In the phrase 'primitive function' the adjective 'primitive' does not refer to the noun 'function', because their relation is much stronger. This pair is always considered as being a mathematical notion as a whole.

- In the phrase 'square root' it will be clear that the root itself is not 'square'. Obviously, these words cannot be separated from each other and therefore form another adjective-noun pair.
We have already learnt in this section that there are three different categories of adjectives in mathematical language. Now we will see how the Adj binder is used to specify adjectives in WTT.

An adjective binder can be encountered in a WTT formula in the same way as we have already seen in Example 4.6: in the form of an assignment or in its 'regular' form, i.e., a plain formula without assignment.

We will show some examples of formulas containing the Adj binder now. After we have seen these examples we will make some additional remarks about them.

**Example 4.13** Formulas containing the Adj binder, together with their translations. The last two formulas contain an assignment to a linguistic adjective.

<table>
<thead>
<tr>
<th>Adj formula</th>
<th>Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{Adj}_{s \in \mathbb{R}} (x &lt; 0) )</td>
<td>&quot;The adjective saying of a real number that it is smaller than 0.&quot;</td>
</tr>
<tr>
<td>( \text{Adj}_{\text{a triangle}} \exists_1, s_2: \text{a side of} \ (s_1 \neq s_2 \land</td>
<td>s_1</td>
</tr>
<tr>
<td>( \text{even} := \text{Adj}_{f: \mathbb{R} \rightarrow \mathbb{R}} \forall x \in \mathbb{R} \ (f(x) = f(-x)) )</td>
<td>&quot;A function from ( \mathbb{R} ) to ( \mathbb{R} ) is called even if for every real number the result of that function, applied to that real number, is equal to the result of that function, applied to the opposite of that real number.&quot;</td>
</tr>
<tr>
<td>( \text{symmetric} := \text{Adj}_{R \subseteq V^2} \forall x \in V \forall y \in V \ ((x, y) \in R \Rightarrow (y, x) \in R) )</td>
<td>&quot;A relation on ( V^2 ) is called symmetric if for every two elements of ( V ) holds that if the pair, consisting of these two elements belongs to the relation, then also the pair, consisting of the second and first element of the previously mentioned pair, belongs to the relation.&quot;</td>
</tr>
</tbody>
</table>

**Table 4.4** Two different styles of formulas containing the Adj binder

The first formula is rather straightforward when reading the translation. The second formula, however, is less easy to read. In fact, its translation is a laborious way to describe the adjective *isosceles*, which we could have made more obvious by directly defining this linguistic adjective to the formula.

The third and fourth formula both make use of the technique of defining linguistic adjectives. We see that these formulas are much more readable and easier to translate.
because the correct adjective is explicitly mentioned and assigned to the formula behind the Adj binder here.

As demonstrated in Table 4.4, the translation of the first two Adj formulas can best be started with the natural language construction "The adjective saying of ... that ...", where the dotted sequences are subsequently filled in with the domain and the part of the formula behind the Adj binder.

The last two formulas in Table 4.4 can be translated into natural language using the following construction: "A ... is called ... if ...". The first dotted sequence in this construction is filled in by the domain that comes with the Adj binder, whereas the second sequence of dots is replaced by the name of the adjective that is defined. Finally, the translation of the formula behind the Adj binder will be placed on the last sequence of dots.

In the introduction of section 4.4 we stated that none of the WTT binders causes alterations to the way in which we represent formulas by trees. We illustrate this by showing how the third formula of Table 4.4 can be transformed into tree representation in Example 4.14.
Example 4.14 \(\text{even} := \text{Adj}_{f: \mathbb{R} \to \mathbb{R}} \forall x \in \mathbb{R} \ (f(x) = f(-x))\)

Figure 4.3 Tree representation of a formula containing the Adj binder

4.4.4 The \(\lambda\) binder

The WTT binder \(\lambda\) is used to specify both predicates and functions. So in fact it plays a double role in weak type theory. In this section we will discuss both roles in detail, starting with the \(\lambda\) binder in its role as function binder.

The \(\lambda\) as function binder

To make clear what we will consider as a function, we will define the notion of function first. We consider a function to be a mapping from a domain to a co-domain, which both are collections of objects. This mapping is carried out according to a specified description that tells us how objects of the domain are connected to objects of the co-domain. The objects from the domain and co-domain are also called inputs and outputs of the functions, respectively.
When we look at different mathematical textbooks, we see that there are a lot of notations for functions. A few of the forms in which they occur are displayed in Example 4.15.

**Example 4.15** Various notations for the function mapping $n$ to $n^2$, $(n \in \mathbb{Z})$. Also the comments or objections to each of these notations are given.

<table>
<thead>
<tr>
<th>Proposed function notation</th>
<th>Comments or objections</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n^2$</td>
<td>This is no function but a value of an unspecified random $n$.</td>
</tr>
<tr>
<td>square</td>
<td>The domain and co-domain $\mathbb{Z}$ are left implicit here.</td>
</tr>
<tr>
<td>$f : n \mapsto n^2$</td>
<td>The domain and co-domain $\mathbb{Z}$ are left implicit here.</td>
</tr>
<tr>
<td>$f(n) = n^2$</td>
<td>The domain and co-domain $\mathbb{Z}$ are left implicit here. Moreover, this does not specify a function but an equality.</td>
</tr>
<tr>
<td>$f : \mathbb{Z} \to \mathbb{Z} : n \mapsto n^2$</td>
<td>Correct and complete, but a rather extensive notation.</td>
</tr>
<tr>
<td>$f : \mathbb{Z} \to \mathbb{Z}$ with $\forall_{n \in \mathbb{Z}} (f(n) = n^2)$</td>
<td>Correct and complete, but a rather extensive and difficult notation.</td>
</tr>
</tbody>
</table>

Table 4.5 All kinds of function notations occur in mathematical literature; some more convenient than others

The notation '$f : \mathbb{Z} \to \mathbb{Z} : n \mapsto n^2$' actually is the best one of those we have seen above. It is both correct and complete. The downside of it is that it is quite laborious to write out a function in this way every time. To overcome this inconvenience, we introduce the $\lambda$ binder (Greek letter lambda; this $\lambda$ notation is derived from lambda calculus, developed by the logician A. Church) to bind the variable $n$ to the domain $\mathbb{Z}$ in this particular case:

$$\lambda_{n \in \mathbb{Z}} (n^2)$$

We see that this formula has become much shorter. The co-domain, which is also $\mathbb{Z}$ in this case, is left out because in $\lambda$-notation it is considered to be irrelevant, which is indeed a valid assumption in lots of applications. Moreover, we don’t see the name of the function appear in this notation. In WTT this can be ‘repaired’ by explicitly assigning a name to the formula with the $\lambda$ binder as follows:

$$f := \lambda_{n \in \mathbb{Z}} (n^2)$$
When translating functions specified with the $\lambda$ binder into natural language, we make use of the following construction which contains two dotted sequences: "The function, sending ... to ...". The first sequence of dots is replaced by the domain of the function, whereas the second sequence is filled in with the actual formula behind the $\lambda$ binder. In Example 4.16 some formulas containing the $\lambda$ binder are translated. We notice that after the $\lambda$ binder, i.e., between the parentheses, an object is specified. The object that is bound in the domain is ‘sent to’ this object.

**Example 4.16** Function descriptions with the $\lambda$ binder, including the corresponding translations.

<table>
<thead>
<tr>
<th>$\lambda$ formula</th>
<th>Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda_{x \in \mathbb{R}_{\geq 1}} (\sqrt{x - 3})$</td>
<td>&quot;The function sending a real number greater than or equal to three to the square root of (that real number decreased by 3).&quot;</td>
</tr>
<tr>
<td>$\lambda_{p: \text{a prime number}} (q^{p-1})$</td>
<td>&quot;The function sending a prime number to $q$ to the power (that prime number minus 1).&quot;</td>
</tr>
<tr>
<td>$f := \lambda_{x \in (0, \pi)} (\tan (x - \frac{\pi}{2}))$</td>
<td>&quot;By definition, $f$ is the function, sending a real number between 0 and $\pi$, to the tangent of (that real number, decreased by the quotient of $\pi$ and 2).&quot;</td>
</tr>
<tr>
<td>$h := \lambda_{n \in \mathbb{N}} (\sum_{i=0}^{n} (x^i + x = 4))$</td>
<td>&quot;By definition, $h$ is the function, sending a natural number to the positive real number for which holds that the sum of (that real number to the power that natural number) and (that real number itself) is equal to 4.&quot;</td>
</tr>
</tbody>
</table>

**Table 4.6** The $\lambda$ binder, used as a function binder

Though the use of the $\lambda$ binder only seems to have advantages, there also is a disadvantage that we need to mention. If we want to specify a formula that maps the domain to the co-domain in various ways, depending on the specific input value, we unfortunately cannot use the $\lambda$ binder. Let’s have a look at the following function, which is specified differently for various domains of input values:

\[
\begin{align*}
  f : \mathbb{R} &\rightarrow \mathbb{R} : \\
  &\begin{cases}
    x \mapsto x^2, &\text{for } x \leq 0 \\
    x \mapsto x, &\text{for } x > 0
  \end{cases}
\end{align*}
\]
A first guess would probably be something like

\[
\lambda_{x \in \mathbb{R}} \begin{cases}
  x^2, & \text{for } x \leq 0 \\
  x, & \text{for } x > 0
\end{cases},
\]

but in WTT it is not allowed to specify a \( \lambda \) binder in this way. So, without introducing additional syntactical constructions in WTT, there is no suitable way to express this formula with the \( \lambda \) binder.

Other mathematical notions can also be represented in WTT using the \( \lambda \) function binder. The most important one is the notion of (infinite) sequence. A sequence can be considered as a function with domain \( \mathbb{Z}^+ \) and co-domain \( \mathbb{R} \); in other words \( \mathbb{Z}^+ \rightarrow \mathbb{R} \).

When we want to specify the infinite sequence \( 'a_1, a_2, a_3, \ldots' \), we can do this with the \( \lambda \) binder: \( \lambda_{n \in \mathbb{Z}^+} (a_n) \). Sometimes in mathematics a sequence is defined to start at \( a_0 \). In this case, we consider a sequence as a function from \( \mathbb{N} \rightarrow \mathbb{R} \) (because \( 0 \in \mathbb{Z}^+ \), but \( 0 \notin \mathbb{N} \)).

**The \( \lambda \) as predicate binder**

We briefly mentioned the double role the \( \lambda \) binder plays in WTT in the beginning of this section. Now we will go into detail about its second role, i.e., the role of predicate binder.

We often encounter adjectives to indicate properties of certain objects (also see section 4.4.3 discussing the Adj binder). An example of this is the sentence "the natural number 14 is a composite number". The adjective 'composite' indicates here that the positive natural number 14 has factors other than just 1 and the number itself: \( 14 = 2 \cdot 7 \). However, we do not have a good name for this property, or it would be something like 'compositeness'.

In mathematical language we can transform our self-invented adjective 'compositeness' into a predicate using the principle of variable binding together with the \( \lambda \) binder. This leads to the following result: "the property of the natural number \( n \), that \( n \) is composite". In this description it is obvious that \( n \) is a bound variable (the natural number 14 in the previously described situation). The formula belonging to this description becomes \( \lambda_{n \in \mathbb{N}} (n \text{ is composite}) \).

Another possible way to express 'compositeness' in natural language is the following: "The predicate that holds for a natural number \( n \) if \( n \) is composite". In this translation we explicitly make use of the word 'predicate' that emphasizes which role the \( \lambda \) binder is playing here.
We will give some examples now showing how predicates containing the $\lambda$ binder can be translated into natural language equivalents. We see that we can explicitly assign a name to a predicate containing the $\lambda$ binder, just as we did in Example 4.16. This allows us to refer to those predicates by only mentioning their assigned name.

**Example 4.17** Predicates with the $\lambda$ binder with an appropriate translation into natural language.

<table>
<thead>
<tr>
<th>$\lambda$ formula</th>
<th>Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda_{n \in \mathbb{N}} \exists_{k \in \mathbb{N}} (n = k^2)$</td>
<td>“The predicate that holds for a natural number if there exists a second (possibly the same) natural number of which the square root is equal to that first natural number.”</td>
</tr>
<tr>
<td>$\lambda_{A \subseteq \mathbb{N}} \exists_{p: \text{a prime number} \mid p &gt; 10} (p \in A)$</td>
<td>“The predicate that holds for a subset of $\mathbb{N}$ if there exists a prime number greater than 10 that is an element of that subset.”</td>
</tr>
<tr>
<td>$P := \lambda_{n \in \mathbb{N} \mid n \neq 0} (n \text{ is even } \land \neg (n^3 &gt; 100))$</td>
<td>“$P$ is the predicate that holds for a natural number, which is not equal to 0, if it is even and its third power is not greater than 100.”</td>
</tr>
<tr>
<td>$Q := \lambda_{n \in \mathbb{N}} \exists_{m \in \mathbb{N}} (</td>
<td>m^2 - n</td>
</tr>
</tbody>
</table>

Table 4.7 Predicates using the $\lambda$ binder, each guided with their translation into natural language

Now that we have seen both applications of the $\lambda$ binder, we can point out the essential difference between them. When in its role as a function binder, an object is specified behind the $\lambda$ binder. When the $\lambda$ binder is encountered as a predicate binder, we will find a statement behind it.

Of course, the formulas of Table 4.7 can be converted to their tree representation just as we did previously in chapter 3. We will show the tree representation of the last formula of this table below:
Example 4.18 \[ Q := \lambda_{m \in \mathbb{N}} \exists_{n \in \mathbb{N}} (|m^2 - n| < 2) \]

Figure 4.4 Tree representation of a formula containing the \( \lambda \) binder

4.4.5 The Abst binder

In natural language as well as in mathematical language, there are situations conceivable where a generic noun depends on another generic noun. For example, we can think of the generic noun 'a CD', which describes an arbitrary object of the class of all CDs, whereas 'a track of a CD' describes a new generic noun, which is dependent on the first one. A 'CD-track' actually is 'a track of some CD', so it describes an object from the class of all tracks of all CDs. A similar reasoning can be followed to conclude that a 'square-multiple' describes 'a square of some multiple'.

To have the possibility to describe such interdependent generic nouns in WTT, we introduce another binder, namely the Abst binder. Abst literally stands for *abstraction* but is also called *despecification* sometimes, because abstraction is the opposite of specification, indicated by the linguistic prefix 'de-'.

- 59 -
It is not only possible to abstract from generic nouns using the Abst binder, but abstraction from specific nouns is also possible. To do this, these specific nouns must depend on each other too. For instance, when we talk about ‘a CD-title’, we mean ‘the title of some CD’, and with ‘a circle-disc’ we mean ‘the disc of some circle’.

When translating a formula containing an Abst binder into natural language, it is noteworthy to point out that in order to make a correct translation, we usually have to look at the part behind the Abst binder first. Only then we look at what kind of variable is bound by the Abst binder. In Example 4.19 we will see this technique when applied to a few WTT formulas containing the Abst binder.

