

A refinement of de Bruijn's formal language of mathematics *

Fairouz Kamareddine (fairouz@macs.hw.ac.uk)[†]

Mathematical and Computer Sciences, Heriot-Watt University, Edinburgh EH14 4AS, UK

Rob Nederpelt (r.p.nederpelt@tue.nl)

Math. and Computing Sc., Eindhoven Univ. of Technology, P.O.Box 513, 5600 MB
Eindhoven, NL

1997/01/23

Abstract. We provide a syntax and a derivation system for a formal language of mathematics called *Weak Type Theory* (WTT). We give the metatheory of WTT and a number of illustrative examples. WTT is a refinement of de Bruijn's Mathematical Vernacular (MV) and hence:

- WTT is faithful to the mathematician's language yet is formal and avoids ambiguities.
- WTT is close to the usual way in which mathematicians express themselves in writing.
- WTT has a syntax based on linguistic categories instead of set/type theoretic constructs.

More so than MV however, WTT has a precise abstract syntax whose derivation rules resemble those of modern type theory enabling us to establish important desirable properties of WTT such as strong normalisation, decidability of type checking and subject reduction. The derivation system allows one to establish that a book written in WTT is well-formed following the syntax of WTT, and has great resemblance with ordinary mathematics books.

WTT (like MV) is weak as regards correctness: the rules of WTT only concern *linguistic* correctness, its types are purely linguistic so that the formal translation into WTT is satisfactory *as a readable, well-organized text*. In WTT, *logico-mathematical aspects* of truth are disregarded. This separates concerns and means that WTT

- can be easily understood by either a mathematician, a logician or a computer scientist.
- acts as an intermediary between the language of mathematicians and that of logicians.

Keywords: Mathematics, Formal Language, Mathematical Vernacular, Weak Type Theory.

1. Introduction

The way in which mathematical ideas are usually expressed in writing (books, papers, etc.) is *informal*, in the sense that there exists no prescribed syntax for the presentation of mathematical contents. We give the name *Common Mathematical Language* (CML) to this linguistic machinery which mathematicians preferably use to express mathematical content and to communicate with their fellow mathematicians. We concentrate on *written* specimens of CML

It is useful to also have a *formal* language for the same communication purposes. Such a language may act as a substitute for CML. We propose a *Weak*

* We are grateful for the anonymous referees for their useful suggestions.

† Contact author. Address as above. Tel: +44 131 451 3868. Fax: +44 131 451 8179.



Type Theory (WTT), intended to be a formal language for expressing mathematical content. WTT aims to keep as much as possible of the advantages, and to remove as much as possible of the disadvantages of CML.

1.1. A COMPARISON BETWEEN CML AND WTT

We start with a list of what we see as the most salient *ADVANTAGES* of CML:

- *Expressive* CML is suited for expressing all kinds of mathematical mental constructs, ranging from mathematical entities and relations, to mathematical reasonings and theories. It also permits relevant mathematico-linguistic categories, such as definitions, theorems and proofs.
- *Time-honoured* CML has a long tradition and is refined by intensive use.
- *Satisfactory* CML is approved by a large community of mathematicians, and still proves to be an adequate communication medium.
- *Universal* CML is used world wide and offers a standard format to users.
- *Flexible* Not only can CML accommodate many branches of mathematics, but it is also easily adaptable to new developments in mathematics.

However, there are also *DISADVANTAGES* of CML:

- *Informal* Since CML is based on natural language – mixed with mathematical symbols and formulas – it has no well-defined formal basis, suffers of imprecision, and relies on the understanding of the user.
- *Ambiguous* CML inherits the ambiguities of natural language, such as unspecified anaphoric relations (*discourse references*) and overloading of frequently used words. It also introduces new ambiguities in the mathematical extensions to the natural language, as used in CML.
- *Incomplete* In CML, much is left implicit, the writer making appeal to the intuition or common sense of the mathematical reader.
- *Poorly organised* A common mathematical text is only partly structured in textual units. Many structural aspects are omitted or only hinted at.
- *Automation-unfriendly* Since a CML-text is a plain text, its mathematical content cannot be exploited by invoking computer assistance.

We revisit the above comparing WTT and CML. First, CML's advantages:

- *Expressive* The expressivity of WTT is comparable to that of CML.
- *Time-honoured* WTT is novel, but it respects the mathematical traditions.

- *Satisfactory* Though the authors have ample experience with WTT, its satisfactory usability of course still has to be demonstrated.
- *Universal* At the moment only potentially.
- *Flexible* WTT is not as flexible as CML. CML allows both language and meta-language whereas WTT only allows language. So meta-language must be pushed to the language level, which may not be possible.

As regards the disadvantages:

- *Formal* The main gain of WTT over CML is its formality. WTT has a well-defined syntax which provides for uniformity of expression.
- *Unambiguous* Most of the ambiguities in the CML-texts disappear in the translation to WTT. For example, the anaphoric obscurities in CML are resolved in WTT by the strict context management.
- *Incomplete* A text in WTT is still incomplete, but less so than CML. Moreover, WTT can be translated further into more complete theories.
- *Clearly organised* A WTT-text is clearly organized.
- *Automation-friendly* WTT is fit for computerization. In translating WTT into stronger versions, the potentials for automation grow fast.

1.2. FEATURES OF WTT

In our design of WTT, we keep close to the Common Mathematical Language. Our main reason for doing this is *reliability*: it is of uttermost importance that the formal WTT-version of a piece of mathematics covers exactly the intended contents of the original CML-version. Since the latter is informal – already in the mind of the mathematician who devised it, but still in the written version – it is impossible to have an objective check on the correspondence between the original and the formalized text. Hence it is necessary to invoke a *human* judgement on this correspondence, for each piece of mathematics under consideration. This judgement can be optimally trustworthy (albeit never for one hundred percent) if the formal version is as close as possible to the informal one. We call this the *reliability criterion*.

As soon as a reliable formal translation of a piece of informal mathematical text has been made, we are on solid ground. Such a translation will still be far from complete, but since it is formal, it is possible to define further translations into more complete versions. Since these subsequent translations are from formal texts to formal texts, it is much easier to check reliability: one only needs to check the reliability of the translation *procedure*.

We list some useful features of WTT as a formal language for mathematics:

- WTT respects all linguistic categories which a mathematician usually employs, so not only the notions *set* and *element* of a set, but also *noun* and *adjective*, already present in *natural* language and amply used in CML. This brings a WTT-text nearer to the intuition of a mathematician, since the fine details of mathematics are better accounted for in a mixture of mathematical and natural language, than in a set-theoretic setting.
- Although the linguistic categories *noun* and *adjective* seem superfluous, they help to minimize communication losses in this first formalisation step (from CML to WTT) and aid the connection with computational linguistic systems.¹ Sieving superfluous ingredients can happen at a later stage (e.g., in a further translation into type theory).
- WTT avoids the purely set-theoretic setting because mathematicians do *not* have a purely set-theoretic view on mathematics! A collective notion (a *type*) is not always identifiable with a *set* and the notion-subnotion differs from the set-subset. Also, although predicates can be identified with subsets, there are subtle differences between the two.
- Ambiguities in CML are detectable and solvable in WTT.
- WTT also accommodates all sorts of *binders* (such as \cup or Σ).
- WTT has basic notions like: *assumption*, *declaration of a fresh variable*, *definition*, *statement* (including *theorem* and *step in a proof*).
- WTT is consistently structured and uses *contexts*, which are lists of assumptions and variable declarations *setting the stage* for a statement or a definition. These contexts reflect the introductory statements usually expressed in mathematics as e.g. *Let ...* or *Assume ...*.
- The overall form of a text in WTT is a so-called *book*, being a sequence of so-called *lines*, which are statements and definitions, each embedded in their own context. Each WTT-line can be seen as the translation of a mathematical expression stating that something *holds* in a certain context. Hence, one finds in a book a –probably connected– fragment of mathematical knowledge, consisting of lines expressing mathematical facts (like theorems, lemma's, but also steps in a reasoning or in a proof) and lines expressing definitions (possibly in a context).

¹ It may be argued that we concentrate in WTT on adjectives and nouns, and not on other linguistic phenomena like anaphors and ellipses that carry structure in CML. We prefer however to concentrate on the incompleteness aspect of WTT in order to illustrate the usefulness of this style compared with the usually assumed theorem proving style of full formalizations of mathematics.

Of course, we do not expect that mathematicians convert to using WTT exclusively. On the contrary, the common mathematical language, as found in books and papers, is good enough and, moreover, usable and familiar to a high degree. However, WTT is formal and so close to the usual linguistic format used by mathematicians, that it can easily be adopted by mathematicians as a *second language*. This can be advantageous for the following reasons:

- In complex situations, the *second language* WTT can help the mathematician to identify the (logico-mathematical) structure where he works.
- It can also help a person (an expert, a teacher or a student) to be fully aware of the complexity of a mathematical notion, structure or reasoning, in order to better understand the situation.
- WTT provides an excellent basis for communication: it enables that many persons work productively on the same task and that the text administration of a mathematical project gets a firm basis.
- WTT may act as a *lingua franca* or a *mathematical vernacular*² for mathematicians, since it enables a mathematician to express mathematics in a uniform way. As WTT is clearly structured, it forces the mathematics writer to think about the interdependencies of the notions used such as contexts and instantiations of constants.
- WTT can act a *specification language* for mathematics, since it enables the mathematician to explicitly specify which mathematical notions, definitions, statements, theorems and proofs he/she likes to use. In this respect, WTT resembles specification languages in computer science, which enable one to formally represent the *requirements*. In computer science, a specification is *realized* in a computer program. In mathematics, a WTT-book can be *realized* in a *type-theoretic program*: a sequence of lines obeying the rules of some system of type theory.
- A WTT-book acts as a (*mathematical*) *discourse representation structure*, bringing a number of CML-implicit structural relations to surface.
- WTT is easy to use, as shown by experiences with mathematics and computer science students at Eindhoven University of Technology. This started in 1979 when de Bruijn developed a course on the mathematical vernacular. This course became part of the curriculum for mathematics teachers. After de Bruijn's retirement, Nederpelt took the course and continues to teach it today using WTT instead of MV.