Example 4.19  Formulas containing the Abst binder with their translations.

<table>
<thead>
<tr>
<th>Abst formula</th>
<th>Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{Abst}_{n \in \mathbb{N}} \ln(n)$</td>
<td>&quot;The natural logarithm of some natural number greater than 20.&quot;</td>
</tr>
<tr>
<td>$\text{Abst}<em>{a \in \mathbb{Z}} \text{Abst}</em>{q \in \mathbb{N}^+}(\frac{a}{q})$</td>
<td>&quot;The quotient of some integer and some positive natural number.&quot;</td>
</tr>
<tr>
<td>$\text{Abst}_{n \in \mathbb{Z}} ({x \in \mathbb{R}</td>
<td>n &lt; x &lt; n + 1})$</td>
</tr>
<tr>
<td>$\text{Abst}<em>{r \in \mathbb{R}} \text{Noun}</em>{a \subseteq \mathbb{R}^+} \forall r \in A (x &lt; s)$</td>
<td>&quot;A subset of the positive real numbers for which holds that all elements of this subset are smaller than some real number.&quot;</td>
</tr>
</tbody>
</table>

Table 4.8 Formulas with the abstraction binder

The first entry in the table of Example 4.19 is the most straightforward one. We see that the Abst binder has the natural numbers greater than 20 as its domain. This means that the whole formula has something to do with some natural number greater than 20. After the Abst binder we see an object occur. This is a definite object, because each natural number has only one natural logarithm. Combining these arguments, we have justified that translation of the first formula results in "the natural logarithm of some natural number greater than 20".

The second formula in the table of Example 4.19 shows that Abst binders can be nested within each other, in fact meaning that we abstract from (in this case) both the integers and the positive natural numbers. In the third formula we see another way in which we can encounter an Abst formula, namely with a set behind the Abst binder rather than an object. Translation works similar to the two cases we already explained here.
The last formula shows a frequently occurring combination of binders, i.e., the Abst - Noun combination. As we already explained in the paragraph prior to Example 4.19, we have to look at the last part of the formula first, meaning that we first have to look at the Noun part. This part specifies a subset of the positive real numbers of which all elements are greater than $s$. This $s$ is the bound variable we abstract from and thus the first part of the formula needs to be translated as "some real number". The whole translation becomes: "A subset of the positive real numbers for which holds that all elements of this subset are smaller than some real number." Note that the Noun binder is responsible for the fact that the translation into natural language starts with 'A subset …' instead of 'The subset …'.

The Abst binder is embedded into our tree representation just like any other binder. We show this by giving the tree representation of the fourth formula of Table 4.8 in Example 4.20.

**Example 4.20** \[ \text{Abst}_{x \in \mathbb{R}} \text{Noun}_{A \subseteq \mathbb{R}} : \forall x \in A (x < s) \]

![Tree representation of a formula containing, among others, the Abst binder](image)

In Example 4.19 we got acquainted with all possible ways in which an Abst binder can be used without explicitly mentioning it. For completeness' sake we conclude by saying that the Abst binder can be applied to either an object, or a set, or a Noun, as the attentive reader would have already noticed.
4.5 Special constants

Besides the constants that came up in section 4.3, we already briefly mentioned that WTT is equipped with two special constants. These constants are \( \uparrow \) and \( \downarrow \), which are called generalization and specification, respectively.

We already described in section 4.4.1 how a formula containing a Noun binder could be translated best to natural language. We also mentioned there, that each generic noun generates an object of a certain class. Such a class of a generic noun we will call a generic type from now on. The constant \( \uparrow \) is used to generalize a generic noun to its generic type, thus it takes a generic noun as its argument and generalizes it to its generic type. The constant \( \downarrow \) takes care of the opposite direction, i.e., it takes a generic type as its argument and specifies it to a generic noun that describes a generic object of that type. The constants \( \uparrow \) and \( \downarrow \) are unary postfix constants, thus occurring behind their only argument.

When generalizing a generic noun, the appropriate natural language construction to use would start with 'The set of ...'. When using specification on a generic type, the proper natural language construction to use would start with the indefinite article 'a'.

By means of Example 4.21 we will show how the generalization and specification constants should be interpreted and/or translated into natural language.

**Example 4.21** Special constants for generalization and specification used in WTT formulas, together with their interpretation/translation into natural language.

<table>
<thead>
<tr>
<th>Formula</th>
<th>Interpretation / Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>((a \text{ positive divisor of 8}) \uparrow)</td>
<td>&quot;The set of positive divisors of 8.&quot; ((= {1, 2, 4, 8}))</td>
</tr>
<tr>
<td>((a \text{ sphere in } \mathbb{R}^3 \text{ with radius 7}) \uparrow)</td>
<td>&quot;The set of spheres in ( \mathbb{R}^3 ) with radius 7.&quot;</td>
</tr>
<tr>
<td>((a \text{ real number}) \uparrow)</td>
<td>&quot;The set of real numbers.&quot; ((= \mathbb{R}))</td>
</tr>
<tr>
<td>((\mathbb{N}^2) \downarrow)</td>
<td>&quot;A pair of natural numbers.&quot;</td>
</tr>
<tr>
<td>((1, \infty) \downarrow)</td>
<td>&quot;A real number greater than 1.&quot;</td>
</tr>
<tr>
<td>({ x \in \mathbb{R} \mid x^2 &gt; 13 }) (\downarrow)</td>
<td>&quot;A real number which has a square greater than 13.&quot;</td>
</tr>
</tbody>
</table>

Table 4.9 The application of \( \uparrow \) and \( \downarrow \) demonstrated by some examples

Since we already indicated that both special WTT constants discussed here are unary postfix constants, it will not be difficult to incorporate them into our tree representation. The same strategy as for, for instance, the factorial operator '!' can be applied, meaning
that the $\uparrow$ and $\downarrow$ constants appear on the right hand side of a closed node and have only one branch downward.

Because we haven't seen either $\uparrow$ or $\downarrow$ occurring in a tree representation so far, we will convert the last formula of Table 4.9 into its tree representation now:

**Example 4.22** \( \{ x \in \mathbb{R} \mid x^2 > 13 \} \downarrow \)

![Figure 4.6 Specification constant embedded in the tree representation of a WTT formula](image)

**4.6 Summary**

In this chapter, we experienced the basic principles of WTT for the first time. We added a 'WTT-flavor' to the knowledge about natural language, mathematical language and formulas that we gained in the chapters 2 and 3.

Various WTT aspects have been discussed, such as the notions of variables and constants, but we mainly focused on the specific WTT binders. These have been invented to be able to map aspects of natural language, such as nouns, adjectives and abstractions, onto mathematical language.

The structure of formulas containing WTT binders has been explained, as well as the various different appearances in which we can encounter WTT binders in formulas. The translation into natural language has also been specified for numerous examples of formulas with WTT binders. In this way we developed a good feeling of how to translate each kind of WTT formula into natural language. The WTT binders can be embedded in our tree representation as well. We have shown numerous examples of how to do that throughout this chapter.
Finally we explained the two special constants $\uparrow$ and $\downarrow$ which fulfill an important role in WTT. They make it possible to 'switch' back and forth between nouns on the one hand and types on the other hand.
5 Constructing WTT formulas

5.1 Introduction

In chapter 4 we have gotten familiar with WTT. We increased our knowledge about WTT in a unidirectional way. This means that up till now we only shed light on translation from a WTT formula to a natural language construction that expresses the same. Of course we can also do it the other way around, taking natural language expressions as our starting point and mould these into a correct WTT formula. This chapter elaborates on how we can do this and on which difficulties are concerned with it.

In the previous paragraph we said something that should have triggered our minds by now. Especially when we recall that WTT heavily relies on the smallest details when the use of language is considered. In WTT, it really makes a difference whether we use a definite article ('the') or an indefinite article ('a', 'an') to specify a certain noun. Therefore, the phrase 'a correct WTT formula' that we used in the first paragraph of this section, implies that natural language constructions could very well have more than one correct translation in WTT. This is true in general. Whichever correct translation is chosen mostly depends on personal preference for a particular format. We will come back to this at the end of this chapter.

In this chapter, we discuss the different WTT binders again, in the same order as we discussed them in chapter 4. We chose for this order to stress the strong link these two chapters have with each other. We approach the WTT binders from the opposite direction now, i.e., we choose natural language as our starting point to construct WTT formulas that are identical in their meaning.

5.2 Generic nouns

In section 4.4.1 we introduced the term generic noun. This is a noun that specifies an arbitrary object of a certain class, such as 'a television', 'a bisector' or 'a real number'. It is left implicit which television, bisector or real number is actually meant, but we want to say something about an object of these classes in general.

In natural language we encounter generic nouns, or indefinite nouns, practically everywhere. In this course text for example, from chapter 1 up to where we are now, we encounter the indefinite articles 'a' and 'an' over 700 times together! To transform a natural language expression containing a generic noun into a WTT formula, the Noun binder is used. The way in which it is constructed is as follows:

\[ Noun_0 (S) \]
where 'D' stands for 'Domain' and 'S' stands for 'Statement'. This means that a Noun is always succeeded by a statement that is either true or false.

The domain must give a generic name to the generic noun, just as we previously did in, for example, 'x ∈ ℝ' and 'p : a prime number'. The x and the p that are bound in this way will occur in the Statement part as a reference to the generic noun.

We will now give some examples in natural language which are translated into WTT formulas containing the Noun binder. After that we will discuss those examples in more detail.

Example 5.1 Expressions in natural language that are translated into a corresponding WTT formula that contains a Noun binder.

<table>
<thead>
<tr>
<th>Natural language</th>
<th>Corresponding WTT formula with Noun binder</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;A circle with center P.&quot;</td>
<td>( \text{Nounc}_C::\text{circle} \ (\text{the \ center \ of \ } C = P) )</td>
</tr>
<tr>
<td>&quot;A real number of which its square is a rational number.&quot;</td>
<td>( \text{Nounc}_{\mathbb{R}} \ (r^2 \in \mathbb{Q}) )</td>
</tr>
<tr>
<td>&quot;An integer greater than 2 but not greater than 5.&quot;</td>
<td>( \text{Nounc}_{\mathbb{Z}} \ (i &gt; 2 \land \neg(i &gt; 5)) )</td>
</tr>
<tr>
<td>&quot;A real number that is the square root of a natural number.&quot;</td>
<td>( \text{Nounc}<em>{\mathbb{R}} \exists</em>{\mathbb{N}} (x = \sqrt{n}) )</td>
</tr>
<tr>
<td>&quot;An element of the sequence ((a_n)_{n \in \mathbb{N}})&quot;</td>
<td>( \text{Nounc}<em>{\mathbb{R}} \exists</em>{\mathbb{N}} (x = a_n) )</td>
</tr>
</tbody>
</table>

Table 5.1 WTT formulas specifying generic nouns, thus containing the Noun binder

The first sentence we see in Table 5.1 is fairly short and simple. When we want to translate it into a Noun formula, we have to take several things into account. First, we notice that the generic noun that is described here is 'a circle', which we have to give a generic name in the Noun formula in order to be able to refer to it. Second, since a circle has only one unique center and the notion center is not mathematically defined here, we can suffice by equating 'P' with 'the center of C', without mathematically specifying any characteristics about centers of circles.

The second and third entry in Table 5.1 both are simple sentences that can easily be translated into a formula starting with the Noun binder. The only remark we make is that in the second formula we see 'is a rational number' being translated as if the formulation was 'is an element of the rational numbers', where 'an element of the' is syntactic sugar (see Example 3.8). This conversion makes it easier to discover a correct formula for it.

The last two sentences in Table 5.1 show a combination which is often used in WTT, namely the Noun – \( \exists \) combination. The use of this combination to translate those two sentences into Noun formulas may not seem perfectly clear immediately. When dealing
with this kind of sentences more often, we will develop a feeling that tells us that the fourth natural language expression can also be read as ‘A real number for which there exists a natural number such that the real number is equal to the square root of the natural number.’ Moreover, the expression behind the Noun binder must be a statement, which justifies our choice for the existential quantification. A straightforward translation such as ‘\( \text{Noun}_{x \in \mathbb{R}} \text{Noun}_{n \in \mathbb{N}} (x = \sqrt{n}) \)’ is simply not allowed in WTT, because in this translation, ‘\( \text{Noun}_{n \in \mathbb{N}} (x = \sqrt{n}) \)’ is not a statement.

In the last sentence, about the sequence \((a_n)_{n \in \mathbb{N}}\), we also have to take the comment from section 4.4.4 into account, saying that we define a sequence to be a function, mapping natural numbers onto real numbers: \( \mathbb{N} \to \mathbb{R} \).

5.3 Specific nouns

As opposed to generic nouns, specific nouns, as introduced in section 4.4.2, are used when we want to indicate a specific object that belongs to a certain class. We have already learnt to use the \( t \) binder for this purpose. Specific nouns denote unique objects, such as ‘the steering wheel of a car’, ‘the integer between -4,5 and -5,25’ and ‘the height of a cone’. So, when looking at a natural language construction, the definite article ‘the’ triggers us to think of a specific noun.

A natural language expression containing such a description of a specific noun can be transformed into a WTT formula containing the \( t \) binder according to the following general format:

\[ t_{v}(S) \]

Again, similar as for the Noun binder, the ‘\( D \)’ stands for ‘Domain’ and the ‘\( S \)’ stands for ‘Statement’. So this means that a noun, regardless of whether it is generic or specific, will always be followed by a statement.

Example 5.2 will show some examples of natural language expressions along with a correct translation into WTT. The translations all contain the \( t \) binder since we are dealing with specific nouns here.

**Example 5.2** WTT formulas with the \( t \) binder that are equivalent to the given natural language expressions.

<table>
<thead>
<tr>
<th>Natural language</th>
<th>Corresponding formula with ( t ) binder</th>
</tr>
</thead>
<tbody>
<tr>
<td>“The natural number that is the solution to the equation ( x^2 + 6 = 5x ) is greater than 1.”</td>
<td>( t_{x \in \mathbb{N}} (x^2 + 6 = 5x) )</td>
</tr>
</tbody>
</table>
"The positive real number of which the sum of its square and its third power is equal to 11."

\[ \forall x \in \mathbb{R} \ (x^2 + x^3 = 11) \]

"The real number for which it holds that for all elements of the sequence \((a_n)_{n \in \mathbb{N}}\) holds that its successor is precisely the amount of that real number greater than itself."

\[ \forall c \in \mathbb{R} \ \forall n \in \mathbb{N} \ (a_{n+1} = a_n + c) \]

"The natural number smaller than 5, for which there exists a positive integer, other than 1 and that natural number itself, which is a divisor of that natural number."

\[ \exists d \in \mathbb{Z}^{\setminus \{1, 5\}} \ (d \text{ is a divisor of } n) \]

<table>
<thead>
<tr>
<th>Table 5.2 Natural language expressions starting with 'the' and corresponding formulas using the (t) binder</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;The positive real number of which the sum of its square and its third power is equal to 11.&quot;</td>
</tr>
<tr>
<td>&quot;The real number for which it holds that for all elements of the sequence ((a_n)_{n \in \mathbb{N}}) holds that its successor is precisely the amount of that real number greater than itself.&quot;</td>
</tr>
<tr>
<td>&quot;The natural number smaller than 5, for which there exists a positive integer, other than 1 and that natural number itself, which is a divisor of that natural number.&quot;</td>
</tr>
</tbody>
</table>

The conversion into WTT formulas containing the \(t\) binder is rather straightforward. Nevertheless we want to make two additional remarks. The first one is that we frequently encounter universal and existential quantifications behind a \(t\) binder, as can be seen in the last two WTT formulas of Example 5.2.

The second remark requires us to look carefully at the first WTT formula we gave. When examining this formula in detail we see that, oddly enough, there are two natural numbers which are solutions to the inner equation and that both of them are greater than 1. Though logically incorrect, we do consider such formulas to be correct in WTT. This is due to the fact that WTT is about the formalization and structuring of mathematical texts, and not about the logical or mathematical correctness of them, as we already stated in section 1.3.

### 5.4 Adjectives

Adjectives are quite easy to discover in natural language, because they provide some additional information about the noun they accompany. Examples of adjectives are 'odd' (for the noun 'function'), 'acute' (for the nouns 'angle' or 'triangle') and 'oblique' (for the nouns 'cylinder', 'cone', 'pyramid' or 'prism' in geometry).

In section 4.4.3 we introduced the use of the Adj binder for handling adjectives in WTT. We also saw the two different appearances of natural language expressions and WTT formulas concerning adjectives. Therefore, there are also two general formats in which a formula containing an Adj binder can occur:

\[ '\text{Adj}_D (S)' \text{ and 'adjname := Adj}_D (S)' \]
The ‘D’ and ‘S’ are serving the same purpose as for generic and specific nouns. This means that also the Adj binder is always followed by a statement that says something about D. Furthermore, we used ‘adjname’ to indicate the name that can be given to an adjective (as can be seen in the last two formulas of Example 4.13).

Whenever we encounter natural language expressions starting with ‘The adjective saying …’ or containing the word ‘called’, as in ‘… is called …’, the first thing to think about should be that this expression deals with an adjective. In Example 5.3 a few expressions in natural language are given, which are converted into the correct formula using the Adj binder.

**Example 5.3** The use of the Adj binder when translating natural language constructions that specify adjectives.

<table>
<thead>
<tr>
<th>Natural language</th>
<th>Corresponding formula with Adj binder</th>
</tr>
</thead>
<tbody>
<tr>
<td>“The adjective saying of a function from A to B, that for all elements of A hold, that the result of that function, applied to such an element, is greater than 0.”</td>
<td>( \text{Adj}_{f:A\to B} \forall a \in A \left( f(a) &gt; 0 \right) )</td>
</tr>
<tr>
<td>“The adjective saying of a relation on the real numbers, that for all triples of real numbers holds that if the first and second element are related, and the second and third element are related, then the first and third element are related as well.”</td>
<td>( \text{Adj}_{R:a \text{ relation on } \mathbb{R}} \forall x,y,z \in \mathbb{R} \left( xRy \land yRz \Rightarrow xRz \right) )</td>
</tr>
<tr>
<td>“A triangle is called obtuse if one of its angles is bigger than 90°.”</td>
<td>( \text{obtuse} := \text{Adj}_{\triangle: \text{a triangle}} \exists \alpha, \text{an angle of } \triangle \left( \alpha &gt; 90° \right) )</td>
</tr>
<tr>
<td>“A sequence of real numbers is called geometric if the quotient of each two successive elements is constant.”</td>
<td>( \text{geometric} := \text{Adj}<em>{a: \mathbb{R}} \exists c \in \mathbb{R} \forall n \in \mathbb{N} \left( a_n / a</em>{n+1} = c \right) )</td>
</tr>
</tbody>
</table>

Table 5.3 Converting the two distinct occurrences of adjectives in natural language into equivalent WTT formulas

In the first two formulas it is obvious that the domain accompanying the Adj binder indicates the noun of which the actual adjective describes a property. In the last two formulas, the property of the noun is explicitly put in front of the formula as an assignment, which makes the formula much more readable and pronounceable.

**5.5 Functions and predicates**

The notions of functions and predicates are totally different in both natural language and WTT. Yet, the latter uses the same symbol to express both notions, viz. the \( \lambda \) binder.
When looking at a natural language expression it will be immediately clear whether we are dealing with a function or predicate, because the words ‘function’ and ‘predicate’ are usually mentioned in it. We will first discuss function descriptions that can be converted into WTT formulas, using the $\lambda$ binder in its role as a function binder. Subsequently we will discuss predicate descriptions using the $\lambda$ binder in its other role, i.e., its role as a predicate binder.

### Function descriptions

Whenever a function is described in natural language, the word ‘function’ will always appear in this description. We will have a look at some functions that are specified with natural language constructions in Example 5.4 first, before discussing them in detail.

**Example 5.4** The $\lambda$ binder is used to translate functions, which are specified in natural language, into equivalent WTT formulas.

<table>
<thead>
<tr>
<th>Natural language</th>
<th>Corresponding formula with $\lambda$ binder</th>
</tr>
</thead>
<tbody>
<tr>
<td>“The function sending a positive integer to its fourth power plus 1.”</td>
<td>$\lambda_{x \in \mathbb{Z}} (i^4 + 1)$</td>
</tr>
<tr>
<td>“The function sending a natural number to the set of all prime numbers that are smaller than that natural number.”</td>
<td>$\lambda_{n \in \mathbb{N} \setminus {m \in \mathbb{N} : \text{a prime number} \land m &lt; n }}$</td>
</tr>
<tr>
<td>“By definition, $f$ is the function, sending a real number greater than 2 to the square root of the absolute value of the sum of that real number and its square.”</td>
<td>$f := \lambda_{x \in \mathbb{R} \setminus {x &gt; 2}} (\sqrt{</td>
</tr>
<tr>
<td>“By definition, $g$ is the function, sending a natural number to the set of all squares of natural numbers that are smaller than that first natural number.”</td>
<td>$g := \lambda_{n \in \mathbb{N}} ({ k^2</td>
</tr>
</tbody>
</table>

**Table 5.4 Applications of the $\lambda$ binder as function binder**

We see that the phrase ‘the function sending’ is characteristic in all these natural language constructions. So, whenever this phrase occurs, we are immediately convinced that we are dealing with a function description and hence need the $\lambda$ binder to convert this description into a WTT formula.

From the examples in Table 5.4 we can deduce that behind the $\lambda$ binder we can encounter either a formula (first and third entry in the table) or a set (second and last entry).
entry). Therefore, the two general formats for a formula containing the $\lambda$ binder in its role as a function binder are the following:

\[ ' \lambda_D (F) ' \text{ and } ' \lambda_D (S)' \]

where, ‘$D$’ and ‘$F$’ denote ‘Domain’ and ‘Formula’ and ‘$S$’ denotes ‘Set’. What we call a formula throughout this entire course is more formally called a term. This clarifies that a function that is specified according to the ‘$\lambda_D (F)$’-format is also called a term-valued function. Similarly, a function written down in the ‘$\lambda_D (S)$’-format is called a set-valued function.

One of the shortcomings of natural language that comes to light here is that in some cases it is multi-interpretable. This proves to be the case in the description of the function $g$ in Table 5.4. In this description it is not clear if the squares of the natural numbers must be smaller than the first natural number, or if the natural numbers themselves must be smaller than the first natural number. Therefore, its equivalent WTT formula could just as well have been ‘$g := \lambda_{n \in \mathbb{N}} ( \{ k^2 | k \in \mathbb{N} \land k < n \} )$’.

As mentioned in section 4.4.2, parentheses in a natural language expression could be used to avoid such ambiguities. This yields an expression that could look like this: “By definition, $g$ is the function, sending a natural number to (the set of all squares of natural numbers) that are smaller than the first natural number.”

The final remark that we, perhaps superfluously, want to make about Example 5.4 is that names can be assigned to functions in the same way as we previously used for adjectives. In the last two WTT formulas of Table 5.4 we applied this technique using the names $f$ and $g$, respectively. By doing this, it becomes a lot easier to refer to a specific function, since we can suffice by mentioning its name instead of a long description.

**Predicate descriptions**

Whenever the word ‘predicate’ is occurring in a natural language construction, it is obvious that this construction describes a predicate. In some cases however, the word ‘predicate’ as such is not used explicitly, but the structure of the natural language construction makes it indisputably clear that we are dealing with a predicate description. A few examples of both ways to describe a predicate are shown in Example 5.5. As we can see, in the third example the word ‘predicate’ is not used in the natural language expression, though its structure tells us that it describes a predicate indeed.
Example 5.5 The second purpose for which the $\lambda$ binder is used is to translate predicate descriptions in natural language into equivalent WTT formulas.

<table>
<thead>
<tr>
<th>Natural language</th>
<th>Corresponding formula with $\lambda$ binder</th>
</tr>
</thead>
<tbody>
<tr>
<td>“The predicate that holds for a natural number if it is even.”</td>
<td>$\lambda_{n \in \mathbb{N}} (n \text{ is even})$</td>
</tr>
<tr>
<td>“The predicate that holds for a set if it contains more than one element.”</td>
<td>$\lambda_{S, a \in S} \exists_{b \in S} (a \neq b)$</td>
</tr>
<tr>
<td>“$P$ holds for a natural number if it is greater than 0.”</td>
<td>$P := \lambda_{n \in \mathbb{N}} (n &gt; 0)$</td>
</tr>
<tr>
<td>“$Q$ is the predicate that holds for a real number if its sine is equal to its cosine.”</td>
<td>$Q := \lambda_{x \in \mathbb{R}} (\sin(x) = \cos(x))$</td>
</tr>
<tr>
<td>“$P$ is the predicate that holds for a natural number if it is equal to the difference of two third powers of prime numbers greater than 2.”</td>
<td>$P := \lambda_{n \in \mathbb{N}} \exists_{k: \text{a prime number} \mid k &gt; 2} (n = k^3 - l^3)$</td>
</tr>
</tbody>
</table>

Table 5.5 Applications of the $\lambda$ binder as predicate binder

The general format for a formula containing the $\lambda$ binder in its role as a predicate binder is as follows:

\[ '\lambda_D (S)' \]

where ‘$D$’ and ‘$S$’ again stand for ‘Domain’ and ‘Statement’. So a predicate tells us whether a certain statement holds for an element from the specified domain.

We can encounter another natural language construction describing a predicate though. In such an alternative description of a predicate the word ‘property’ can be found. An example would be the following: “The property ‘...’ for ...”, as in “The property ‘being even’ for a natural number.”. When we convert this particular example into a WTT formula using the $\lambda$ binder, we get $\lambda_{n \in \mathbb{N}} (n \text{ is even})$, which is the exact same formula as we already saw in Table 5.5. It makes no difference for the resulting WTT formula whether the formulation in natural language makes use of the predicate or property point of view, or whether it explicitly contains the word ‘predicate’ or not.

We conclude this section by saying something about another natural language construction that frequently occurs, namely the natural language construction “Let ... hold for ... if ...”. When we consider “Let $P$ hold for a natural number if it is even.” for example, the translation into a WTT formula, again, becomes $P := \lambda_{n \in \mathbb{N}} (n \text{ is even})$. 

- 72 -
5.6 Abstractions

Abstractions can be recognized in natural language by the word ‘some’. It indicates that it doesn’t really matter whichever element we choose from a particular class or domain; we can just pick a general one.

We will show some examples of natural language formulations using abstractions now. After these examples we will discuss the different forms of abstraction in more detail.

Example 5.6 The word ‘some’ points out an abstraction that can be converted into a WTT formula with the use of the Abst binder.

<table>
<thead>
<tr>
<th>Natural language</th>
<th>Corresponding formula with Abst binder</th>
</tr>
</thead>
<tbody>
<tr>
<td>“The square of some natural number.”</td>
<td>$\text{Abst}_{n\in\mathbb{N}} (n^2)$</td>
</tr>
<tr>
<td>“The square root of the factorial of some natural number.”</td>
<td>$\text{Abst}_{m\in\mathbb{N}} (\sqrt{m!})$</td>
</tr>
<tr>
<td>“The set of real numbers greater than some natural number.”</td>
<td>$\text{Abst}_{n\in\mathbb{N}} ({ x \in \mathbb{R}</td>
</tr>
<tr>
<td>“A natural number which, increased by 1, is equal to 2 to the power some positive integer.”</td>
<td>$\text{Abst}<em>{k\in\mathbb{Z}} k \geq 0 \text{Noun}</em>{n\in\mathbb{N}}(l+1=2^k)$</td>
</tr>
</tbody>
</table>

Table 5.6 The Abst binder used to mould the described abstraction into an equivalent WTT formula

From these examples we see that we abstracted from three different things using the Abst binder. We abstracted from a formula (term), a set and a noun. Therefore, a WTT formula containing the Abst binder can have the following general formats:

\['\text{Abst}_D (F)', \ '\text{Abst}_D (S)' \ and \ '\text{Abst}_D (\text{Noun}(F))'.\]

The ‘Domain’ is denoted by ‘$D$’, while ‘$S$’ and ‘$F$’ denote ‘Set’ and ‘Formula’ again. When we recall Figure 3.3, we can observe that the third general format as displayed above actually is included in the first one, because we defined ‘\(\text{Noun}(F)\)’ to be a formula itself, starting with a binder. We chose to mention abstraction from a noun explicitly here because the \(\text{Abst} - \text{Noun}\) combination is frequently used in WTT.

The WTT formula ‘\(\text{Abst}_{f: \mathbb{N}\to\mathbb{N}} \lambda_{n\in\mathbb{N}} (f(n) + f(n))\)’ is in a correct WTT format, namely the ‘\(\text{Abst}_D (F)\)’ – format, as well for the same reason, because ‘\(\lambda_{n\in\mathbb{N}} (f(n) + f(n))\)’ is a formula, starting with the \(\lambda\) binder. To be complete, we will give a possible translation of this WTT formula into natural language, in which we use parentheses to avoid any ambiguity in the description: “The function sending a natural number to the double of (a function applied to this natural number), for some function on the natural numbers.”.
5.7 Typing sentences

In chapter 2 we briefly mentioned typing sentences. Typing sentences are a subkind of the statements. Their purpose is to describe of which type an object is or to which class an object belongs. For these two purposes we have two means at our disposal, i.e., a specific typing sentence and a generic typing sentence.

A specific typing sentence can be recognized by the phrase ‘... is an element of ...’. We will denote this particular phrase with the symbol ‘∈’, which we already used in the domains of binders silently. For instance, when we want to express that an object x is an element of a class (or set) V, we denote this by ‘x ∈ V’. Here, the class V is the specific type of x. Other examples would be ‘5 ∈ ℤ’ and ‘k ∈ K’.