² The name *mathematical vernacular* was coined by de Bruijn, who was the first to develop such a *language for mathematics* (see F3 of [14]) upon which this work is based. *Vernacular* means: *the language of a particular group* (in this case: the mathematicians). The word *mathematical vernacular* is now used for all kinds of *languages for mathematics*.

1.3. AUTOMATH-RELATED MATHEMATICAL VERNACULARS

De Bruijn intended Automath (quote from [6], cf. [14] p. 201 and 210) *not just [...] as a technical system for verification of mathematical texts, it was rather a life style with its attitudes towards understanding, developing and teaching mathematics*. He added: *The way mathematical material is to be presented to the system should correspond to the usual way we write mathematics. The only things to be added should be details that are usually omitted in standard mathematics*. This paper is based on de Bruijn's ideas.

- A direct source of inspiration for WTT was the mathematical language WOT (*Wiskundige Omgangstaal* or mathematical everyday language) [8] devised by de Bruijn. The description of WOT has always been quite general and descriptive. The language WTT is based on ideas employed in WOT, but it is independent and worked out into details.
- We also take ideas from de Bruijn's languages SEMIPAL and PAL [7] used to represent the *administrative structure* of mathematical texts. In both languages one can account for contexts, parameter lists and variables (in PAL also types). However, these languages are by nature insufficient to represent mathematical contents and they miss the expressivity necessary to comply with the reliability criterion.
- Similarly, we take ideas from the *mathematical language Automath* [7]. However, the latter requires mathematical content to be completely formalized in order to enable immediate *theorem checking*. By its amount of details, it is far too complex to obey the reliability criterion.

The greatest influence however is MV which accommodates de Bruijn's ideas for a language of mathematics. De Bruijn presented his MV in two rounds (we quote F3 of [14], p. 868): *In the first round we express the general framework of organization of mathematical texts. It is about books and lines, introduction of variables, assumptions, definitions, axioms and theorems [...]. In the second round we get the rules about validity*. His division in two rounds corresponds in WTT to our abstract syntax as round 1 and our derivation rules as round 2. But MV accommodates more logic than is found in mathematical texts and hence MV does not comply with the reliability criterion.

Similarly to adopting his rounds, we adopted in WTT de Bruijn's wish not to *take sets as the primitive vehicles for describing elementhood* (F3 of [14], Section 1.12). De Bruijn looked at (imaginary) substantives like *demisemitri-angle*, but also well-known ones as *point, number, function,...* He did not want to use sets only. In F3, Section 1.15, he says: *Our effort in describing a large part of the language in terms of both substantives and sets, instead of sets only, gives some duplication in the language rules that might be considered as superfluous. We of course would like to try to eliminate one of the two, and*

deal with sets only, or with substantives only. Both can be done, of course, but none of the two seems to give anything that looks more satisfactory than what we have in our MV. De Bruijn talks about *substantives, names and adjectives*. In WTT, we also use substantives (which we call *nouns*) and adjectives.

Similarly, we will adopt de Bruijn's notions of high- and low-typing. We will use the meta-typing (high-typing) in WTT itself (rather than in the meta-language) and will keep the usual low typing. An example of high-typing in MV is $A :: \text{substantive}$ whereas $a : A$ is a low-typing. These correspond in WTT to the high-typing $A :: \mathcal{N}$ and the low-typing $a : A$.

In MV, many logical and mathematical choices are made, which WTT still postpones. Moreover, MV incorporates certain correctness requirements, there is for example a hierarchy of types corresponding with sets and subsets. Therefore, MV is suited for a partial formalization of mathematical content, but it is already *on its way* to a full formalization, while WTT is not. Hence, WTT is *closer to* a given informal mathematical content than MV.³

De Bruijn said in F3 of [14] (p. 865): *It is quite conceivable that MV, or variations of it, can have an impact on computing science. A thing that comes at once into mind, is the use of MV as an intermediate language in expert systems. Another possible use might be formal or informal specification language for computer programs.* Our WTT may open the possibility for the mathematician to get computer help in the development of his ideas. E.g.:

- *Verification* of mathematical theories, e.g. by a type-theoretic computer program. This requires translating, in one or more steps, a WTT-text into type theory and the use of a typechecker. This translation is greatly technical and may be done by a type theory expert, not the mathematician who wrote the WTT-text. This leads to *separation of concerns* relieving the mathematician from the cumbersome task of filling the (possibly uninteresting) details. Another possibility is that a clever computer program does (part of) the translation, in interaction with either the mathematician who wrote the WTT-text or with a type theory expert.
- *Documentation* of bodies of checked mathematical texts in an archive or a database which is publicly accessible. One of the *views* for the inspection of such a database could be WTT.
- Computer assistance in the *development* of mathematics. A mathematician may use WTT as a *rough* formal language in which he expresses ideas and conjectures. A computer program translates the WTT-text into type theory, in communication with the mathematician, keeping track of all *holes* (open places in the reasoning) and *proof obligations*.

³ It is worth while to investigate whether MV (or a variant thereof) is suited as a next stage of intermediate language in the direction of a full formalization.

1.4. AN OVERVIEW OF THE PAPER

- In Section 2 we give the abstract syntax of WTT which describes the build-up of a book from its atoms (variables, constants, binders), via phrases (terms, sets, nouns and adjectives) and sentences (statements and definitions) up to (WTT-)contexts, (WTT-)lines and (WTT-)books.
- In Section 3 we give a derivation system for WTT following the syntax described in Section 2. The derivation system allows one to establish that a book written in WTT is well-formed following the syntax of WTT, and has great resemblance with ordinary mathematics books.
- In Section 4 we establish the metatheory of WTT and the properties of its derivation system. We show that weak type checking is decidable, enjoys subject reduction with respect to the unfolding of definitions in a book and that the unfolding of definitions is strongly normalising.
- In Section 5 we give a number of examples.
- In Section 6, we compare with other work and we conclude.

2. The abstract syntax for WTT

We present a syntax for Weak Type Theory, based on linguistic categories. Constants and binders (like Σ and \cup) are taken as *first-class citizens*. The categories include nouns and adjectives, which are not usually present in formalizations of mathematics (apart from Mizar [1, 17]). With a view to these categories, we introduce a number of binders, to facilitate linguistic constructions. Definitions play a prominent role in WTT and reflect the mathematicians’s habit to use the definition-mechanism. As in type theory, contexts are important in WTT, giving the immediate background for statements and definitions by listing their free variables together with their types. The notion of *line* expresses a statement or a definition together with its context. The final entity in the WTT-syntax is the *book*, being a sequence of lines. The book is the formal counterpart of a *mathematical text*.

2.1. LINGUISTIC CATEGORIES

In Weak Type Theory (or WTT) we have the following linguistic categories:

- On the *atomic* level: *variables*, *constants* and *binders*,
- On the *phrase*⁴ level: *terms* \mathcal{T} , *sets* \mathcal{S} , *nouns* \mathcal{N} and *adjectives* \mathcal{A} ,

⁴ According to the Concise Oxford Dictionary, a phrase is *a group of words forming a conceptual unit, but not a sentence*, a discourse is *a connected series of utterances*.

- On the *sentence* level: *statements* \mathcal{S} and *definitions* \mathcal{D} ,
- On the *discourse* level: *contexts* \mathbf{I} , *lines* \mathbf{l} and *books* \mathbf{B} .

There is a hierarchy between these levels: atoms are part of phrases; atoms and phrases are part of sentences; and discourses are built from sentences.

The syntax given below, establishes *well-formedness* conditions for these categories. We assume that the sets of variables, constants and binders are fixed, given beforehand, and mutually disjoint. For convenience, we suppress the word *well-formed* in the syntactic description of all categories.

2.2. ABSTRACT SYNTAX

We use abstract syntax for the description of the various syntactic categories. For example, in Section 2.11 we describe the collection of all books, \mathbf{B} , in abstract syntax as: $\mathbf{B} = \mathbf{0} \mid \mathbf{B} \circ \mathbf{l}$ to express that a book is either *the empty book* or a book \mathbf{B} followed by a line \mathbf{l} . By convention, $\mathbf{0} \circ \mathbf{l}$ is written as \mathbf{l} .

We make use of binders \mathbf{B} (e.g. \sum or \forall), in the abstract format $\mathbf{B}_Z(\mathcal{E})$, where the *subscript* Z is a *declaration* introducing a (bound) variable and its type, e.g. $x \in \mathbb{N}$ (see Section 2.7.1). Expressions \mathcal{E} are given in Section 2.5.