When we come upon one of the phrases ‘... is a(n) ...’ or ‘... is of type ...’, then we can be sure that we are dealing with a generic typing sentence here. Such a generic typing sentence will be written down with the symbol ‘:’ (colon). For example, when we want to express that an object x is of type t, we write ‘x : t’. Some more examples would be ‘CDEFG : a pentagon’, ‘11 : a prime number’ and ‘α : an angle of ΔABC’. Notice that this colon-notation for a generic typing sentence matches with the way in which we have been denoting functions in this course already: ‘f : ℜ → ℜ’.

Both notations have more or less the same meaning, which is obvious when looking at the sentences ‘1/2 : a rational number’ and ‘3/4 ∈ ℚ’. In general, these notations are related in a way that can be captured perfectly by using the ↑ and ↓ constants. If an object x belongs to a class V, then that object x is of type V ↓ and if an object y is of type u, then that object y is an element of u ↑. These formulations can be expressed using typing sentences, as we can see below:

- If ‘x ∈ V’ then ‘x : a (V ↓)’;
- If ‘x : a t’ then ‘x ∈ t ↑’.

Sometimes another symbol is used in a specific typing sentence instead of ‘∈’. This is mainly the case when we are dealing with sets and subsets. The symbol that replaces the ‘∈’ symbol is the subset symbol ‘⊆’. An example of a specific typing sentence using ‘⊆’ is ‘T ⊆ S’, meaning that T is a subset of (or: is included in) S.

To conclude we will now give some examples of typing sentences with their translation into an equivalent natural language expression.

Example 5.7 Different generic and specific typing sentences, accompanied by their translation into natural language. Of course, these examples can be read vice versa also, i.e., from typing sentence to natural language expression.
<table>
<thead>
<tr>
<th>Natural language equivalent</th>
<th>Typing sentence</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;The square of 8 is a third power.&quot;</td>
<td>(8^2: \text{a third power})</td>
</tr>
<tr>
<td>&quot;The pair ((x,y)) is an element of (\mathbb{R}^2).&quot;</td>
<td>((x,y)\in \mathbb{R}^2)</td>
</tr>
<tr>
<td>&quot;343 is the third power of some prime number.&quot;</td>
<td>343: (\text{Abst}_{p, \text{a prime number}} \ (p^3))</td>
</tr>
<tr>
<td>&quot;(t) is the square root of the third power of some natural number.&quot;</td>
<td>(t: \text{Abst}_{n \in \mathbb{N}} (\sqrt{n^3}))</td>
</tr>
<tr>
<td>&quot;The function sending a real number to its fifth power is an odd function.&quot;</td>
<td>(\lambda_{x \in \mathbb{R}}(x^5): \text{an odd function})</td>
</tr>
</tbody>
</table>

Table 5.7 Natural language expressions with their conversion into typing sentences

### 5.8 Equivalences between WTT formulas

Up till now we have seen numerous examples of all existing WTT binders in this chapter. We have also developed an intuition of how to discover and recognize these WTT binders in natural language expressions. In the examples we have seen so far, we could have asked ourselves whether a conversion of natural language expressions into WTT formulas is always unique, i.e., whether there is exactly one correct formula for each natural language expression. Maybe other formulas, using different binders, can be correct as well? As hinted in the introduction of this chapter, this indeed is a frequently occurring phenomenon.

The most important remark we want to make in this context is about the close relationship between the Noun binder and the Abst binder. We recall that the difference between those two is very subtle: a Noun binder is always followed by a statement, whereas an Abst binder is followed by a term, set or noun.

A WTT formula containing an Abst binder can always be rewritten without it. To show this, we will provide an example now in which we rewrite the Abst formulas from Example 5.6 to a form in which the Abst binder does not occur. All these examples have the same structure, viz. they start with a Noun binder and an inside \(\exists\) replaces the outside Abst.

**Example 5.8** The Abst in WTT formulas can be safely replaced by the equivalent Noun – \(\exists\) combination.
<table>
<thead>
<tr>
<th>Formula containing Abst binder</th>
<th>Equivalent formula without Abst binder</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Abst_{\in\mathbb{N}}(n^2))</td>
<td>(Noun_{m\in\mathbb{N}}(m=n^2))</td>
</tr>
<tr>
<td>(Abst_{\in\mathbb{N}}(\sqrt{m!}))</td>
<td>(Noun_{k\in\mathbb{N}}(k=\sqrt{n!}))</td>
</tr>
<tr>
<td>(Abst_{\in\mathbb{N}}{x\in\mathbb{R} \mid x&gt;n})</td>
<td>(Noun_{\in\mathbb{N}}{x\in\mathbb{R} \mid x&gt;n})</td>
</tr>
<tr>
<td>(Abst_{k\in\mathbb{Z}\mid k&gt;0}Noun_{i\in\mathbb{N}}(l+1=2^k))</td>
<td>(Noun_{i\in\mathbb{N}}\exists_{k\in\mathbb{Z}\mid k&gt;0}(l+1=2^k))</td>
</tr>
</tbody>
</table>

Table 5.8 Equivalent alternatives for the Abst binder in WTT

Whenever we encounter a natural language expression in which the indefinite article ‘a’ is present, this means that there is a choice of how to transform this expression into a correct WTT-formula. For instance, when we look at the (typing) sentence “\(x\) is a natural number between 10 and 15”, we have lots of possibilities to come up with a correct formula. We could opt for either one of the following four formulas:

- ‘\(x: Noun_{\in\mathbb{N}}(10<n<15)\)’
- ‘\(x:(\{n\in\mathbb{N} \mid 10<n<15\})\)’
- ‘\(x\in\{n\in\mathbb{N} \mid 10<n<15\}\)’
- ‘\(x\in\mathbb{N} \land 10<x<15\)’

The last mentioned one is the easiest of the four. It is correct indeed, though it does not contain a WTT binder or WTT formula symbol. It is in fact a WTT formula as well as a general mathematical formula. This conclusion also holds for the third formula, whereas the first and second alternatives contain a WTT binder and formula symbol, respectively. Since all these alternatives are correct, it depends on one’s own preference which one is chosen to translate the given natural language expression.

Another good example where multiple formulas are correct is the sentence “The square root of the difference of two squares of natural numbers.” Here we can choose from the following, both correct, WTT formulas:

- ‘\(Noun_{x\in\mathbb{R}}\exists_{y,z\in\mathbb{N}}(x=\sqrt{y^2-z^2})\)’
- ‘\(Abst_{y,z\in\mathbb{N}}(\sqrt{y^2-z^2})\)’

When opting for the first formula, we made the choice to read the natural language expression as if it had been formulated as: “A real number for which there are two natural numbers such that this real number is equal to the square root of the difference of the squares of those two natural numbers.” This is allowed, because this rather elaborate description expresses the same as the given natural language expression.
Translating the second of the two alternatives into natural language would result in: "The square root of the difference of the squares of some (pair of) natural numbers." This description is also equal to the given natural language expression.

Finally, we want to point out that all the rewrite rules for universal and existential quantifications can be applied in WTT as well. Stating that it is not the case that some predicate holds for all elements of a particular domain is logically equivalent to stating that there exists (at least) one element within that domain for which the predicate does not hold. Reversely, when we claim that it is not the case that there exists an element from a certain domain satisfying a particular predicate, we are in fact saying that this predicate does not hold for all elements of that domain. These two rules are usually expressed as follows, where \( x \) denotes an element of a certain domain and \( P \) denotes a predicate.

\[
\neg \forall x \ P(x) \iff \exists x \ \neg P(x)
\]
\[
\neg \exists x \ P(x) \iff \forall x \ \neg P(x)
\]

For an extensive explanation and overview of all the rules and equivalences for logical connectives and quantifiers, we recommend [10], in which the basics about logical calculations and equivalences are discussed in great detail.

5.9 Summary

In this chapter, we tried to approach the notions as explained in chapter 4 from the other way around. We used natural language expressions as our starting point and tried to construct equivalent WTT formulas for them.

We mentioned the most frequently occurring natural language expressions describing generic and specific nouns, adjectives, formulas, predicates and abstractions, respectively. By providing general formula formats for all different WTT binders, we gave the syntax for these binders in a rather informal way. We have developed a sense for recognizing certain linguistic patterns in natural language by giving a lot of examples of translations of natural language expressions into WTT formulas.

The notion of typing sentences has been introduced more thoroughly in this chapter, for they were only mentioned casually in chapter 2. We made an explicit distinction between generic and specific typing sentences; the former using ':' and the latter using '∈'. The relation between these two typing sentences can be described very well using the two WTT constants \( \uparrow \) and \( \downarrow \).

Finally, we focused on the variety of correct WTT formulas that can exist for one single natural language construction. There often are alternatives which are mutually interchangeable. Whichever WTT formula format is chosen depends on one's personal preferences.
6 Analyzing mathematical texts

6.1 Introduction

In this chapter we focus on the linguistic level of texts. We consider sentences as our elementary text units from which complete texts and text fragments are built. In chapters 4 and 5 we introduced WTT as a formalism to rewrite mathematical texts written in natural language in a more structured, orderly manner. Here, in chapter 6, we extend this formalism by presenting two different representation styles called flag notation and book notation. These representation styles are WTT-instruments that can display a mathematical text in such a way that we immediately see the omissions, shortcomings and ambiguities of this text as it was formulated in natural language.

The first part of this chapter is mainly about the notion of context. Contexts are very important in mathematics because they influence the way mathematical statements and definitions must be interpreted. A lot of statements and definitions are context dependent, meaning that without a properly generated context with the appropriate elements in it, they cannot be read in the way they should.

When we are familiar with the notion of context, we introduce context markings to display a WTT text. These markings are drawn as flags and therefore we say that a WTT text written in this style is in flag notation or flag representation. The flags are put up to make the types of context sentences stand out more strikingly.

When we have established a WTT text in flag notation, we can straightforwardly derive its book representation. This is the other of the two representation styles we discuss. Both styles offer more insight in the precise way a mathematical text is built up. The book representation is more compact than the flag representation, but it is not possible to point out nested contexts as accurate as this is possible in flag representation.

We discover that most mathematical texts are too implicit or even incomplete, causing confusion or ambiguity for the reader. This especially is the case with definite and indefinite articles that are used in natural language. We rigorously criticize examples of mathematical texts by formalizing them and explicitly adding the things they were implicitly supposed to say.

6.2 Relations between sentences

In section 2.6, we already discussed sentences briefly. In this section we will go into more detail about sentences as elementary text units. A text or a text fragment is a sequence of at least two sentences, which are interrelated to each other. Such relations can for instance be causal, sequential or temporal, to mention only a few.
In mathematical texts we frequently encounter so-called mathematical text markers. These markers, which have been listed in section 2.7, help the reader to interpret the subsequent text and to recognize the kind of relation the subsequent sentences have.

But when leaving mathematical text markers out of consideration, how else can we recognize relations between sentences in a written text? Usually we can derive these relations by external 'instruments', like the arrangement of the sentences and the punctuation that is used. This course for example is arranged into six chapters, which in turn are divided into various sections and subsections. Such a structure tells us which topics are closely related to each other and which topics are less related.

6.3 Context

Words that make up sentences in a text are only useful to us when we know their meaning. We cannot understand a text before we know the meaning of each word occurring in it. In general, the meaning of words is traditionally determined and therefore established somewhere outside the text in which they occur. Dictionaries are available for a lot of modern languages, describing the traditional meaning of words of a certain language.

But even knowing the meaning of all words occurring in a text is not sufficient to understand it. We also have to have linguistic knowledge on a higher level, i.e., knowledge about the rules, habits and meanings of phrases, sentences and groups of related sentences, to fully understand a text.

To this particular knowledge a subfield in linguistics is devoted. This subfield is about the study of language meaning and is called semantics. Every text is related to its 'linguistic environment' because the meaning of the occurring words and phrases is established outside this text. This comprehensive knowledge that contributes to the overall understanding of a text in a specific language is the global context of this text. Concluding, we can say that a text is comprehensible only when one is familiar with the global context of the language that this particular text is written in.

For a fragment of text that has been picked out of a larger, complete text, we have to have additional information at our disposal to read and interpret it correctly. However, we do not need an entire global context here, but a local context is sufficient. A local context is the minimal amount of extra information required to be able to read a text fragment without considerable problems. In the following example we will show a text fragment consisting of two sentences. We will discuss what the local context of this fragment should contain.
Example 6.1 The following two natural language sentences form a text fragment:

"The engine broke down. He didn’t know what they could do next."

When reading this text fragment, there are several things not entirely clear. We need more specific, additional information, which should be present in the local context, such as:

- To who is the personal pronoun ‘he’ referring?
- To who is the personal pronoun ‘they’ referring?
- Which ‘engine’? Of a car? Of a plane? An engine in a factory hall?
- ‘next’: What has been done previously?

From Example 6.1 we can conclude how important a local context is to be able to read and interpret a text fragment in the way the author had intended to.

In mathematical language, context is of similar importance as it is in natural language. Mathematical standard words, like \( \pi, \infty \) and \( \mathbb{N} \), belong to the global context of mathematical texts. This global context can be seen as a kind of ‘dictionary’ but then for mathematics. Thus the mathematical global context is universal for all mathematical texts.

A mathematical text fragment has its own mathematical local context, just as a natural language text fragment has its own natural language local context. In Example 6.2 we will show a mathematical text fragment and point out what its local context should contain.

Example 6.2 The following two mathematical language sentences form a text fragment:

"a and b are both Mersenne-numbers. f is a continuous function on the interval \([a,c]\)."

The mathematical local context for this particular text fragment should contain the following:

- How is a ‘Mersenne-number’ defined?
- What function does \( f \) denote?
- What is a ‘continuous function’?
- What is \( c \)?

The elements we mentioned in this particular example as being in the mathematical local context can be divided into two categories. To the first category belong names referring
to objects, such as $f$ and $c$. To the second category we count definitions, such as ‘Mersenne-number’ and ‘continuous function’.

Definitions are a special kind of sentences in mathematical language. They belong to the context sentences. In the next section we will explain the different types of context sentences existing in WTT.

### 6.4 Context sentences

A special type of sentences in WTT is the type of context sentences. Context sentences are used, as their name already states, in a context dependent way. They extend the mathematical local context with the knowledge that is necessary to read and understand a mathematical text fragment properly. Such a context extension is temporarily. We will come back to durations of contexts in section 6.5.

We distinguish three different types of context sentences in WTT, viz. *definitions*, *generations* (also called *declarations*) and *assumptions*. We will discuss them in the same order in this section.

A definition establishes the meaning of a word, phrase or sentence that will be used later on. In a mathematical text, definitions are often marked by the mathematical text marker ‘Definition’. In WTT we use two different symbols for definitions:

- ‘:=’ We encounter this symbol in programming languages as well, where it is used to execute an assignment. In WTT this symbol is used somewhat similarly, namely for definitions of nouns (both generic and specific) and adjectives.

- ‘$\vDash$’ This symbol we use for the definition of a whole sentence, which is very well possible in WTT.

In Example 6.3 we show examples of definitions of a specific noun, a generic noun, an adjective and a sentence, respectively.

**Example 6.3** Definitions in all possible occurrences in mathematical texts or text fragments.

<table>
<thead>
<tr>
<th>Definitions in natural language</th>
<th>Translation into a WTT definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>“By definition, $x$ is one fourth of the square root of 3.”</td>
<td>$x := \frac{1}{4} \sqrt{3}$</td>
</tr>
<tr>
<td>“A Fermat number is defined as an integer which is the sum of two squares of integers.”</td>
<td>Fermat number $= \text{Noun}<em>{x \in \mathbb{Z}} \exists</em>{y \in \mathbb{Z}} (x = y^2 + z^2)$</td>
</tr>
</tbody>
</table>
"A function on the natural numbers is unbounded if for every natural number there exists a natural number of which the value of that function, applied to that last natural number, is greater than the first natural number."

\[
\text{unbounded} := \text{Adj}_{f: \mathbb{N} \rightarrow \mathbb{N}} \forall y \in \mathbb{N} \exists x \in \mathbb{N} (f(x) > y)
\]

"A set \( V \) has a minimum if it contains an element which is smaller than all other elements of that set."

\[
V \text{ has a minimum} \equiv \exists x \in V \forall y \in V \setminus \{x\} (x < y)
\]

<table>
<thead>
<tr>
<th>Table 6.1 Definitions and their translations into natural language</th>
</tr>
</thead>
<tbody>
<tr>
<td>The first definition in Table 6.1 is straightforwardly defining the specific object name ‘( x )’. It is defining a specific noun, because the right hand side of the definition specifies a fixed, unique number (( \approx 0.433 )). If this definition was followed by a text in which this ‘( a )’ occurred, then we could read every occurrence of ‘( a )’ as ‘( \frac{1}{2} \sqrt{3} )’.</td>
</tr>
<tr>
<td>The second and third formula in Table 6.1 are not entirely new for us. We already saw that we can allocate a name to an adjective in section 4.4.3. The exact same strategy can be applied to descriptions containing the Noun binder as well. It provides the same conveniences as described for adjectives. Finally, the last formula in the table shows how a sentence can be defined in WTT.</td>
</tr>
<tr>
<td>In a generation, a name for a generic or specific object is introduced, together with its type, so a generation in fact is a typing sentence (see section 5.7). There is no difference in the way in which generic or specific objects are introduced. Generations like ‘( x \in \mathbb{R} )’ and ‘( n : \text{a natural number} )’ are characteristic in WTT.</td>
</tr>
<tr>
<td>An assumption is a proposition that is treated as if it were known to be true. Actually, we don’t know whether assumptions are really true, and sometimes they will turn out to be false, but when adding assumptions in a certain local context, we create an imaginary situation for ourselves in which we assume them to be true. Once such a situation has been created, the assumptions made can be used in the subsequent text.</td>
</tr>
</tbody>
</table>

6.5 Context markings

In the previous section we learnt that in mathematical language it is of great importance to conscientiously keep track of the local context. To ease this laborious chore, we will introduce a formalism, which is called ‘flag notation’, as a means to demarcate contexts in a mathematical text or text fragment. By using another kind of context demarcation for each type of context sentence, we want to emphasize once more that a local context is of essential importance to interpret a mathematical text correctly. Furthermore, keeping track of the local context in a text or text fragment becomes a lot simpler when using the
flag notation, because the definitions, generations and assumptions in the text are really conspicuous now.

The basis for using markings to emphasize certain important parts of a proof, like assumptions, was laid in [8], where Fitch makes use of vertical lines drawn on the left hand side of the proof lines. He also introduced a short, horizontal dash to indicate that the lines above it were assumptions (Fitch actually uses the word hypotheses here, having the same meaning). A proof or deduction in which these markings are present is said to be a 'Fitch-style' proof or a 'Fitch-style' natural deduction.

The markings used in [8] have been further refined by Nederpelt in [11], in which the predecessor of the flag notation, as we will treat it here, is introduced. He brought up the fact that in proofs the term of validity of an assumption is left to the reader's insight. He introduced a way to clearly demarcate assumptions in proofs by framing them. Moreover, he found a way to indicate the term of validity of an assumption in a proof, which we will see later on in this section.

Finally, Nederpelt discovered that not only proof texts could benefit from these frames, but that they could be used on a much larger scale, viz., in all kinds of mathematical texts. He introduced frames in different shapes for assumptions, definitions and generations in [12]. These frames are called flags because they closely resemble a flag in the way it is commonly visualized. We will use the flag notation as defined in [12] for the remainder of this course, though we see other, often closely related ways of how flags are used in mathematical proofs and texts (see [3], [8], [11]).

First, in Figure 6.1 we will depict the flags that we will be using before we specify how they are used in WTT. From top to bottom we see a demarcation of two definitions, a generation and an assumption, respectively.

\[
x := \frac{1}{2} \sqrt{3}
\]

\[
V \text{ has a minimum} :\iff \exists x \in V \forall y \in V \setminus \{x\} (x < y)
\]

\[
x \in \mathbb{R}
\]

\[
 n \geq 0
\]

Figure 6.1 Three different flags, which are used as a framework for context sentences

The first two flags in Figure 6.1 are rounded flags, which indicate that we are dealing with definitions (in fact, we already saw these exact same definitions in Example 6.3). A
good mnemonic to remember this is that these flags resemble an elongated version of the capital letter ‘D’, which is the first letter of the word ‘definition’ obviously. The third flag, which is a pointed flag, shows us how the generic name \( x \) is generated as an element of the real numbers. The last flag we see in this figure is a straight flag, pointing out that the formula or text it encloses is an assumption.

The expression enclosed in a flag is called the flag text. A pointed flag introducing a new generic name, like the third flag we saw in Figure 6.1, can be considered to be a binding place for later occurrences of that name. But what is the scope of such a binding place? It doesn’t necessarily have to be the end of the text; generations can end much sooner than at the end of a text. The same reasoning also applies to assumptions: we don’t have to assume something all the way to the end of the text; assumptions are made temporarily and often have a scope that does not cover the whole text.

So the flags we depicted in this section still have a shortcoming: they don’t indicate the scope of the flag text. When picturing our usual image of a flag, we come to the conclusion that the flags we have drawn in this section do not exactly resemble this image. The things that are still missing in our flags are the flagpoles, which we will draw downwards, starting from the left side of the flag. They are extraordinarily appropriate to function as a means to indicate the scope of the flag texts present in our flags. The flagpoles are drawn in front of the subsequent text that belongs to the scope of the flag text.

In Example 6.4, which is directly borrowed from [12], a pointed flag is depicted. The flagpole attached to it displays the duration of the generation of the object name \( x \). The text that is preceded by the flagpole belongs to the scope of this generation.

Example 6.4 Text fragment consisting of a generation and three sentences, which are all included in the scope of this generation.

\[
\begin{align*}
  x &\in \mathbb{R} \\
  \text{If } x &\geq 0, \text{ then } |x| = x, \text{ so } x^2 - 2 \cdot |x| + 1 = (x - 1)^2 \geq 0. \\
  \text{If } x &\leq 0, \text{ then } |x| = -x, \text{ so } x^2 - 2 \cdot |x| + 1 = (x + 1)^2 \geq 0. \\
  \text{So, always } x^2 - 2 \cdot |x| + 1 &\geq 0 \text{ holds.}
\end{align*}
\]

We already mentioned that assumptions often have a shorter scope than the entire text in which they occur. So it needs no further explanation that we will also use flagpoles to indicate the scope of the flag text within a straight flag.

In mathematical texts, quite frequently we need more than one generation or assumption. This means for our flag notation that it is allowed to put up a flag inside the scope of a flag that has been put up earlier. If this is the case, we say that the particular flags are
nested, just as parentheses can occur in nested fashion (see section 3.3.3). When examining Example 6.4 again, especially the two 'if'-sentences, we see that in each sentence a veiled assumption is made about the generated real number $x$. So we could write down the exact same example using nested flags now. This leads to the result as displayed in Example 6.5.

**Example 6.5** Structured text fragment containing nested flags.

$$\begin{align*}
  x &\in \mathbb{R} \\
  x &\geq 0 \\
  |x| &= x, \text{ so } x^2 - 2\cdot |x| + 1 = (x - 1)^2 \geq 0. \\
  x &\leq 0 \\
  |x| &= -x, \text{ so } x^2 - 2\cdot |x| + 1 = (x + 1)^2 \geq 0.
\end{align*}$$

So, always $x^2 - 2\cdot |x| + 1 \geq 0$ holds.

Though probably unnecessary, we point out here again that the flagpole of an inner flag may never outrun the flagpole of an outer flag it is contained in. This is because the nesting structure must be taken into account. Similar as for the nesting of parentheses, flags must be correctly nested. The structure of nested flags can be checked analogously to the way in which the nesting of parenthesis is checked (see section 3.3.3).

Unlike for assumptions and generations, we won't use flagpoles for definitions, because the scope of a definition can be rather large. Using flagpoles would do harm to our concept of flag notation. A definition defines a name for an object, which can be used and invoked until this name gets redefined in another meaning. So it has a much longer lifetime than generations and assumptions, of which the validity is retracted as soon as their flagpole ends.

Stating this, implies that if a defined name, say $x$, is defined somewhere and it doesn't get redefined in the remainder of the text, then this $x$ can be used anywhere in the text, even outside the scope of a generation or assumption in which it has possibly been defined initially. So definitions contribute to the local context in a different manner than generations and assumptions do, since the latter two cannot be used outside the scope of the flag in which they have been defined. We will demonstrate this in the following two examples.
Example 6.6 A definition used outside the context in which it has been defined.

\[ p \text{: a prime number} \]

\[ \text{primesum}(p) := \text{the sum of all prime numbers smaller than } p \]

\[ \text{primesum}(17) - \text{primesum}(11) > 20 \]

Without \( p \) in the context as a prime number, we may still formulate statements using the defined \text{primesum}, as long as we apply it to an argument of the right type. Since 17 and 11 are indeed prime numbers, we can be convinced of the correctness of the statement \('\text{primesum}(17) - \text{primesum}(11) > 20'\). Evaluating it would give \text{true} as a result, because \(41 - 17 > 20\) indeed.

Example 6.7 A definition used inside a context in which it hasn't been defined.

\[
\begin{align*}
\text{f} & := \lambda x \in \mathbb{Z} \left( x^2 \right) \\
\text{n} & \in \mathbb{Z} \\
\text{n} & < 0 \\
\text{f}(n) & > 0
\end{align*}
\]

The definition may be used anywhere in the subsequent text, because the \( f \) doesn't get redefined anywhere. Its 'lifetime' reaches until the end of the text.

Generally, different names are used in generations that occur within the same text. Moreover, introducing different names is strongly advised when these generations are nested, i.e., one generation flag is put up within the scope of a previously put up generation flag. If we are stubborn and disregard this advice, a situation like the one displayed in Example 6.8 can occur.
Example 6.8  Introducing the same generic name twice within nested generations causes problems.

In this example, two x's are introduced. They are in fact different: the first one has type \( \mathbb{R} \) and the second one type \([2,3]\). This suggests we may replace the dots in the example by an expression like \( 'x = 1,53' \), because \( 1,53 \in \mathbb{R} \), but when we do this we have a variable conflict because the inner \( x \) has type \([2,3]\) and \( 1,53 \in [2,3] \).

So in general, we have the following rule of thumb: A variable should not be redefined when we still want to use it in the meaning in which we last defined it. Obviously, we should have chosen another name in the second generation here, for instance y. If we had done this, we could have safely placed either \( 'x = 1,53' \) or \( 'y = 2' \) on the dots in Example 6.8.

To end this section, we will provide a rather difficult text fragment in natural language, which, when formalized using the means discussed so far in this chapter, contains all three types of flags. This gives us an indication of how powerful the means of WTT actually are and how closely we can stick to the actual text when formalizing a mathematical text in WTT. We are perfectly aware of the fact that you probably don't understand every single detail in this formalization. We will only make a few brief remarks about it at the end, because we just wanted to show a larger formalization, and we will leave it at that for now.

Example 6.9  A more difficult example of a mathematical text and its formalization using flag notation.

"Let \( V \) be a plane with distance function \( d \), that sends each pair of points of \( V \) to a non-negative real number, in such a way that:

\[
\begin{align*}
(a) \quad & d(x, y) = 0 \iff x = y; \\
(b) \quad & d(x, y) = d(y, x); \\
(c) \quad & d(x, z) \leq d(x, y) + d(y, z).
\end{align*}
\]
The distance from a point in \( V \) to a straight line in \( V \) is equal to the minimum of the distances of that point to an arbitrary point on that line.

\[
\forall x, y \in V ((a)) \land \forall x, y \in V ((b)) \land \forall x, y, z \in V ((c))
\]

- A pair of points in \( V \) is denoted by an element of the commonly used Cartesian product \( V \times V \).
- We write \( 'l \subseteq V' \) and not \( 'l \in V' \) because we consider a line to be a set as well, viz. a set of points.

### 6.6 The roles of articles

In sections 5.2 and 5.3 we already referred to the importance of the indefinite article 'a' (or 'an') in both natural language and mathematical language. It can adopt a lot of different roles. When we wanted to translate a sentence into WTT, we discovered that 'a' and 'an' often describe a noun or an abstraction. However, there are more roles in which these articles occur, for instance as a variable introduction in a generation or even as an existential quantifier.

For the definite article 'the' the same holds, i.e., it doesn’t always have to indicate that a specific noun is described. Sometimes it may serve as a universal quantification as well, as we will see later on in this chapter.
Which role a specific article plays is not always very clear to us immediately. Natural language text fragments can be formulated in such a way that a lot of things are left implicit to the reader. By analyzing this implicit context the structure and meaning of these text fragments can be increased substantially.

To exemplify the abovementioned roles articles can play, we will discuss them by giving examples of text fragments, mostly describing a definition within a certain context. In those text fragments, we will point out the role of the articles. Furthermore, we will provide a formalization into WTT by means of our flag notation.

**Example 6.10** “The point of intersection of the medians of a triangle is called the centroid of that triangle.”

<table>
<thead>
<tr>
<th>Article</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>the (point of intersection)</td>
<td>( t )</td>
</tr>
<tr>
<td>the (medians)</td>
<td>( \forall )</td>
</tr>
<tr>
<td>a (triangle)</td>
<td>generation</td>
</tr>
<tr>
<td>the (centroid)</td>
<td>definition</td>
</tr>
</tbody>
</table>

\[
t: a \text{ triangle} \\
\text{the centroid of } t := t \cdot a \text{ point } \forall m: a \text{ median of } t (p \text{ lies on } m)
\]

This particular text fragment describes a definition in the context of a triangle, recognizable by the words ‘is called’. This definition contains a hidden parameter, namely the generic name \( t \) that we gave to the triangle. The second ‘the’ is translated with a universal quantifier, because all three medians of a random triangle intersect each other in the centroid of the triangle.