- $\sum_{x \in \{0,1,\dots,10\}}(x^2)$ and $\forall_{x \in \mathbb{N}}(x \geq 0)$ are examples of formulas with binders.
- The binding symbol for set comprehension, $\{\dots \mid \dots\}$, fits in this format after a slight modification. E.g., write $\{x \in \mathbb{R} \mid x > 5\}$ as $\mathbf{Set}_{x \in \mathbb{R}}(x > 5)$.
For uniformity, our standard for set notation will be the latter one.

Figures 1 and 2 give a list of the syntactic categories and their abstract syntax, followed by *metasymbols* for our various categories. We note:

- *Expressions* are used in Section 2.5 and represent the various categories that can follow a binder. An expression is a kind of *collective* category. So after a binder we may find a term, a set, a noun or a statement.
- *Parameters* (see Section 2.4) represent the categories on which constants may depend. Parameters have a collective character.
- *Typings* and *declarations* are special statements, see Section 2.7.1.

NOTATION 2.1. *In the abstract syntax, upper indices and lower indices play different roles. Upper indices are part of the symbol, but lower indices belong to the abstract syntax. For example, with $\mathbf{B}_Z^T(\mathcal{E})$, we mean all constructs composed of a binder in the set \mathbf{B}^T (e.g. \lim), subscripted with a declaration from Z (e.g. $n \in \mathbb{N}$) and followed by an expression in \mathcal{E} (e.g. $\frac{1}{n}$) between parentheses. The superscript T attached to \mathbf{B} says that the binders in \mathbf{B}^T are term-forming. Hence, $\lim_{n \in \mathbb{N}}(\frac{1}{n})$ is a term belonging to $\mathbf{B}_Z^T(\mathcal{E})$.*

level	Main category	abstract syntax	Meta-symbol
atomic	<i>variables</i>	$\mathbf{V} = \mathbf{V}^T \mathbf{V}^{\mathbb{S}} \mathbf{V}^{\mathcal{S}}$	x
	<i>constants</i>	$\mathbf{C} = \mathbf{C}^T \mathbf{C}^{\mathbb{S}} \mathbf{C}^{\mathcal{N}} \mathbf{C}^{\mathcal{A}} \mathbf{C}^{\mathcal{S}}$	c
	<i>binders</i>	$\mathbf{B} = \mathbf{B}^T \mathbf{B}^{\mathbb{S}} \mathbf{B}^{\mathcal{N}} \mathbf{B}^{\mathcal{A}} \mathbf{B}^{\mathcal{S}}$	b
phrase	<i>terms</i>	$\mathcal{T} = \mathbf{C}^T(\vec{\mathcal{P}}) \mathbf{B}_Z^T(\mathcal{E}) \mathbf{V}^T$	t
	<i>sets</i>	$\mathbb{S} = \mathbf{C}^{\mathbb{S}}(\vec{\mathcal{P}}) \mathbf{B}_Z^{\mathbb{S}}(\mathcal{E}) \mathbf{V}^{\mathbb{S}}$	s
	<i>nouns</i>	$\mathcal{N} = \mathbf{C}^{\mathcal{N}}(\vec{\mathcal{P}}) \mathbf{B}_Z^{\mathcal{N}}(\mathcal{E}) \mathcal{A}\mathcal{N}$	n
	<i>adjectives</i>	$\mathcal{A} = \mathbf{C}^{\mathcal{A}}(\vec{\mathcal{P}}) \mathbf{B}_Z^{\mathcal{A}}(\mathcal{E})$	a
sentence	<i>statements</i>	$\mathcal{S} = \mathbf{C}^{\mathcal{S}}(\vec{\mathcal{P}}) \mathbf{B}_Z^{\mathcal{S}}(\mathcal{E}) \mathbf{V}^{\mathcal{S}}$	S
	<i>definitions</i>	$\mathcal{D} = \mathcal{D}^{\emptyset} \mathcal{D}^{\mathcal{S}}$ $\mathcal{D}^{\emptyset} = \mathbf{C}^T(\vec{\mathbf{V}}) := \mathcal{T} \mathbf{C}^{\mathbb{S}}(\vec{\mathbf{V}}) := \mathbb{S} $ $\mathbf{C}^{\mathcal{N}}(\vec{\mathbf{V}}) := \mathcal{N} \mathbf{C}^{\mathcal{A}}(\vec{\mathbf{V}}) := \mathcal{A}$ $\mathcal{D}^{\mathcal{S}} = \mathbf{C}^{\mathcal{S}}(\vec{\mathbf{V}}) := \mathcal{S}$	D
discourse	<i>contexts</i>	$\mathbf{\Gamma} = \emptyset \mathbf{\Gamma}, \mathcal{Z} \mathbf{\Gamma}, \mathcal{S}$	Γ
	<i>lines</i>	$\mathbf{l} = \mathbf{\Gamma} \triangleright \mathcal{S} \mathbf{\Gamma} \triangleright \mathcal{D}$	l
	<i>books</i>	$\mathbf{B} = \emptyset \mathbf{B} \circ \mathbf{l}$	B

Figure 1. Main categories of syntax

Other category	abstract syntax	Meta-symbol
<i>expressions</i>	$\mathcal{E} = \mathcal{T} \mathbb{S} \mathcal{N} \mathcal{S}$	E
<i>parameters</i>	$\mathcal{P} = \mathcal{T} \mathbb{S} \mathcal{S}$ (note: $\vec{\mathcal{P}}$ is a list of \mathcal{P} s)	P
<i>typings</i>	$\mathbf{T} = \mathbb{S} : \text{SET} \mathcal{S} : \text{STAT} \mathcal{T} : \mathbb{S} \mathcal{T} : \mathcal{N} \mathcal{T} : \mathcal{A}$	T
<i>declarations</i>	$\mathcal{Z} = \mathbf{V}^{\mathbb{S}} : \text{SET} \mathbf{V}^{\mathcal{S}} : \text{STAT} \mathbf{V}^T : \mathbb{S} \mathbf{V}^T : \mathcal{N}$	Z

Figure 2. Categories of syntax

2.3. VARIABLES

The set $V = V^T | V^S | V^S$ is fixed, infinite, and divided into three disjoint subsets:

(V^T) Variables ranging over *terms*,

(V^S) Variables ranging over *sets*,

(V^S) Variables ranging over *statements*.

2.4. CONSTANTS

Constants play an important role in mathematical language. They are either *primitive*⁵ or they act as an abbreviation. In the latter case a constant is introduced in the left hand side of a *definition*, being a special kind of sentence (see Section 2.8). Both primitive and defined constants can be *used* after having been introduced. *Doing mathematics without* constants (hence without definitions) is theoretically possible but practically unfeasible [14].

The set $C = C^T | C^S | C^N | C^A | C^S$ is fixed, infinite and is disjoint from the set of variables. C is divided into the following five disjoint subsets:

(C^T) Constants for *terms*,

(C^S) Constants for *sets*,

(C^N) Constants for *nouns*,

(C^A) Constants for *adjectives*,

(C^S) Constants for *statements*

A constant is always followed by a *parameter list*. We denote this as $C(\vec{P})$. This list has for each constant a fixed length ≥ 0 , the *arity* of the constant. Parameters P are either terms, sets or statements: $P = T | S | S$.⁶

(If the parameter list is empty we write c instead of $c()$.)

REMARK 2.2. *We often use sugared versions of the combination constant followed by parameter list. For example, instead of the centre (C) we write the centre of C , and instead of $+(3, 6)$ we write the infix formula $3 + 6$.*

We also often write things like $x \in \mathbb{N}$ instead of $x : \mathbb{N}$. In doing this, we confuse is element of with has type. Again, this is for easy understanding.

These sugarings are not part of the syntax of WTT and should be undone whenever formal accuracy is at stake. We do not incorporate sugaring formally in WTT, since we want to keep WTT simple: it is not easy to decide where sugar is useful and adding rules for sugaring introduces arbitrariness. We are aware that this policy of ours undermines our claim that WTT is as close as possible to CML. We leave this dilemma to further research.

⁵ *Primitive* constants are introduced axiomatically, they are not defined in terms of other notions. E.g., the primitive set \mathbb{N} of the natural numbers, the primitive function s (*successor*) from \mathbb{N} to \mathbb{N} or the primitive element 0 in \mathbb{N} .

⁶ We use a list format for the parameters (un-Curried), because this is usual in Automath-like systems and also because this is how mathematicians use parameters.

EXAMPLE 2.3. *For each kind of constants, we give examples of constants with parameter lists and then state what the constants resp. the parameters are.*

(\mathcal{C}^T) *Constants for terms with parameter lists:*

π , the centre of C , $3 + 6$, the arithmetic mean of 3 and 6, $d(x, y)$, ∇f .

The constants are: π , the centre, $+$, the arithmetic mean, d and ∇ .

The parameter lists are: $()$, (C) , $(3, 6)$, $(3, 6)$, (x, y) and (f) , resp.

(\mathcal{C}^S) *Constants for sets with parameter lists:* \mathbb{N} , A^c , $V \rightarrow W$, $A \cup B$.

(where A^c is the complement of A). *The constants are:* \mathbb{N} , c , \rightarrow , \cup .

The parameter lists are: $()$, (A) , (V, W) , (A, B) .

(\mathcal{C}^N) *Constants for nouns with parameter lists:* a triangle,

an eigenvalue of A , an edge of $\triangle ABC$, a reflection of V with respect to l .