**Example 6.11** “A tangent to a circle is by definition a line that intersects that circle in exactly one point.”

<table>
<thead>
<tr>
<th>Article</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>a (tangent)</td>
<td>Definition</td>
</tr>
<tr>
<td>a (circle)</td>
<td>Generation</td>
</tr>
<tr>
<td>a (line)</td>
<td>Noun</td>
</tr>
</tbody>
</table>
This fragment describes another definition, now in the context of a circle. We recognize this context because 'that circle' refers to 'a circle', so we must put up a flag to generate such a circle in the beginning. We also see the use of the uniqueness quantification here, as introduced in section 3.4.1. This notation is commonly used to narrow down 'existence' to 'unique existence', i.e., the existence of precisely one.

Example 6.12 “A point is called a vertex of a rhombus if that point is the end point of a side of that rhombus.”

<table>
<thead>
<tr>
<th>Article</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>a (point)</td>
<td>Noun</td>
</tr>
<tr>
<td>a (vertex)</td>
<td>definition</td>
</tr>
<tr>
<td>a (rhombus)</td>
<td>generation</td>
</tr>
<tr>
<td>the (end point)</td>
<td>∃ (or Abst)</td>
</tr>
<tr>
<td>a (side)</td>
<td>∃ (or Abst)</td>
</tr>
</tbody>
</table>

This fragment describes the definition of a vertex of a rhombus. Note that we translated 'the end point' with an existential quantification, because a side has two end points, of which only one of them has to be equal to (in this case) P, in order for P to be a vertex of R indeed. Additionally we want to point out that this definition could be formulated by typing P instead of using another ∃ just as well: ‘a vertex of R := Noun P a point ∃ S a side of R ∃ Q an end point of S (P = Q)’. This last
remark also makes clear that we can replace the Noun – ∃ combination by the equivalent Abst – Noun combination (see section 5.8).

The examples 6.10 to 6.12 are all examples for which we can imagine ourselves what they are roughly about with only a bit of mathematical knowledge. However, we can also formulate more abstract examples, using generic names instead of mathematical notions. Example 6.13 will show that we can still formalize such an abstract example into WTT using flag notation. This example has been borrowed from [12].

Example 6.13 "An α of the β of a γ is equal to the δ of an ε of that γ."

<table>
<thead>
<tr>
<th>Article</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>an (α)</td>
<td>∀</td>
</tr>
<tr>
<td>the (β)</td>
<td>t</td>
</tr>
<tr>
<td>a (γ)</td>
<td>generation</td>
</tr>
<tr>
<td>the (δ)</td>
<td>t</td>
</tr>
<tr>
<td>an (ε)</td>
<td>∃</td>
</tr>
</tbody>
</table>

\[
x : a \gamma \\
∀ y : \alpha \text{ of the } \beta \text{ of } x \exists z : \alpha \text{ of } x \quad (y \text{ is equal to the } \delta \text{ of } z)
\]

In this fragment Greek letters are used instead of mathematical notions. We generated an x of type γ as the context for our statement. A little bit tricky in this example is, that the phrase 'an α' in fact hides a universal quantifier. Even if this sentence would have started with "The α of the β of a γ...", i.e., with a definite article, then it would still be translated with the ∀ binder, because the statement describes something that holds for all objects of type γ.

6.7 Lines and books

In WTT we introduce two 'new' notions at the level of texts, viz. books and lines. They are not really new to us, because intuitively we already know what they usually mean. We will call a complete WTT text, formalized using flag notation, a book from now on. Since 'normal' books that we can find in stores or libraries contain a number of lines, we also consider a WTT book to be a list of lines. In turn, we define a line to contain either a statement or a definition, relative to a context.
Note that the book notation is the only official format to display a formalized WTT text. Therefore we will display some book notations of mathematical texts in the coming examples as well. The flag notation actually is no more than sugaring. The flags are not part of WTT, but only serve as a means to create an intuitively clarifying view of a WTT text. We give preference to the flag notation in this course, because it indicates the duration of generations and assumptions very explicitly, whereas the book notation does not.

To denote a line from a WTT book, we need a special separation symbol to be able to make a clear distinction between the context-part of a line and the statement -or definition-part of a line. The separation symbol we will use is ‘\( \triangleright \)’ and it is pronounced as ‘induces’. So a line could look like this: ‘\( x \in \mathbb{R}, \ y \in \mathbb{R} \triangleright x + y \in \mathbb{R} \)’. The context-part of the line is written in front of ‘\( \triangleright \)’ and the statement part that is induced from this context is written behind ‘\( \triangleright \)’.

In order to write a WTT text that has been formalized using flag notation into its WTT book representation, we have to number each line in the flag notation first. This makes it easier to refer to specific lines in the WTT book. We will show in Example 6.14 how to write the short and simple flag notation that we have seen already in Example 6.5 into its WTT book representation.

Example 6.14 Numbered flag notation of a WTT text, followed by its WTT book representation.

Flag notation

\[
\begin{align*}
(1) & \quad x \in \mathbb{R} \\
(2) & \quad x \geq 0 \\
(3) & \quad |x| = x, \text{ so } x^2 - 2 |x| + 1 = (x - 1)^2 \geq 0. \\
(4) & \quad x \leq 0 \\
(5) & \quad |x| = -x, \text{ so } x^2 - 2 |x| + 1 = (x + 1)^2 \geq 0. \\
(6) & \quad \text{So, always } x^2 - 2 |x| + 1 \geq 0 \text{ holds.}
\end{align*}
\]

Book notation

\[
\begin{align*}
\forall x \in \mathbb{R}, \ x \geq 0 & \quad \triangleright \quad |x| = x, \text{ so } x^2 - 2 |x| + 1 = (x - 1)^2 \geq 0. \\
\forall x \in \mathbb{R}, \ x \leq 0 & \quad \triangleright \quad |x| = -x, \text{ so } x^2 - 2 |x| + 1 = (x + 1)^2 \geq 0. \\
\exists x \in \mathbb{R} & \quad \triangleright \quad \text{So, always } x^2 - 2 |x| + 1 \geq 0 \text{ holds.}
\end{align*}
\]
From this example we can deduce that the context for the statement in line (3) contains lines (1) and (2), line (5) is induced by lines (1) and (4) and line (6) only has line (1) as its context. In general we say that for every line on which the scope of a flagpole ends in the flag notation, we also get a line in the book representation. The context-part belonging to each line of a WTT book is formed by all the generations and assumptions of which the flagpole is present in front of this line.

The following example is taken from [9] and is an excellent example to show the whole trajectory from a mathematical text, via a formalization using flags to a WTT book representation. In the book representation we will only mention the line numbers referring to the line numbers in the flag notation from now on.

**Example 6.15** Translation of a mathematical text into its flag notation once more. Moreover, the WTT book representation of the flagged formalization is given.

**Mathematical text**

"Let \( h \neq 0 \), let \( f \) be a function from \( A \) to \( \mathbb{R} \), \( a \in A \) and \( a + h \in A \). Then \( \frac{f(a+h) - f(a)}{h} \) is the difference quotient of \( f \) in \( a \), with difference \( h \). We call \( f \) differentiable at \( x = a \) if \( \lim_{h \to 0} \frac{f(a+h) - f(a)}{h} \) exists. The function \( f(x) = \sqrt{|x|} \) is not differentiable at \( 0 \)."

**Flag notation**

<table>
<thead>
<tr>
<th>Line</th>
<th>Notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( A \subseteq \mathbb{R} )</td>
</tr>
<tr>
<td>2</td>
<td>( f : A \to \mathbb{R} )</td>
</tr>
<tr>
<td>3</td>
<td>( a \in A )</td>
</tr>
<tr>
<td>4</td>
<td>( h \in \mathbb{R} )</td>
</tr>
<tr>
<td>5</td>
<td>( h \neq 0 )</td>
</tr>
<tr>
<td>6</td>
<td>( a + h \in A )</td>
</tr>
<tr>
<td>7</td>
<td>the difference quotient of ( f := \frac{f(a+h) - f(a)}{h} )</td>
</tr>
<tr>
<td>8</td>
<td>( f ) is differentiable at ( a := \lim_{h \to 0} \frac{f(a+h) - f(a)}{h} ) exists</td>
</tr>
<tr>
<td>9</td>
<td>( \neg (\lambda x \in \mathbb{R} , (\sqrt{</td>
</tr>
</tbody>
</table>
We will make a few important remarks to acquire a better insight into it:

- In line (1) we declared \( A \) to be a subset of \( \mathbb{R} \). This was not explicit in the original text, though it was necessary because otherwise we could not speak of a function from \( A \) to \( \mathbb{R} \) in line (2).
- It is important to keep track of our context administration. With this we mean that it is important to find a minimal context for a definition or statement and we should avoid contexts with scopes that are too long. For that reason we cut off the flagpoles from the flags in lines (4) – (6) for the definition in line (8). We don't need the declaration of \( h \) because \( h \) already occurs as a bound variable in \( \lim_{h \to 0} \frac{f(a+h)-f(a)}{h} \). Also the assumptions in lines (5) and (6) are undesirable for the definition in line (8).
- Line (8) can be rewritten as the definition of an adjective if we like. This would be the adjective differentiable, which with the Adj binder would look like this:

\[
\text{differentiable at } a := \text{Adj}_{f:A \to \mathbb{R}} \left( \lim_{h \to 0} \frac{f(a+h)-f(a)}{h} \text{ exists} \right) .
\]

However, this requires us to remove line (2) from our derivation, since \( f \) occurs as a subscript of the Adj binder now. Moreover, this requires us to slightly adapt line (9) as follows:

\[
\forall (x : \mathbb{R}) : \text{differentiable at } 0 .
\]

- The existence of the limit stated in line (8) of course should also have been defined beforehand. However, this is not a proof obligation for us: remember that WTT is not about correctness but about formalization and structuring of mathematical texts.
- In line (9), none of the previously put up flags is still 'alive', meaning that the statement in that line needs no context at all. We denote this by the symbol for the empty context \( \emptyset \).

6.8 Criticism on mathematical texts

Mathematical texts that we can find in mathematical textbooks or papers are often not explicit enough. With this we mean that in these texts things that are obvious to the author are frequently omitted. These omissions may cause more difficulties, such as ambiguity, and context hiding, which makes the structure of the entire text unclear.

Writing down a mathematical text without the flaws mentioned in the previous paragraph is very hard and most authors of such texts do not succeed in it. Because of their large...
mathematical knowledge, things, which are not immediately obvious to people reading these texts (students for example), have become self-evident for them, so they will not write those things down explicitly. In Example 6.16 we will demonstrate how important it is that authors of mathematical texts are as clear and explicit as they can possibly be.

**Example 6.16**  "For a real number \( r \) and a real number \( x \), we write \( r^x := e^{x \ln r} \)."

When we analyze this sentence, we see that two real numbers are needed in the context. So our flag notation of this sentence contains two generations and the actual definition of \( r^x \).

At first sight there may seem nothing wrong or nothing missing. After all, the WTT flag notation here exactly captures what the sentence intended to say. Though there is something missing in the natural language sentence, and therefore also in the flag notation, that is essential. The author of this sentence probably figured that no one reading it would not know that the natural logarithm function ‘\( \ln \)’ is only defined for (real) numbers greater than 0. But instead of assuming this, he could better have formulated the sentence as: “For a real number \( r \), which is greater than 0, and a real number \( x \), we write \( r^x := e^{x \ln r} \)."

We conclude that the flag notation from Example 6.16 needs an extra assumption to make it complete. We will display the correct flag notation belonging to the sentence in Example 6.16 below, in Example 6.17:

**Example 6.17**  Corrected flag notation for Example 6.16.
The previous two examples illustrated a very simple case of what can go wrong when converting mathematical texts into an equivalent flag notation. Most errors in flag notations occur due to the fact that the accompanying mathematical texts are not specified explicitly enough. Even the simplest assumption, as we saw in Example 6.16, can cause errors. In [6], which is in Dutch, De Bruijn makes some remarks about the incorrect and unclear use of natural language we often encounter in mathematical texts.

Criticizing a mathematical text mainly is about analyzing what the author of the text intended to say and compare this to what he is literally saying. The differences that come to light form our criticism on the text. Frequently occurring phenomena we criticize are listed in Table 6.2.

<table>
<thead>
<tr>
<th>Flaw / Error description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Omission of domain</td>
<td>“The pair (x,y) is an element of ordering R.”</td>
</tr>
<tr>
<td>Omission of domain restrictions</td>
<td>“For a real number r, which is greater than 0, and a real number x, we write ( r^x := e^{x \ln r} ).” (See Example 6.16 and Example 6.17)</td>
</tr>
<tr>
<td>Wrongly defined functions</td>
<td>“The function ( f = x^2 + px + q \ldots )”</td>
</tr>
<tr>
<td>Nested definitions</td>
<td>“A circle is the set of all points in ( \mathbb{R}^2 ) that are at a constant distance of a given point (the centre).”</td>
</tr>
</tbody>
</table>

Table 6.2 Flaws that often appear in mathematical texts

We mentioned only a few of the possible flaws in Table 6.2. The first flaw indicates that \( x \) and \( y \) are used without their domain is stated: are they real numbers, elements of another set, or maybe something else? We don’t know. The omission of domain restrictions has been discussed sufficiently in Example 6.16 and Example 6.17.

Functions are often wrongly defined, as we can see when we look at the third flaw. The example that we give does not define a function, but it states an equality. What is meant by it is correctly specified as ‘\( f(x) = x^2 + px + q \)’ and as a formula in WTT it would look like ‘\( f := \lambda x \in \mathbb{R} (x^2 + px + q) \)’.

The last flaw is about nested definitions. In the example that we give of such a flaw, we see that the noun ‘a circle’ is defined and within that definition a veiled definition of ‘the center of a circle’ appears. This can be written down much more orderly by cutting this whole sentence into two pieces which contain one definition each.

We will now give some examples of (short) mathematical texts and we will criticize them as clearly as possible. In this way we want to create insight in how to look at a text and how to judge whether it is explicit enough.
Example 6.18  “Is \( f(x) \) is continuous on the closed interval \([a,b]\), with \( f(a) = A \) and \( f(b) = B \), and if, moreover, \( C \) lies between \( A \) and \( B \), then there exists at least one point \( c \) between \( a \) and \( b \), at which \( f(c) = C \).”

Criticism:
- \( f(x) \) instead of just \( f \).
- Domain and range of \( f \) are missing.
- ‘the closed interval’, should be ‘a closed interval’.
- \( f(a) = A \) and \( f(b) = B \) are both definitions.
- ‘\( C \) lies between \( A \) and \( B \)’ means implicitly that \( A \neq B \).
- ‘at least one’ can be replaced by ‘one’, which is also sufficient.

Example 6.19  “Let \( A \) be a subset of a metric space \((R,d)\). A point \( x_0 \in A \) is called isolated in \( A \) if there exists a neighborhood \( N \) of \( x_0 \) in \((R,d)\) such that the intersection of \( N \) and \( A \) only consists of the point \( x_0 \). If every point in \( A \) is isolated, then each set in the subspace \( A \) is an open set.”