The constants are: a triangle, an eigenvalue, an edge, a reflection.

The parameter lists are: $()$, (A) , $(\triangle ABC)$, (V, l) .

(\mathcal{C}^A) *Constants for adjectives with parameter lists:* prime, surjective,

Abelian, continuous on $[a, b]$.

The constants are: prime, surjective, Abelian, continuous.

The parameter lists are: $()$, $()$, $()$, $([a, b])$.

(\mathcal{C}^S) *Constants for statements with parameter lists:*

P lies between Q and R , $5 \geq 3$, $p \wedge q$, $\neg \forall_{x \in \mathbb{N}}(x > 0)$.

The constants are: lies between, \geq , \wedge , \neg .

The parameter lists are: (P, Q, R) , $(5, 3)$, (p, q) , $(\forall_{x \in \mathbb{N}}(x > 0))$.⁷

2.4.1. Special constants

We introduce two special constants in order to switch between the two categories *noun* and *set*. These categories are both present and frequently used in CML and it turns out to be useful to be able to easily change from the one to the other. Note that nouns and sets are in a sense interchangeable and one could restrict oneself to only one of these categories, without losing expressive power (as is actually done in the set-theoretic formalization).

The first constant is \uparrow , of category \mathcal{C}^S . The second one is \downarrow , of category \mathcal{C}^N . They have complementary roles. The unary constant \uparrow *lifts* a noun to the corresponding set, \downarrow does the opposite. Here are examples of these constants:

⁷ Note that the parameters in parameter lists are either *terms* or *sets*. Only in the case of statements the parameters may be *statements* as well, as is shown in the last two examples.

$(\mathcal{C}^{\mathbb{S}})$ (a natural number) $\uparrow = \mathbb{N}$, (a divisor of 4) $\uparrow = \{1, 2, 4\}$,⁸
 $(\mathbf{Noun}_{x \in \mathbb{R}}(x > 5))\uparrow = \mathbf{Set}_{x \in \mathbb{R}}(x > 5)$.

$(\mathcal{C}^{\mathcal{N}})$ $\mathbb{Z}\downarrow$ is an integer, $(\mathbf{Set}_{x \in \mathbb{R}^2}(|x| = 1))\downarrow$ is $\mathbf{Noun}_{x \in \mathbb{R}^2}(|x| = 1)$ or
 a point on the unit circle.

2.5. BINDERS

As a third set given beforehand and infinite, we have the set of *binders*. This set is disjoint from both the set of variables and the set of constants. We divide the set \mathbf{B} of binders into five subcategories, depending on the resulting category of the bound expression $\mathbf{B}_Z(\mathcal{E})$ in which the binder occurs (recall that Z is a declaration, e.g. $x : \mathbb{N}$). Hence, $\mathbf{B} = \mathbf{B}^{\mathcal{T}} | \mathbf{B}^{\mathbb{S}} | \mathbf{B}^{\mathcal{N}} | \mathbf{B}^{\mathcal{A}} | \mathbf{B}^{\mathcal{S}}$ where:

$(\mathbf{B}^{\mathcal{T}})$ Binders giving *terms*, $(\mathbf{B}^{\mathbb{S}})$ Binders giving *sets*,
 $(\mathbf{B}^{\mathcal{A}})$ Binders giving *adjectives*, $(\mathbf{B}^{\mathcal{S}})$ Binders giving *statements*,
 $(\mathbf{B}^{\mathcal{N}})$ Binders giving *nouns*,

In $\mathbf{B}_Z(\mathcal{E})$, the body \mathcal{E} is one of four categories $\mathcal{E} = \mathcal{T} | \mathbb{S} | \mathcal{N} | \mathcal{S}$. The next examples list bound expressions according to the category of the binder:

- $\mathbf{B}_Z^{\mathcal{T}}(\mathcal{E}) = \min_Z(\mathcal{T}) | \Sigma_Z(\mathcal{T}) | \lim_Z(\mathcal{T}) | \int_Z(\mathcal{T}) | \lambda_Z(\mathcal{T}) | \lambda_Z(\mathbb{S}) | \iota_Z(\mathcal{S}) | \dots$
- $\mathbf{B}_Z^{\mathbb{S}}(\mathcal{E}) = \mathbf{Set}_Z(\mathcal{S}) | \bigcup_Z(\mathbb{S}) | \iota_Z(\mathcal{S}) | \dots$
- $\mathbf{B}_Z^{\mathcal{N}}(\mathcal{E}) = \mathbf{Noun}_Z(\mathcal{S}) | \mathbf{Abst}_Z(\mathcal{T}) | \mathbf{Abst}_Z(\mathbb{S}) | \mathbf{Abst}_Z(\mathcal{N}) | \dots$
- $\mathbf{B}_Z^{\mathcal{A}}(\mathcal{E}) = \mathbf{Adj}_Z(\mathcal{S}) | \dots$
- $\mathbf{B}_Z^{\mathcal{S}}(\mathcal{E}) = \forall_Z(\mathcal{S}) | \dots$

Some of these binders are given in what follows (for \mathbf{Set} , see Section 2.2).

2.5.1. The λ -binder

The format of an expression bound by Church's λ -binder is: $\lambda_Z(\mathcal{T}/\mathbb{S})$. Here $\lambda_Z(\mathcal{T})$ is a term-valued function and $\lambda_Z(\mathbb{S})$ is a set-valued function. Examples:

$(\mathcal{E} \equiv \mathcal{T})$ The term $\lambda_{x \in \mathbb{R}}(x^2)$ denotes the squaring function on the reals.

$(\mathcal{E} \equiv \mathbb{S})$ The term $\lambda_{n \in \mathbb{N}} \mathbf{Set}_{k \in \mathbb{N}}(k \leq n)$ sends a natural number n to the set $\{0, 1, \dots, n\}$.

⁸ Here again, we used sugaring. We write, $\{1, 2, 4\}$ for $\mathbf{Set}_{n \in \mathbb{N}}(n = 1 \vee n = 2 \vee n = 4)$. However, the notation with \mathbf{Set} is the only *official* WTT-format.

2.5.2. The ι -binder

Russell's ι is used for a *definite description*: *the* such and such, such that The general format for an expression bound with the ι -binder is: $\iota_Z(S)$. The result of the binding of a sentence by means of ι can either be a term or a set (therefore we find $\iota_Z(S)$ both in the B^T - and in the B^S -list). For example:

- The term $\iota_{n \in \mathbb{N}}(2 < n < \pi)$ describes natural number 3.
- The set $\iota_U: \text{SET}(3 \in U \wedge |U| = 1)$ describes the singleton set $\{3\}$ (or $\text{Set}_{n \in \mathbb{N}}(n = 3)$ in unsugared format). (The declaration $U : \text{SET}$ expresses that U is a set. See also Section 2.7.1.)

2.5.3. The **Noun**-binder

Since nouns (indefinite noun phrases) are first-class citizens in WTT, they are treated similarly to sets. Consequently, next to set comprehension, we allow *noun comprehension*, i.e. the construction of a noun. For noun comprehension we introduce the binder **Noun**. It is used for an *indefinite description*: *a* such and such, such that Hence, the general format of a phrase with **Noun**-binder is: $\text{Noun}_Z(S)$, i.e. *a noun saying of Z that S*. Examples include:

- The noun $\text{Noun}_{x \in \mathbb{R}}(5 < x < 10)$ is *a real number between 5 and 10*.
- $\text{Noun}_V: \text{SET}(|V| = 2)$ is *a set with two elements*.

2.5.4. The **Abst**-binder

The **Abst**-binder *abstracts* from a term \mathcal{T} , a set \mathbb{S} or a noun \mathcal{N} and delivers a noun. It is the formal counterpart of the modifier *for some* One may read $\text{Abst}_Z(\mathcal{T}/\mathbb{S}/\mathcal{N})$ as *a term \mathcal{T} , or a set \mathbb{S} , or a noun \mathcal{N} , for some Z*.

Here are examples of the three kinds of nouns $\text{Abst}_Z(\mathcal{T}/\mathbb{S}/\mathcal{N})$:

- $(\mathcal{E} \equiv \mathcal{T})$ $\text{Abst}_{n \in \mathbb{N}}(n^2)$ represents *a term n^2 for some natural number n , i.e. the square of some natural number*.
- $(\mathcal{E} \equiv \mathbb{S})$ $\text{Abst}_{n \in \mathbb{N}} \text{Set}_{x \in \mathbb{R}}(x > n)$ represents *a set $\{x \in \mathbb{R} \mid x > n\}$ for some natural number n , i.e. an interval of the form (n, ∞) , with $n \in \mathbb{N}$* .
- $(\mathcal{E} \equiv \mathcal{N})$ $\text{Abst}_{n \in \mathbb{N}} \text{Noun}_{x \in \mathbb{R}}(10n \leq x < 10n + 1)$ represents *a real number in the interval $[10n, 10n + 1)$ for some n , i.e. a non-negative real number which, written in decimal notation, has a zero at the position just before the decimal point*.

REMARK 2.4.

1. The **Abst**-binder is useful and compact. It enables one to put the abstraction quantification on the outside of the expression. However, a noun constructed with the **Abst**-binder can always be rewritten without it. We show this by rewriting the examples in a form without **Abst**, viz.:

$$\begin{aligned} (\mathcal{E} \equiv \mathcal{T}) \text{ Abst}_{n \in \mathbb{N}}(n^2) &\sim \text{Noun}_{k \in \mathbb{N}} \exists_{n \in \mathbb{N}}(k = n^2), \\ (\mathcal{E} \equiv \mathcal{S}) \text{ Abst}_{n \in \mathbb{N}} \text{Set}_{x \in \mathbb{R}}(x > n) &\sim \text{Noun}_{V: \text{SET}} \exists_{n \in \mathbb{N}}(V = \text{Set}_{x \in \mathbb{R}}(x > n)), \\ (\mathcal{E} \equiv \mathcal{N}) \text{ Abst}_{n \in \mathbb{N}} \text{Noun}_{x \in \mathbb{R}}(10n \leq x < 10n + 1) \\ &\sim \text{Noun}_{x \in \mathbb{R}} \exists_{n \in \mathbb{N}}(10n \leq x < 10n + 1). \end{aligned}$$

In all these examples, an inside \exists takes the role of the outside **Abst**.

2. In the third case, the **Abst**-binder, transforming a noun into a noun, corresponds to the \cup -binder, transforming a set into a set. This can be expressed by the abstract transformation: $(\text{Abst}_Z(\mathcal{N}))\uparrow = \cup_Z((\mathcal{N})\uparrow)$. See the third example: $\cup_{n \in \mathbb{N}} \text{Set}_{x \in \mathbb{R}}(10n \leq x < 10n + 1)$ is the set of all real numbers in some interval $[10n, 10n + 1)$. This set can also be written without \cup :

$$\cup_{n \in \mathbb{N}} \text{Set}_{x \in \mathbb{R}}(10n \leq x < 10n + 1) \sim \text{Set}_{x \in \mathbb{R}} \exists_{n \in \mathbb{N}}(10n \leq x < 10n + 1).$$

Note again the inside \exists , this time in the place of the outside \cup .

2.5.5. The **Adj**-binder

Adjectives, being first-class citizens as well, can be constructed with the **Adj**-binder. One can read $\text{Adj}_Z(S)$ as: *the adjective saying of Z that S*. E.g.:

$\text{Adj}_{n \in \mathbb{N}}(\exists_{k \in \mathbb{N}}(n = k^2 + 1))$ is an adjective saying of a natural number that it is a square plus 1. One could give this adjective a name, say *oversquare* and hence say things like *5 is oversquare* or *Let m be an oversquare number*.

2.6. PHRASES

Phrases can be terms, sets, nouns or adjectives:

$$\begin{aligned} \mathcal{T} &= \mathcal{C}^{\mathcal{T}}(\vec{\mathcal{P}}) | \mathcal{B}_Z^{\mathcal{T}}(\mathcal{E}) | \mathcal{V}^{\mathcal{T}} & \mathcal{S} &= \mathcal{C}^{\mathcal{S}}(\vec{\mathcal{P}}) | \mathcal{B}_Z^{\mathcal{S}}(\mathcal{E}) | \mathcal{V}^{\mathcal{S}} \\ \mathcal{N} &= \mathcal{C}^{\mathcal{N}}(\vec{\mathcal{P}}) | \mathcal{B}_Z^{\mathcal{N}}(\mathcal{E}) | \mathcal{A}\mathcal{N} & \mathcal{A} &= \mathcal{C}^{\mathcal{A}}(\vec{\mathcal{P}}) | \mathcal{B}_Z^{\mathcal{A}}(\mathcal{E}). \end{aligned}$$

Examples of $\mathcal{C}^{\mathcal{T}}(\vec{\mathcal{P}})$, $\mathcal{C}^{\mathcal{S}}(\vec{\mathcal{P}})$, $\mathcal{C}^{\mathcal{N}}(\vec{\mathcal{P}})$ and $\mathcal{C}^{\mathcal{A}}(\vec{\mathcal{P}})$ were given in Example 2.3, examples of $\mathcal{B}_Z^{\mathcal{T}}(\mathcal{E})$, $\mathcal{B}_Z^{\mathcal{S}}(\mathcal{E})$, $\mathcal{B}_Z^{\mathcal{N}}(\mathcal{E})$ and $\mathcal{B}_Z^{\mathcal{A}}(\mathcal{E})$ in Section 2.5.

The combination $\mathcal{A}\mathcal{N}$ gives a (new) noun which is a combination of an adjective and a noun. E.g.: *isosceles triangle*, *convergent series*.

Note that, as in Section 2.3, variables ranging over nouns or adjectives are missing in this scheme: such variables are not required in either CML or WTT.

2.7. STATEMENTS

Abstract syntax for the category of *statements* is: $\mathcal{S} = \mathcal{C}^{\mathcal{S}}(\vec{\mathcal{P}}) | \mathbb{B}_{\mathcal{Z}}^{\mathcal{S}}(\mathcal{E}) | \mathbb{V}^{\mathcal{S}}$.

Examples of $\mathcal{C}^{\mathcal{S}}(\vec{\mathcal{P}})$ were given in Example 2.3. An example of $\mathbb{B}_{\mathcal{Z}}^{\mathcal{S}}(\mathcal{E})$ (with the \forall -binder for $\mathbb{B}^{\mathcal{S}}$) was given in Section 2.5.

2.7.1. Typings and declarations

A *typing statement* or *typing*, expresses the relation between something and its type. In WTT we have five kinds of typings, depending on the nature of the type. This type can be: SET (the type of all sets), STAT (the type of all statements), a set, a noun or an adjective. Each of these typing statements relates a *subject* (the left hand side) with its *type/predicate* (the right hand side). The abstract syntax for the set \mathbf{T} of typing statements ($\mathbf{T} \subseteq \mathcal{S}$) is: $\mathbf{T} = \mathbb{S} : \text{SET} | \mathcal{S} : \text{STAT} | \mathcal{T} : \mathbb{S} | \mathcal{T} : \mathcal{N} | \mathcal{T} : \mathcal{A}$. Here, $s : \text{SET}$, $S : \text{STAT}$, $t : s$, $t : n$ and $t : a$ stand for s is a set, S is a statement, t is an element of s , t is n and t is a . Examples of these cases include: $\text{Set}_{n \in \mathbb{N}}(n \leq 2) : \text{SET}$, $p \wedge q : \text{STAT}$, $3 \in \mathbb{N}$,⁹ $AB : \text{an edge of } \triangle ABC$, $\lambda_{x \in \mathbb{R}}(x^2) : \text{differentiable}$.

Clearly, \mathbf{T} is a subcollection of $\mathcal{C}^{\mathcal{S}}(\vec{\mathcal{P}})$, the set of relational statements, with symbol “:” as special element in $\mathcal{C}^{\mathcal{S}}$. See also Section 3.7.

At its turn, a subcollection of the typings is formed by the *declarations*, \mathcal{Z} , where the subject is a variable:¹⁰ $\mathcal{Z} = \mathbb{V}^{\mathcal{S}} : \text{SET} | \mathbb{V}^{\mathcal{S}} : \text{STAT} | \mathbb{V}^{\mathcal{T}} : \mathbb{S} | \mathbb{V}^{\mathcal{T}} : \mathcal{N}$.

Here the variable $\mathbb{V}^{\mathcal{S}}$, $\mathbb{V}^{\mathcal{S}}$ or $\mathbb{V}^{\mathcal{T}}$ is the *introduced* or *declared* variable.

Subscripts of binders (Section 2.5) can be taken from \mathcal{Z} .

2.8. DEFINITIONS

The category $\mathcal{D} = \mathcal{D}^{\mathcal{P}} | \mathcal{D}^{\mathcal{S}}$ of *definitions* introduces new constants. We distinguish between *phrase definitions* $\mathcal{D}^{\mathcal{P}}$ and *statement definitions* $\mathcal{D}^{\mathcal{S}}$. Phrase definitions fix a constant representing a phrase. Statement definitions introduce a constant embedded in a statement. In definitions, the defined constant is separated from the phrase or statement it represents by the symbol “:=”.

REMARK 2.5. *We have decided not to include definitions for binders. However, it is not very hard to include binder definitions in the syntax using an abstract syntax like: $\mathcal{D}^{\mathcal{B}} = \mathbb{B}_{\mathcal{Z}}^{\mathcal{T}/\mathbb{S}/\mathcal{N}/\mathcal{A}/\mathcal{S}}(\mathcal{E}) := \mathcal{T}/\mathbb{S}/\mathcal{N}/\mathcal{A}/\mathcal{S}$.*

⁹ As this example shows, we often replace $t : s$ by $t \in s$, with abuse of notation.

¹⁰ There are no declarations with an adjective as type. This seems strange, at first sight, since it is usual to write things like: *Let f be differentiable*. Such a sentence however, is used either: (1) as an elliptic version of the introduction of a new variable with a *noun* (not an adjective) as type: *Let f be a differentiable function*, or (2) as an *assumption* about an f which is already known (not new), and hence as a typing statement, *not* a declaration.

2.8.1. *Phrase definitions*

We take $\mathcal{D}^\circ = \mathcal{C}^T(\vec{V}) := \mathcal{T} \mid \mathcal{C}^S(\vec{V}) := \mathcal{S} \mid \mathcal{C}^N(\vec{V}) := \mathcal{N} \mid \mathcal{C}^A(\vec{V}) := \mathcal{A}$

The newly *defined constants* are \mathcal{C}^T , \mathcal{C}^S , \mathcal{C}^N or \mathcal{C}^A , respectively.