Criticism:
- \( A \) is a subset of \( R \), not of \((R,d)\).
- Domain of \( N \) is missing: \( N \subseteq R \).
- ‘If every point in \( A \) is isolated in \( A \) …’.
- Confused use of subset and subspace.
- ‘Each set in subspace \( A \) …’. Better: ‘Each subset of \( A \) …’.

Example 6.20  “The sequence \( c_n = \left( \frac{n+1}{n} \right)^n \) is strictly increasing, the sequence \( d_n = \left( \frac{n}{n-1} \right)^n \) is strictly decreasing. All the time it holds that \( \frac{n+1}{n} < \frac{n}{n-1} \), so \( c_n < d_n \). From this follows that \( \lim_{n \to \infty} c_n \) and \( \lim_{n \to \infty} d_n \) exist. Notice that \( d_{n+1} = \left( \frac{n+1}{n} \right)^{n+1} = \frac{n+1}{n} \cdot c_n \). So \( \lim_{n \to \infty} c_n = \lim_{n \to \infty} d_n \). This limit is called \( e \).

Let \( f \) be the function sending the real number \( x \) to \( e^x \). Then the derivative of \( f \) is equal to \( f \) itself.”

Criticism:
- ‘The sequence \( c \) …’ instead of ‘The sequence \( c_n = \left( \frac{n+1}{n} \right)^n \) …’.
- Add \( \forall n \in \mathbb{N} \) before \( c_n \), before \( d_n \) and (instead of ‘All the time’) before \( c_n < d_n \).
- What is the type of \( n \)? Natural numbers possibly?
- \( \ldots = \ldots = \ldots \) : not very elegant.
• Omitted that \( c \) is undefined for \( n = 0 \) and \( d \) is undefined for \( n = 0 \) and \( n = 1 \).
• What is the range of \( f \)?

**Example 6.21** “A circle is the set of all points in \( \mathbb{R}^2 \) that are at a constant distance of a given point (the center). A line through the center of a circle is called a diameter. A tangent to a circle is a line that intersects the circle in precisely one point. That point is called the point of tangency. A tangent is always perpendicular to the diameter that goes through the point of tangency.”

**Criticism:**
• What is ‘a constant distance’?
• ‘A circle is the set …’: definite and indefinite articles mixed up.
• Definition of ‘the center’ afterwards (see
• Table 6.2).
• A diameter is not a line, but a line segment.
• ‘A tangent to a circle … the circle …’: Which circle?
• ‘A tangent is always …’ : Better: ‘A tangent to a circle is always …’.

**6.9 Summary**

In this chapter, we started by discussing the relations that exist between sentences. We introduced the notion of context, which is very important in WTT. We discussed the three types of context sentences, viz. definitions, generations and assumptions, and we presented demarcations for each of them, using flags.

Articles in natural language frequently fulfill varying roles in WTT. Especially the indefinite article ‘a’ can ‘disguise’ itself, which makes it more difficult to interpret the text correctly. We provided a number of examples to illustrate this.

The last part of this chapter made us familiar with the notions book and line. A WTT text is called a book and consists of lines. Though book notation is the only official WTT format to represent a formalized text with, we have given preference to the use of the flag notation. We gave preference to the use of flags because they formalize and analyze a text in a way intuitively more appealing to the human mind.

Finally we showed by a number of examples that mathematical texts, as found in textbooks on mathematics, are generally not formulated specific enough, leaving things to the imagination of the reader which could have been easily added to the original text.
these examples we showed how to look critically at a text and how to formulate criticism about it.
7 Examples

In this last chapter, we provide some larger examples than we have already seen in chapter 6. To keep things as orderly as possible, we devote a separate section to each example. The examples in this chapter consist of mathematical texts in natural language, which are translated into a corresponding flag notation. For the first two examples we extensively discuss how we constructed them and which considerations we took into account while constructing them. We do this in a stepwise manner, taking one sentence at a time. The last three examples are displayed without such a discussion, but they are thoroughly analyzed and criticized in the same way we did previously in section 6.8.

Once more, we want to state clearly that there may be other solutions to translate the example texts into flag notation that are correct as well. However, we choose not to end up enumerating all possible alternatives here, because that would only distract us from the actual goal of this chapter, which is to show some elaborate flag notations and to criticize the way the texts are formulated.

7.1 Example 1

Mathematical text

"A sequence is a mapping $\mathbb{N} \rightarrow \mathbb{R}$. We denote the value of a function $a(n)$ as $a_n$.

A sequence $a$ has limit $A$ if for every $\varepsilon > 0$ there exists a natural number $N$, such that $|a_n - A| < \varepsilon$ if $n > N$.

A sequence is called convergent if the limit of $a$ exists.

The sequence $1, \frac{1}{4}, \frac{1}{9}, \frac{1}{16}, ...$ has limit 0."

Considerations

- **A sequence is a mapping $\mathbb{N} \rightarrow \mathbb{R}$.**
  The first sentence starts with the phrase "A sequence is ...". Such a construction tells us that the following part of the phrase is a definition, defining what a sequence is. Since it starts with the indefinite article 'a', we started our translation with a Noun binder here. We can choose how to interpret this definition now:

  - We consider it to be defined as 'a mapping $\mathbb{N} \rightarrow \mathbb{R}$', of which nothing more is said.
  - We consider it to be defined as 'a mapping', of which we state that its type is $\mathbb{N} \rightarrow \mathbb{R}$.
Choosing the first leads to the translation \( a \text{ sequence} := \text{Noun}_a : N \rightarrow \mathbb{R} \ (\text{true}) \), which we used in our example. The second interpretation would lead to \( a \text{ sequence} := \text{Noun}_a : a \text{ mapping} \ (a : N \rightarrow \mathbb{R}) \).

- We denote the value of a function \( a(n) \) as \( a_n \).

To translate this sentence, we need an instance of the sequence we have just defined, as well as an instance of a natural number which we call \( n \). This inevitably means that we should put up two pointed flags, generating \( a \) and \( n \) as a sequence and a natural number, respectively.

In the context of these two generations we can now formulate the definition that \( a_n \) is denoted, i.e., defined, as \( a(n) \) as follows: \( a_n := a(n) \).

- A sequence \( a \) has limit \( A \) if for every \( \varepsilon > 0 \) there exists a natural number \( N \), such that \( |a_n - A| < \varepsilon \) if \( n > N \).

The sentence "\( a \) has limit \( A \)" is defined here, within the context of the previously generated sequence \( a \), but outside the scope of the flag that generated the natural number \( n \). Since the domain of \( A \) is left implicit, we first put up a generation flag that instantiates \( A \) as real number. Then we can write down the definition itself. The structure of the sentence tells us that we should start with a universal quantifier binding \( \varepsilon \), followed by an existential quantifier binding \( N \) and (again) a universal quantifier binding \( n \). Furthermore, the \( \varepsilon \) is restricted to positive values and the \( n \) is restricted to values strictly greater than \( N \). Finally, the statement \( |a_n - A| < \varepsilon \) follows after these three quantifiers. The resulting translation is:

\[
\text{\( a \) has limit \( A \)} : \iff \forall \varepsilon \exists N \forall n : N |a_n - A| < \varepsilon.
\]

- A sequence is called convergent if the limit of \( a \) exists.

This sentence can be interpreted in two different ways again:

- We consider it to be a definition of the sentence "\( a \) is convergent", within the context of the previously generated sequence \( a \).
- We consider it to be a definition of the adjective "convergent", placed outside the scope of the generated sequence \( a \).

We have chosen the last option here. We could have chosen the first option as well, which would then look like this: \( a \text{ is convergent} : \iff \exists A \in \mathbb{R} \ (a \text{ has limit } A) \).

We stress again that this definition is only correct when the flagpole of the generated sequence \( a \) runs up to and including the line containing this definition.
The sequence $1, \frac{1}{4}, \frac{1}{9}, \frac{1}{16}, \ldots$ has limit 0.

This last sentence shows a particular sequence and states that it has limit 0. Let us first look at the sequence and see if we can discover a pattern in it. The pattern that is present here can be found in the denominator and can be characterized by rewriting this sequence as a function using the $\lambda$ binder and binding the natural number $n$.

Since this sentence states that this particular sequence has limit 0, we can simply make use of our definition of "a has limit $A$" here: $\lambda_{n \in \mathbb{N}} \left( \frac{1}{(n+1)^2} \right)$ has limit 0.

By taking all these remarks into account and by making the choices that we explicitly made whenever there were more possible translations, we constructed the following flag notation.

Flag notation

\[
\text{a sequence} := \text{Noun}_{a: \mathbb{N} \rightarrow \mathbb{R}} \text{ (true)}
\]

\[
a : \text{a sequence}
\]

\[
n \in \mathbb{N}
\]

\[
a_n := a(n)
\]

\[
A \in \mathbb{R}
\]

\[
a \text{ has limit } A : \iff \forall \varepsilon \in \mathbb{R} | \varepsilon > 0 \exists N \in \mathbb{N} \forall n \in \mathbb{N} | n > N (|a_n - A| < \varepsilon)
\]

\[
\text{convergent} := \text{Adj}_{a : \text{a sequence}} \exists A \in \mathbb{R} (a \text{ has limit } A)
\]

$\lambda_{n \in \mathbb{N}} \left( \frac{1}{(n+1)^2} \right)$ has limit 0.
7.2 Example 2

Mathematical text

"A relation between \( X \) and \( Y \) is a function from \( X \times Y \) to the Booleans. The graph of a relation between \( X \) and \( Y \) is the set of all pairs from \( X \times Y \) that have R-value true. We call such an \( R \) functional if for every \( x \in X \) there exists exactly one \( y \in Y \) such that \( R(x,y) \) is true. The graph of such a functional relation is called a functional graph in \( X \) and \( Y \) or a mapping of \( X \) onto \( Y \). Example: If \( b \) is an element of \( Y \), then \( X \times \{ b \} \) is a functional graph, called the constant mapping of \( X \) onto \( Y \) with value \( b \). For \( Y = X \) the relation \( y = x \) is functional. The graph of this relation is the set of all pairs \((x,x)\) and is called the diagonal of \( X \times X \)."

Considerations

- **A relation between \( X \) and \( Y \) is a function from \( X \times Y \) to the Booleans.**
  This sentence defines what ‘A relation between \( X \) and \( Y \)’ is, but it leaves implicit what \( X \) and \( Y \) actually are. So we first have to generate the sets \( X \) and \( Y \), which may be done by using just one flag, because \( X \) and \( Y \) are both of the same type here.
  
  This particular relation is defined as ‘a function from \( X \times Y \) to the Booleans’, so our translation must start with a Noun, binding a function that maps elements from \( X \times Y \) onto either true of false (T or F). Since a Noun must be followed by a statement, but nothing additional is stated about this function, we can suffice by writing true (T) behind it, resulting in the following formula: ‘a relation between \( X \) and \( Y \) := Noun_{f: X \times Y \to \{T,F\}}(T)’.

- **The graph of a relation between \( X \) and \( Y \) is the set of all pairs from \( X \times Y \) that have R-value true.**
  The graph of a relation between \( X \) and \( Y \) is defined here, so first we need a flag generating a relation: ‘\( R : a \) relation between \( X \) and \( Y \)’. Then we can define ‘the graph of \( R \)’ as a set as follows: ‘the graph of \( R \) := \{(x,y) \in X \times Y \mid R(x,y)\} ’.

- **We call such an \( R \) functional if for every \( x \in X \) there exists exactly one \( y \in Y \) such that \( R(x,y) \) is true.**
  We have two options to interpret this sentence:
  - We consider it to be a definition of the sentence ‘\( R \) is functional’.
  - We consider it to be a definition of the adjective ‘functional’ (on relations).
We choose for the first option, which entails that this definition should be placed within the scope of the previously generated relation $R$ (and consequently also within the scope of the generated sets $X$ and $Y$). The definition looks like this: ‘$R$ is functional $\iff \forall x \in X \exists! y \in Y (R(x, y))$’.

If we would have chosen the second option, we would have gotten the following definition: ‘functional $\coloneqq \text{Adj}_{R \text{ a relation between } X \text{ and } Y} \forall x \in X \exists! y \in Y (R(x, y))$’.

We have to be careful here though, because we have to place this definition outside of the scope of the flag that previously generated $R$, but within the scope of the flag that generated $X$ and $Y$ in the beginning!

- **The graph of such a functional relation is called a functional graph in $X$ and $Y$ or a mapping of $X$ onto $Y$**.

This sentence contains two definitions for which we both still need the sets $X$ and $Y$ in our context. The first one defines ‘a functional graph in $X$ and $Y$’. We can now opt for the Noun – $\exists$ combination to translate this sentence into WTT. The Noun binder binds a graph $G$, which is a subset of $X \times Y$, whereas the existential binder binds a functional relation $R$ between the sets $X$ and $Y$. This results in the following definition:

‘a functional graph in $X$ and $Y$ $\coloneqq \text{Noun}_{G \subseteq X \times Y} \exists R \text{ a relation between } X \text{ and } Y \mid R \text{ is functional} (G = \text{the graph of } R)$’.

The second definition defines ‘a mapping of $X$ onto $Y$’. This definition can be straightforwardly translated because it is another way of saying ‘a functional graph in $X$ and $Y$’, which we have just defined. So we can suffice with the following definition:

‘a mapping of $X$ onto $Y$ $\coloneqq$ a functional graph in $X$ and $Y$ ’.

- **Example: If $b$ is an element of $Y$, then $X \times \{b\}$ is a functional graph, called the constant mapping of $X$ onto $Y$ with value $b$**.

We will split up this sentence, because the first part, up to and including the word ‘graph’ is a statement whereas the last part contains a definition.

The statement part of it in fact is a veiled universal quantification, because it states something that is true for every element of $Y$. We still need the sets $X$ and $Y$ in the context for this statement. It can be formulated, using the definition of ‘a functional graph in $X$ and $Y$’, like this:

‘$\forall b \in Y (X \times \{b\} : \text{a functional graph in } X \text{ and } Y )$’.

The second part of the sentence is a definition of ‘the constant mapping of $X$ onto $Y$ with value $b$’. Before we can speak of something with value $b$ we have to put up a pointed flag to generate $b$ as an element of $Y$. After this $b$ has been
introduced, it has become rather simple to translate the definition into WTT: ‘the constant mapping of X onto Y with value b := X x \{b\}’.

- **For Y = X the relation y = x is functional.**
  The part ‘For Y = X’ is an assumption we have to make before we continue. Therefore, we put up a straight flag within the context of the sets X and Y. Now comes the tricky part. It says that ‘the relation y = x is functional’, but what it actually means is that ‘equality’, i.e., ‘=’, is a functional relation in X and Y. So this is a typing sentence that can be translated, in the context of the assumption ‘Y = X’, as follows: ‘equality: a functional relation in X and Y’. (it is also allowed to use ‘=’ as left hand side of the typing sentence).

- **The graph of this relation is the set of all pairs (x,x) and is called the diagonal of X X X.**
  Again, we split this sentence into two parts, of which the first part is up to and including ‘(x,x)’. This first part states that ‘the graph of this relation’ is equal to the set containing all pairs ‘(x,x)’. Of course, ‘this relation’ refers to the equality relation of the previous sentence, so the translation becomes: ‘the graph of ‘equality’ = \{(x, y) \in X \times Y \mid x = y\}’.