Note that the parameters occurring after the \mathcal{C} in the left hand side of each definition must be *variables*. The reason is of course, that a definition should be as general as possible and hence may *depend* on a list of variables. Later, when *using* the definition in a certain situation, all these variables must be *instantiated* according to that situation. Examples of phrase definitions are:

$(\mathcal{C} \equiv \mathcal{C}^T)$ *the arithmetic mean of a and b* $:= \mathbf{t}_{z \in \mathbb{R}}(z = \frac{1}{2}(a + b))$,

$(\mathcal{C} \equiv \mathcal{C}^S)$ $\mathbb{R}^+ := \mathbf{Set}_{x \in \mathbb{R}}(x > 0)$,

$(\mathcal{C} \equiv \mathcal{C}^N)$ *a unit of G with respect to \cdot* $:= \mathbf{Noun}_{e \in G}(\forall a \in G(a \cdot e = e \cdot a = a))$ ¹¹,

$(\mathcal{C} \equiv \mathcal{C}^A)$ *prime* $:= \mathbf{Adj}_{n \in \mathbb{N}}(n > 1 \wedge \forall k, l \in \mathbb{N}(n = k \cdot l \Rightarrow k = 1 \vee l = 1))$.¹²

The variable lists in the four examples are: (a, b) , $()$, (G, \cdot) , $()$. These variables must be introduced (*declared*) in a context (see Section 2.9). For the first definition, such a context can be e.g. $a : \mathbb{R}, b : \mathbb{R}$. For the third definition the context is: $G : \mathbf{SET}, \cdot : G \rightarrow G$. Both contexts consist of *declarations* only.

However, definitions may also depend on *assumptions*. This is reflected in Section 2.9, where it is stated that a context consists of a list of declarations *and* assumptions. For an example, take the definition of the natural logarithm (this is again case $\mathcal{C} \equiv \mathcal{C}^T$): $\mathbf{In}(x) := \mathbf{t}_{y \in \mathbb{R}}(e^y = x)$.

Here variable x has to be declared in a context, e.g.: $x : \mathbb{R}, x > 0$ which declares x of type \mathbb{R} and assumes x is positive.

In general, definitions are not complete without such a context. That is to say, the *ground has to be prepared* before the actual definition is stated.

In Section 3 we see that in *weakly well-typed* definitions, the variables in variable list \vec{V} of a definition are the same as the declared variables in the context, and listed in the same order. (The *assumptions* occurring in the context are not accounted for in the parameter list of a WTT-constant.)¹³ E.g., the parameter list of *the arithmetic mean* is (a, b) , which is the same as the list of the declared variables occurring in the context $a \in \mathbb{R}, b \in \mathbb{R}$.¹⁴

An *instantiation* of a defined notion is the *use* of a defined constant, thereby replacing the variables occurring in the variable list, by actual terms or sets.

Examples of instantiations of the first example definition are: *the arithmetic mean of 3 and 6*, or, for given x : *the arithmetic mean of x and x^2* .

¹¹ Of course, $a \cdot e = e \cdot a = a$ is a sugared version of e.g. $a \cdot e = a \wedge e \cdot a = a$.

¹² Here obviously $\forall_{k, l \in \mathbb{N}} \dots$ acts as a syntactic sugaring of $\forall_{k \in \mathbb{N}} \forall_{l \in \mathbb{N}} \dots$

¹³ In type theory, however, variables *inhabiting* assumptions are added to the variable list.

¹⁴ Since such a parameter list can be reconstructed from the context in which the definition is embedded, these parameter lists (or parts of it) are often omitted, just as in Automath.

2.8.2. Statement definitions

$\mathcal{D}^S = \mathcal{C}^S(\vec{V}) := \mathcal{S}$ is the category of statement definitions *defining constant* \mathcal{C}^S . For example (note again that to make the definition self-contained, we need a context like: a : a line, b : a line (i.e., *Let a and b be lines*)):

$(\mathcal{C} \equiv \mathcal{C}^S)$ a is parallel to $b := \neg \exists p. \text{ a point}(P \text{ lies on } a \wedge P \text{ lies on } b)$.

The notion *is parallel to* can only be considered as a two-place *relation*, and hence its definition must be a statement definition. In other cases, things are not so clear and the WTT-user has to make a choice. For example, the definition of x is the opposite of y can be treated as a statement definition:

(1) x is the opposite of $y := x + y = 0$,

with context consisting of e.g. $x : \mathbb{R}, y : \mathbb{R}$. Here we have the case $\mathcal{C} \equiv \mathcal{C}^S$.

But it is possible to define the same notion in a *phrase* (term) definition:

(2) *the opposite of y* := $\iota_{x \in \mathbb{R}}(x = -y)$.

Now the context is only $y : \mathbb{R}$ and we have the case $\mathcal{C} \equiv \mathcal{C}^T$.

The difference is whether one considers (*is*) *the opposite of* to be a *relation* or a *constant*. The latter choice allows more freedom, since a phrase definition can be used as part of a sentence, but not the other way round.

E.g., if one chooses for (2), then it is possible to instantiate this definition:

- in a phrase: *the opposite of 5*,
- but also in a sentence: $-5 = \textit{the opposite of } 5$.

Likewise, it is more flexible to define the phrase (in this case: the noun) *a unit of* (G, \cdot) than to define the statement e is a unit of (G, \cdot) .

2.9. CONTEXTS

A *context* Γ is a list of declarations \mathcal{Z} and statements \mathcal{S} : $\mathbf{\Gamma} = \emptyset \mid \mathbf{\Gamma}, \mathcal{Z} \mid \mathbf{\Gamma}, \mathcal{S}$. A declaration in a context represents the *introduction of a variable* of a known type. A statement in a context stands for an *assumption*.¹⁵ For example we give the context $x : \mathbb{R}, x > 0$, as well as the context a : a line, b : a line.

2.10. LINES

A *line* l contains either a statement or a definition, relative to a context: $\mathbf{l} = \mathbf{\Gamma} \triangleright \mathcal{S} \mid \mathbf{\Gamma} \triangleright \mathcal{D}$. The symbol \triangleright is a separation marker between the context and the statement or definition. Here are two examples of lines:

A statement line: $x : \mathbb{N}, y : \mathbb{N}, x < y \triangleright x^2 < y^2$,

¹⁵ According to our syntax, such an assumption can also be a declaration – being a statement with *a variable* as subject. However, the typing rules for contexts given in Section 3, will ensure that there is no *newly* introduced variable in an assumption, so it is always clear whether a context statement represents the introduction of a new variable, or an assumption.

A definition line: $x : \mathbb{R}, x > 0 \triangleright \ln(x) := \iota_{y \in \mathbb{R}}(e^y = x)$.

2.11. BOOKS

A *book* B is a list of lines: $\mathbf{B} = \emptyset \mid \mathbf{B} \circ \mathbf{l}$.

A simple example of a book consisting of two lines is the following:

$x : \mathbb{R}, x > 0 \triangleright \ln(x) := \iota_{y \in \mathbb{R}}(e^y = x)$ \circ
 $\emptyset \triangleright \ln(e^3) = 3$.

3. A derivation system for WTT

A WTT-book constructed with our derivation system must obey the syntax given in the previous section, and is hence well-formed. However, the derivation rules only give a *subset* of the well-formed constructs obtained with the abstract syntax of Section 2, since the rules enforce that those constructs obey certain (weak) *typing requirements*. Constructs obtained by repeated application of the derivation rules, are hence called *weakly well-typed*. The overall properties of weakly well-typed constructs are summarized by:

- All constructs obtained with the derivation system have a weak type, which corresponds with a linguistic category.
- The derivation system is *syntax-driven* in the sense that for each (sub)goal in a derivation, only one rule is applicable.

A book which has been constructed with our derivation system is transparently structured. Yet, it has a great resemblance with an ordinary mathematical text: see the examples in Section 5. Therefore, a weakly well-typed book can be seen as a natural formalization of mathematics, highlighting a number of characteristic features of mathematical texts. Among these features are:

- A WTT-book reflects the line-for-line development of the original mathematical text, in the order of the lines which form the WTT-book.
- The important role of contexts, both for (mathematical) statements and for (mathematical) definitions, is made explicit in a WTT-book .
- There is an important role for defined constants with parameter lists.
- Binders of different sorts are incorporated.

One can also consider a WTT-text to be a first step towards a complete formalization into type theory. Type theory is a natural formalism for such a formalization as is shown by many examples: think e.g. of Automath [14],

being defined in the late sixties and applied to express numerous mathematical subject matters. Other examples include theorem provers like Coq ([2]) developed in the nineties and firmly embedded into type theory.

However, note that the type restrictions of WTT are only weak. Consequently, a successful formalization of a mathematical text into WTT does not at all guarantee that its mathematical content is in any sense *meaningful*. A prerequisite for the construction of a sensible WTT-text is that the person writing this text has a mathematical subject in mind, which he chooses to express in this formalism, as faithfully as possible. But only a further translation into type theory will give a complete picture, which still has to be *authorized* by the original text writer: *This is (or is not) what I had in mind*.

In this section, we discuss weak types and the *preface* of a book B , meant to establish weak type information about all constants occurring in B but not explicitly defined in B . Then we follow the construction of the various weak types bottom up, assuming that we have already a weakly well-typed book B and a weakly well-typed context Γ relative to B . We start with variables and constants relative to B and Γ . Next, we discuss the construction of phrases beginning with a binder and of phrases in general. Finally, we give derivation rules for sentences (statements and definitions) and for contexts and books.

3.1. WEAK TYPES

Our derivation rules enable one to extend a weakly well-typed book B with a line l , in order to form a new weakly well-typed book $B \circ l$. This is the case if the added line l obeys certain weak well-typedness requirements itself, relative to the book B . Since a line l always has the form $\Gamma \triangleright S$ or $\Gamma \triangleright D$, with statement S or definition D (see Section 2.10), we also have to consider weak well-typedness of a context Γ *relative to a book B* , and weak well-typedness of a statement S or a definition D *relative to a book B and a context Γ* .

For establishing weak well-typedness, we need a notion of *weak typing* between an entity and its weak type. This relation is denoted by a bold-faced double colon ($::$). As weak types, denoted by \mathbf{W} , we use an eight element subset of our linguistic categories of Section 2.1: $\mathbf{W} = \mathbf{B}, \mathbf{I}, \mathcal{T}, \mathcal{S}, \mathcal{N}, \mathcal{A}, \mathcal{S}, \mathcal{D}$. These stand for *books, contexts, terms, sets, nouns, adjectives, statements and definitions*.

E.g., $B :: \mathbf{B}$ expresses that the weak type of B is \mathbf{B} (or: B is a weakly well-typed book) and $n :: \mathcal{N}$ expresses that the weak type of n is \mathcal{N} .

We also introduce a notion \vdash of (*relative*) *derivability*. We distinguish between three formats for derivability, in the form of so-called *judgements*:

- (1) B is a weakly well-typed book: $\vdash B :: \mathbf{B}$.
- (2) Γ is a weakly well-typed context relative to book B : $B \vdash \Gamma :: \mathbf{I}$.
- (3) t is a weakly well-typed term, etc., relative to book B and context Γ :

$$\begin{array}{lll}
B; \Gamma \vdash t :: \mathcal{T}, & B; \Gamma \vdash s :: \mathcal{S}, & B; \Gamma \vdash n :: \mathcal{N}, \\
B; \Gamma \vdash a :: \mathcal{A}, & B; \Gamma \vdash S :: \mathcal{S}, & B; \Gamma \vdash D :: \mathcal{D}
\end{array}$$

NOTATION 3.1. We abbreviate $\vdash B :: \mathbf{B}$, $B \vdash \Gamma :: \mathbf{I}$, by: $OK(B; \Gamma)$.

3.2. THE PREFACE

DEFINITION 3.2.

- We say that $l \in B$ if l is one of the lines constituting B . If line l is a definition, it contains exactly one defined constant (see Section 2.8.1).
- Let $l \in B$ be a definition line $\Gamma \triangleright D$ where D is of the form $c(x_1, \dots, x_n) := A$. Then the defined constant of the definition line, or $\text{defcons}(D)$, is c .
- $\text{defcons}(B) = \{\text{defcons}(D) \mid \Gamma \triangleright D \text{ is a line of } B, \text{ for some } \Gamma\}$. We call these constants the internally defined or internal constants of B .

Parameterized constants occurring in a book B outside a definition, represent defined notions with instantiated parameter lists. Such constants may not be internally defined: B is usually a text-fragment, part of a larger text, the rest of which is omitted. So, a parameterized constant occurring in a book B needs not have a corresponding definition as part of a line *inside* B . Such constants are called *externally defined* or *external* constants (relative to B).

EXAMPLE 3.3.

- In many books the constants \mathbb{N} and \mathbb{R} will be used without definition. (Both \mathbb{N} and \mathbb{R} are parameterized constants with empty parameter list.)
- One also uses many other well-known constants without defining them, such as $\sqrt{}$. (This constant needs a parameter list of length one, e.g., $\sqrt{5}$).
- Another well-known constant is \ln , which will be an external constant for WTT-books. It has one declared variable in its context. The corresponding definition line, which may not occur in the book B under consideration, is for example, $x \in \mathbb{R}, x > 0 \triangleright \ln(x) := \iota_{y \in \mathbb{R}}(e^y = x)$, with declaration $x \in \mathbb{R}$, so x is the declared variable¹⁶. A prerequisite for this definition of \ln is, that e has already been defined. When using the constant \ln we need a parameter list of length one, e.g., $\ln(5)$. Note that 5 matches with x as it belongs to the same linguistic collection \mathcal{T} .

¹⁶ $x > 0$ is an assumption, i.e. a statement which is not a declaration, see Section 2.8.

In order to judge weak well-typedness of the *externally* defined constants, we extend a book B on the front with a *preface* (a kind of *signature*), consisting of a list of constants, together with the weak types of its parameters and the resulting weak type of the constant. It is not essential how the externally defined constant is exactly defined, we can suffice with a description of the weak types connected with such a constant.

DEFINITION 3.4. We denote a preface for a book B by $\mathbf{pref}(B)$. The constants listed in this preface, are gathered in $\mathbf{prefcons}(B)$. If $c \in \mathbf{pref}(B)$, if $\kappa_1, \dots, \kappa_n$ is the list of the weak types of the parameters of c , and κ is the resulting weak type of the full construct $c(\dots)$, then we attach the type $\kappa_1 \times \dots \times \kappa_n \rightarrow \kappa$ to c .

EXAMPLE 3.5. A preface for a book B could look like:

constant name	weak type	constant name	weak type
\mathbb{R}	\mathcal{S}	\cup	$\mathcal{S} \times \mathcal{S} \rightarrow \mathcal{S}$
$\sqrt{\quad}$	$\mathcal{T} \rightarrow \mathcal{T}$	\geq	$\mathcal{T} \times \mathcal{T} \rightarrow \mathcal{S}$
$+$	$\mathcal{T} \times \mathcal{T} \rightarrow \mathcal{T}$	\wedge	$\mathcal{S} \times \mathcal{S} \rightarrow \mathcal{S}$

- \mathbb{R} has no parameters and is a set.
- $\sqrt{\quad}$ is a constant with one parameter, a term, delivering a term.
- \geq is a constant with two parameters, terms, delivering a statement.
- $\mathbf{prefcons}(B) = \{\mathbb{R}, \sqrt{\quad}, +, \cup, \geq, \wedge\}$.

We assume that a preface for a book B , describing the weak types for all externally defined constants of B , is constructed by the book-writer himself and that each book B is extended with an appropriate preface $\mathbf{pref}(B)$.

REMARK 3.6. When translating a given mathematical text into WTT, it may be good practice to also add two columns on either side of a WTT-book B , one with useful labels labeling lines in B , and another with comments going with specific lines. This makes it much easier for the translator of a WTT-book into type theory who does not have access to the original text. (See the example in Section 5.4.) This also helps since the original text possibly has interesting labels attached to paragraphs or to subtexts, such as Theorem 5.2 or Proof. Moreover, the original text may have many intermediate statements about interdependencies in the text, which are very useful when translating further into type theory. Think of comments such as use Theorem 5.2, by the definition of c or using formula 2.1.

3.3. VARIABLES

We define $\mathbf{dvar}(\Gamma)$ which collects all subject variables of *declarations* in Γ , in their order of appearance in Γ ,¹⁷ ignoring the possible subject variables occurring in *assumptions* in Γ . (See Sections 2.8 and 2.9.)

DEFINITION 3.7. *The list \mathbf{dvar} of declared variables of a context Γ is:*

- (1) *If $\Gamma = \emptyset$, then $\mathbf{dvar}(\Gamma) = \emptyset$.*
- (2a) *If $\Gamma = \Gamma', x : A$ and $x \notin \mathbf{dvar}(\Gamma')$, then $\mathbf{dvar}(\Gamma) = \mathbf{dvar}(\Gamma'), x$.*
- (2b) *Otherwise, if $\Gamma = \Gamma', S$, then $\mathbf{dvar}(\Gamma) = \mathbf{dvar}(\Gamma')$.*

Now the derivation rule for *variables* is (recall Notation 3.1 for $OK(B; \Gamma)$):

$$\frac{OK(B; \Gamma), \quad x \in \mathbb{V}^{\mathcal{T}/\mathbb{S}/\mathcal{S}}, \quad x \in \mathbf{dvar}(\Gamma)}{B; \Gamma \vdash x :: \mathcal{T}/\mathbb{S}/\mathcal{S}} \quad (\mathit{var})$$

NOTATION 3.8. *Here and in the rest of the paper, we combine two or more cases, distinguished by the slash /. In the first case of the above (var) rule, $x \in \mathbb{V}^{\mathcal{T}}$, the conclusion is: $B; \Gamma \vdash x :: \mathcal{T}$. In the second case, $x \in \mathbb{V}^{\mathbb{S}}$, we get: $B; \Gamma \vdash x :: \mathbb{S}$. In the third case, $x \in \mathbb{V}^{\mathcal{S}}$, we get: $B; \Gamma \vdash x :: \mathcal{S}$.*