In the second part of this sentence a definition occurs, defining ‘the diagonal of X X X’. This definition looks as follows:
‘the diagonal of X X X := the graph of ‘equality’’.

Following the list of considerations, which we extensively spelled out here, results in the flag notation that is displayed here:
Flag notation

\( X, Y : a \text{ Set} \)

a relation between \( X \) and \( Y \) := \( \text{Noun}_{f : X \times Y \rightarrow \{T, F\}} \( T \) \)

\( R : a \text{ relation between } X \text{ and } Y \)

the graph of \( R := \{ (x, y) \in X \times Y \mid R(x, y) \} \)

\( R \) is functional \( \iff \forall x \in X \exists y \in Y \( (R(x, y) \)

a functional graph in \( X \) and \( Y := \text{Noun}_{G \subseteq X \times Y \exists R : a \text{ relation between } X \text{ and } Y \mid R \text{ is functional}} \( G = \text{the graph of } R \) \)

a mapping of \( X \) onto \( Y := a \text{ functional graph in } X \text{ and } Y \)

\( \forall b \in Y (X \times \{b\} : a \text{ functional graph in } X \text{ and } Y) \)

\( b \in Y \)

the constant mapping of \( X \) onto \( Y \) with value \( b := X \times \{b\} \)

\( X = Y \)

'equality' : a functional relation between \( X \) and \( Y \)
the graph of 'equality' = \( \{ (x, y) \in X \times Y \mid x = y \} \)

the diagonal of \( X \times X := \text{the graph of 'equality'} \)
7.3 Example 3

Mathematical text

"A function $f : A \to \mathbb{R}$ is bounded if there exists a real number such that the absolute value of $f(x)$ is not greater than that real number, for all $x \in A$.

A function $f : [a, b] \to \mathbb{R}$ is called continuous in the point $x_0 \in [a, b]$, if

$$\lim_{x \to x_0} f(x) = f(x_0).$$

A function $f$ is called continuous if it is continuous in every point of its domain.

The function $x \to x^2$ is continuous.

Let $f : [a, b] \to \mathbb{R}$ be a continuous function. Then $f$ is bounded."

Flag notation
Criticism

- \( A \) is not introduced.
- \( \exists_{r \in \mathbb{R}} \forall_{x \in A} \) or \( \forall_{x \in A} \exists_{r \in \mathbb{R}} \)?
- \( a \) and \( b \) are not introduced.
- the point \( x_0 \)?
- 'A function \( f \ldots \)': Domain? Co-domain?
- \( x \rightarrow x^2 \): a function?
- It is left implicit that a limit must exist.

7.4 Example 4

Mathematical text

"Definition. A pyramid is a polyhedron of which one plane is a polygon whereas the other planes are triangles with a common vertex. This polygon is the base of the pyramid and the triangles are the faces. The common vertex of the faces is called the apex. The height is the distance from the apex to the base. Theorem. The volume of a pyramid is equal to \( \frac{1}{3}gh \), where \( g \) is the area of the base and \( h \) the height."
Flag notation

A pyramid is defined as a noun phrase denoting a polyhedron with a plane for its base and three planes for its faces, each of which is a triangle. The vertex of a triangle is defined as a point, and the height of a triangle is defined as the distance from its vertex to its base. The volume of a pyramid is given by the formula \( V = \frac{1}{3} \times \text{area of base} \times \text{height} \).

Criticism:
- Can the base itself be a triangle?
- What is the base if all faces are triangles?
- "The other planes of the polyhedron ...".
- "The pyramid ...": which pyramid?
- "This polygon is ..." and "the triangles are ...": unclear that these are definitions.
- "The height of the pyramid ..." (twice).
- "The volume ... where ...": two definitions afterwards. Preferably present them more clearly as definitions and rather beforehand than afterwards.
7.5 Example 5

Mathematical text

“We say that the set $A$ is a subset of the set $B$, if all elements of $A$ are elements of $B$ as well. Notation: $A \subseteq B$.

For all sets holds that $\emptyset \subseteq A$ and $A \subseteq A$.

The intersection $A \cap B$ of the sets $A$ and $B$ is the set of all elements that are both in $A$ and in $B$.

We call $A$ and $B$ disjoint if $A \cap B = \emptyset$.

Let $V$ be a non-empty set. A classification of $V$ is a set of subsets of $V$ with the following properties:

1. none of the subsets is empty;
2. every two distinct subsets are disjoint;
3. the set of all elements of the subsets together, is equal to $V$.

Let $K$ be a classification of $V$. Notice that an element of $K$ is a set! Those elements are called the classes of $K$.

The relation $R$ on $V$ is given by: $aRb$ if and only if $a$ and $b$ are elements of the same class. $R$ is an equivalence relation on $V$.”
Flag notation

\[ A : a \text{ Set} \]

\[ B : a \text{ Set} \]

**A is a subset of B:** \( \forall x \in A \ (x \in B) \)

\( A \subseteq B :\Rightarrow A \text{ is a subset of } B \)

\( \forall A : a \text{ Set} \ (\emptyset \subseteq A) \)

\( \forall A : a \text{ Set} \ (A \subseteq A) \)

\[ A : a \text{ Set} \]

\[ B : a \text{ Set} \]

**The intersection of A and B:** \( \exists v : \text{ a set} \ ((x \in V \iff x \in A \land x \in B)) \)

\( A \cap B = \text{the intersection of } A \text{ and } B \)

\( A \text{ and } B \text{ are disjoint} :\Rightarrow A \cap B = \emptyset \)

\[ V : a \text{ Set} \]

\( V \neq \emptyset \)

**A classification of V:** \( Noun \_{K : a \text{ Set}} (\forall x \in K \ (x \subseteq V) \land \forall x, y \in K \ (x \neq \emptyset) \land \forall x, y \in K \ (x \text{ and } y \text{ are disjoint}) \land t_{W : a \text{ Set}} (u \in W \iff \exists x \in K \ (u \in X)) = V \)
K : a classification of V

∀x ∈ K (X : a Set)

a class of K := Noun x ∈ K (true)

R := λ(a,b) ∈ V² ∃x ∈ K (a ∈ K ∧ b ∈ K)

R : an equivalence relation on V

Criticism

- 'the intersection A ∩ B ...' instead of 'the intersection \( A \cap \ldots \)'.
- 'For all sets A holds that ...'.
- 'The intersection of A and B ...' instead of 'The intersection A ∩ B ...'.
- 'We call the sets A and B ...'.
- '1. none of the subsets of V is empty;' (idem for 2. and 3.)
- 'aRb' : What is a? What is b? Domains?
References


Appendix A: Exercises

Exercises for section 2.4

1) Examine the difference between the 'normal', i.e., natural language, meaning and the mathematical meaning of the following words:

'origin', 'argument', 'relation', 'power', 'function', 'monotonic', 'domain', 'diagonal', 'inequality' and 'integral'.

2) Give the universal meaning (in natural language) of the following mathematical standard words:

∀, ℤ, ℚ, Σ, ⇒, ≥, ∪, e, lim, π, cos and ∞

Exercises for sections 2.5 and 2.6

1) Classify the following sentences as being either a statement/proposition, a definition or a typing sentence.

a) ‘γ:= y/2’

b) ‘x ∈ ℤ’

c) ‘γ= y/2’

d) ‘a²+by=10’

e) ‘x: an integer’

f) ‘R ⊆ V²’

g) ‘∀ s ∈ S, s=0 (s ⊆ t)’

h) ‘b:= the bisector of D’

2) Classify the following cases as an object name or a proposition.

a) ‘x = 17’

b) ‘The collection of natural numbers smaller than 10’

c) ‘6 and 17 are elements of the collection of sixfolds’

d) ‘an equilateral triangle’

e) ‘the smallest integer bigger than π’

f) ‘∀ p,q (3, (¬q ⇒ (s ∨ p)))’

g) ‘cos (2π −1) ∈ R’
Exercise for section 3.2

1) Recall the 7 formulas in section 3.2 on page 11. For each of these formulas classify the occurring formula symbols according to the categories in Table 3.1. Also indicate what the arity of each occurring formula symbol is. (unary, binary, ternary, etc.)

Exercises for section 3.3

1) What is wrong in the following formula?

\[ g(f(a \cdot b) + f(c), g(h(c + b), d - a)) > h(f(b) - g(a, a)) - (d \cdot g(h(f(c), d), f(a))) \]

(Hint: check parentheses and arities)

2) Write the following formulas in Polish and reverse Polish notation.

a) \('n + 2 \cdot \sqrt{n}!\)'

b) The first part (in front of the ‘>’ symbol) of the formula in exercise 1) above.

c) \('\frac{x^2 - 6x^2}{5x} = 0\)'

3) Identify the main formula symbol(s) in the following formulas:

a) \('P \land \neg S \lor Q \Rightarrow R \land P \iff \neg Q \Rightarrow \neg S\)'

b) \('(2x^3 + 5y^2 + 5xy - 3)\)\(^{\frac{1}{3}}\)'

4) Give tree representations of the following formulas:

a) \('\sqrt{2 + \sqrt{3}} \cdot a\)'

b) \('\sin(y) = 2 \cdot \sin(\frac{1}{2} y) \cdot \cos(\frac{1}{2} y)\)'

c) \('p(p(q(r), q(r)), p(q(r), q(r))) < q(r)\)'

d) \('\ln(x + z \cdot y) \leq y^2 \leq z\)'

e) \('\frac{1}{2}(z + c)^2 - \frac{3}{\sqrt{3}}\)'

5) Does the outcome of the formula ‘\(a - b \cdot c + d \cdot e / f\)’ when reading it according to the left-associativity rule, differ from the outcome when reading it according to the right-associativity rule?

6) Write the formula of exercise 5) above in tree representation. Assume the hierarchy of Table 3.3 is used, as well as the left-associativity rule.
7) Assume we have the following hierarchy of formula symbols:

```
- \. + / 
```

Add parentheses to the formula ‘\( a + b / d \cdot c - a \cdot b + c / a \)’ according to this hierarchy and the left-associativity rule. Also, give the tree representation of this formula.

Exercises for section 3.4

1) Draw trees of the following formulas containing binders. Also indicate the places where variables are initially bound by drawing dashed lines in the trees.

a) ‘\( \forall x \in \mathbb{R} \exists y \in \mathbb{R} \, y > 0 \land y \neq x \) (\( y > x + 2 \))’

b) ‘\( \sum_{n=1}^{m} n^2 + \sum_{n=0}^{m} 3n^2 \)’

c) ‘\( \exists f : \mathbb{R} \rightarrow \mathbb{R} \, \forall x \in \mathbb{R} \) (\( f(x) = f(-x) \))’

d) ‘\( \{ n \in \mathbb{Z} \mid (n + m)^2 \leq 12 \} \)’

2) Determine the following formulas in which substitutions have to be done:

a) Subst(‘\( 3a \), ‘\( b \), ‘\( b^3 + 2b^2 + 3a - c \)’)

b) Subst(‘\( 2(x - 1) \), ‘\( x \), ‘\( x^2 + 3x + 2 \)’)

c) Subst(‘\( x + 1 \), ‘\( y \), Subst(‘\( y + x \), ‘\( x \), ‘\( x + y + 1 < xy \)’) )

d) Subst( (‘\( a \), ‘\( y + 2b \)’), (‘\( b \), ‘\( x \)’), ‘\( b^{a+y} - 3x + b \)’)

Exercise for section 4.2 and 4.3

1) Determine the variables and constants in the following cases. Also say whether the occurring variables are free or bound.

a) \( f (2 \cdot 4, p, \pi) \)

b) \( \forall x \in T (\neg (x \in U)) \Leftrightarrow (T \cap U = \emptyset) \)

c) \( \text{l runs parallel with m} \)
Exercises for section 4.4

1) Examine in which of the following cases we are dealing with a generic noun and in which cases we are dealing with a specific noun.

a) ‘\( \mathbb{N} \)’
b) ‘a unit vector in \( \mathbb{R}^2 \)’
c) ‘the greatest common divisor of 48 and 84’
d) ‘real number’
e) ‘the identity element of multiplication’
f) ‘\( \lim_{x \to \infty} \frac{1}{e^x} \)’

2) Examine in which of the following cases we are dealing with a function description and in which cases we are dealing with a predicate description.

a) ‘\( \lambda_{x \in \mathbb{R}}(x^3) \)’
b) ‘\( \lambda_{m \in \mathbb{N}} \exists_{n \in \mathbb{N}} (m^3 - n^3 = m - n) \)’
c) ‘\( \lambda_{x \in (\mathbb{R}, \forall x)} (\sin(x) - \cos(x + \frac{\pi}{2})) \)’
d) ‘\( \lambda_{n \in \mathbb{Z}} (\{ k \in \mathbb{Z} | k \geq n + 1 \}) \)’

3) Translate the following formulas into natural language, without bound variables and as few mathematical symbols as possible.

a) ‘Noun \( \exists_{k \in \mathbb{Z}} (x = \pi^k) \)’
b) ‘\( t : \text{Abst}_{n \in \mathbb{N}} (3n^2) \)’
c) ‘\( \neg \exists_{x \in \mathbb{N}} (\cos(n) > e^{x+1}) \)’
d) ‘\( f := \lambda_{m \in \mathbb{N}} (\frac{m}{n}) \)’
e) ‘idempotent := Adj_{f: \mathbb{R}^2 \to \mathbb{R}} \forall_{x \in \mathbb{R}} (f(x, x) = x)’
f) ‘\( t \in \mathbb{N} \land (n^2 - 10n + 16 = 0) \)’
g) ‘\( \forall_{x \in \mathbb{R}} (\sin(x) = 1 \Rightarrow \exists_{k \in \mathbb{Z}} (x = k \cdot \frac{\pi}{2})) \)’
h) ‘\( \text{Abst}_{a, b : \text{a prime number}} (a + b) \)’
i) ‘\( \text{Adj}_{a : \mathbb{N} \to \mathbb{R}} \forall_{n \in \mathbb{N}} (a(n) - a(n+1) < 0) \)’
j) ‘\( \text{Abst}_{m \in \mathbb{N}} (\sqrt{m!}) \)’

4) Draw trees of the formulas c), d), e), h) and j) of the previous exercise.
Exercise for section 5.2 – 5.6

1) Translate the following natural language expressions into an equivalent WTT formula.

a) “x is a real number between 0 and 5.”

b) “The square of the logarithm from an odd natural number.”

c) “An integer that is the difference of two squares of natural numbers.”

d) “By definition, P is the predicate ‘even and not greater than 100’ on the natural numbers unequal to 0.”

e) “The solution of the equation $x^2 + 4 = 4x$ is greater than 1.”

f) “A function on the real numbers is called positive is if all possible function values are positive.”

g) “Let $f$ be the function sending a natural number to set of all its divisors except 1 and that natural number itself.”

h) “A pyramid is a polyhedron of which one plane is a polygon, while the other planes are triangles with a common vertex.”