3.4. CONSTANTS

Constants defined internally in B are divided in four kinds of *phrase definitions*, for terms, sets, nouns and adjectives, and also in *statement definitions*. See Section 2.8. We define, for parameter P , the weak type \mathbf{wt} of P with respect to B and Γ as: $\mathbf{wt}_{B; \Gamma}(P) = \mathbf{W}$ iff $B; \Gamma \vdash P :: \mathbf{W}$. The derivation rule for internal constants is:

$$\frac{\begin{array}{l} OK(B; \Gamma), \quad \Gamma' \triangleright D \in B, \\ \mathbf{dvar}(\Gamma') = \{x_1, \dots, x_n\}, \quad \mathbf{defcons}(D) = c \in \mathbb{C}^{\mathcal{T}/\mathbb{S}/\mathcal{N}/\mathcal{A}/\mathcal{S}}, \\ \mathbf{wt}_{B; \Gamma}(P_i) = \mathbf{wt}_{B; \Gamma'}(x_i), \text{ for all } i = 1, \dots, n. \end{array}}{B; \Gamma \vdash c(P_1, \dots, P_n) :: \mathcal{T}/\mathbb{S}/\mathcal{N}/\mathcal{A}/\mathcal{S}} \quad (\mathit{int-cons})$$

Note that the list of declared variables of Γ' , viz. x_1, \dots, x_n , is the same as the variable list following the defined constant c in the definition line $\Gamma' \triangleright D$ (this follows from the rule ($\mathit{int-def}$), see Section 3.8). Otherwise said: $D \equiv c(x_1, \dots, x_n) := \dots$. Hence, the above derivation rule determines how such a $c(x_1, \dots, x_n)$ can be instantiated, with result $c(P_1, \dots, P_n)$. The rule expresses

¹⁷ The *order* in the \mathbf{dvar} list of a context Γ is reflected in the order of the variables in the variable list going with any constant defined with respect to Γ . See Section 3.4.

binder name	weak type	binder name	weak type
min	$\mathcal{T} \rightarrow \mathcal{T}$	\cup	$\mathbb{S} \rightarrow \mathbb{S}$
Σ	$\mathcal{T} \rightarrow \mathcal{T}$	Set	$\mathcal{S} \rightarrow \mathbb{S}$
lim	$\mathcal{T} \rightarrow \mathcal{T}$	Noun	$\mathcal{S} \rightarrow \mathcal{N}$
\int	$\mathcal{T} \rightarrow \mathcal{T}$	Abst	$\mathcal{T}/\mathbb{S}/\mathcal{N} \rightarrow \mathcal{N}$
λ	$\mathcal{T} \rightarrow \mathcal{T}$	Adj	$\mathcal{S} \rightarrow \mathcal{A}$
ι	$\mathcal{S} \rightarrow \mathcal{T}/\mathbb{S}/\mathcal{S}$	\forall	$\mathcal{S} \rightarrow \mathcal{S}$

Figure 3. Binding symbols and their weak types

that an instantiation is only allowed if each variable x_i becomes replaced by a formula P_i of the same weak type \mathbf{W} as x_i (see example 3.3).

If c is an *external* constant of B , then it has a weak type as given in the preface of B . For such constants we have the following derivation rule:

$$\frac{OK(B;\Gamma), \quad c \text{ external to } B, \quad c :: \kappa_1 \times \dots \times \kappa_n \rightarrow \kappa, \\ B;\Gamma \vdash P_i :: \kappa_i \quad (i = 1, \dots, n)}{B;\Gamma \vdash c(P_1, \dots, P_n) :: \kappa} \quad (ext-cons)$$

A special kind of external constants is the pair \uparrow and \downarrow (see Section 2.4.1). They also have weak types, given in the following list:

constant name	weak type
\uparrow	$\mathcal{N} \rightarrow \mathbb{S}$
\downarrow	$\mathbb{S} \rightarrow \mathcal{N}$

We assume that this list is always part of the preface of a book B . Hence, expressions with \uparrow or \downarrow can also be derived with the rule (*ext-cons*) above.

3.5. BINDERS

Binders have fixed weak types. For example, binder ι (see Section 2.5.2) takes a statement and delivers a term or a set (the subscript Z is not important, in this respect). Figure 3 lists all the weak types corresponding to all binding symbols used in Section 2.5.

For expressions constructed by means of a binder we have a derivation rule:

$$\frac{OK(B;\Gamma,Z), \quad b \in B, \quad b :: \kappa_1 \rightarrow \kappa_2, \quad B;\Gamma,Z \vdash E :: \kappa_1}{B;\Gamma \vdash b_Z(E) :: \kappa_2} \quad (bind)$$

constant name	weak types
: SET	$\mathbb{S} \rightarrow \mathcal{S}$
: STAT	$\mathcal{S} \rightarrow \mathcal{S}$
:	$\mathcal{T} \times \mathbb{S} / \mathcal{N} / \mathcal{A} \rightarrow \mathcal{S}$

Figure 4. A basic list in a standard preface

Note the subscript Z : since E may depend on the subject variable of the declaration Z , we require that $B; \Gamma, Z \vdash E :: \kappa_1$, i.e., E is correct with respect to book B and context Γ extended with declaration Z . (This shift of Z from the context to the subscript is also present in the formation and abstraction rules of type theory, with which $\Pi_Z(E)$ and $\lambda_Z(E)$ are formed and typed.)

3.6. PHRASES

The derivation rule of the combination \mathcal{AN} of an adjective and a noun is:

$$\frac{B; \Gamma \vdash n :: \mathcal{N}, \quad B; \Gamma \vdash a :: \mathcal{A}}{B; \Gamma \vdash an :: \mathcal{N}} \quad (\text{adj-noun})$$

3.7. STATEMENTS

The derivation rules given above also suffice for the constructs given in the abstract syntax for statements (see Section 2.7). This includes constants \mathcal{C} for statements, as well as logical quantifiers covered by the construct $\mathbb{B}_Z^{\mathcal{S}}(\mathcal{E})$.

For *typings*, there are two kinds of derivations, dependent on the *level* of the statement: the first kind is for a statement of the form $s : \text{SET}$ or $S : \text{STAT}$, saying that s is a set or that S is a statement, the second is for statements of the form $t : s/n/a$, expressing that term t has type set s , noun n or adjective a . Both kinds can be treated with the above rule (*ext-cons*) provided we take the list given in Figure 4 in our standard preface.

For the first kind of typing, for example $s : \text{SET}$, we let $: \text{SET}$ be a *unary* constant, with $\mathbb{S} \rightarrow \mathcal{S}$ as weak type. This unary behaviour is necessary since SET is not covered by our linguistic categorization. For the second kind of statement we use $:$ as a *binary* constant.

3.8. DEFINITIONS

For internal definitions, we have the so-called (*int-def*) derivation rule:

$$\frac{B; \Gamma \vdash t/s/n/a/S :: \mathcal{T}/\mathbb{S}/\mathcal{N}/\mathcal{A}/S, \quad \text{dvar}(\Gamma) = \{x_1, \dots, x_n\}, \quad c \in \mathcal{C}^{\mathcal{T}/\mathbb{S}/\mathcal{N}/\mathcal{A}/S}, \quad c \notin \text{prefcons}(B) \cup \text{defcons}(B)}{B; \Gamma \vdash c(x_1, \dots, x_n) := t/s/n/a/S :: \mathcal{D}}$$

REMARK 3.9. *Sometimes (as a form of sugaring), the initial part of the parameter list of a defined constant is omitted, since it can be reconstructed by listing the declared variables (see Section 3.3) of the context.*

3.9. CONTEXTS

Empty contexts are typable and *declarations* and *assumptions* may be added.

$$\frac{\vdash B :: \mathbf{B}}{B \vdash \emptyset :: \mathbf{I}} \quad (\text{emp-cont})$$

$$\frac{OK(B; \Gamma), x \in V^{\mathbb{S}/S}, x \notin \text{dvar}(\Gamma)}{B \vdash \Gamma, x : \text{SET} / \text{STAT} :: \mathbf{I}} \quad (\text{set/stat-decl})$$

$$\frac{OK(B; \Gamma), B; \Gamma \vdash s/n :: \mathbb{S}/\mathcal{N}, x \in V^{\mathcal{T}}, x \notin \text{dvar}(\Gamma)}{B \vdash \Gamma, x : s/n :: \mathbf{I}} \quad (\text{term-decl})$$

$$\frac{OK(B; \Gamma), B; \Gamma \vdash S :: \mathcal{S}}{B \vdash \Gamma, S :: \mathbf{I}} \quad (\text{assump})^{18}$$

3.10. BOOKS

Books are lists of lines, containing either a definition or a statement in a context. The empty book is derivable and every weakly well-typed line, with respect to a book B , may lead to a weakly well-typed extension of B :

$$\frac{}{\vdash \emptyset :: \mathbf{B}} \quad (\text{emp-book})$$

¹⁸ Note that the statement S can be a typing $x : A$ but that x cannot be *new* with respect to Γ , since $B; \Gamma \vdash S :: \mathcal{S}$ (by Lemma 4.5, (2)). Hence, S cannot be a *declaration*.

$$\frac{B; \Gamma \vdash S/D :: S/\mathcal{D}}{\vdash B \circ \Gamma \triangleright S/D :: \mathbf{B}} \quad (\text{book-ext})$$

4. The meta theory and properties of the derivation system

4.1. FORMAL MACHINERY

We use *formula* (denoted A, Φ) to refer to either a sentence, a phrase or a declaration.

DEFINITION 4.1. *Free variables for formulas are defined as follows:*¹⁹

$$\begin{aligned} FV(x) &= \{x\} \\ FV(c(P_1, \dots, P_n)) &= FV(P_1) \cup \dots \cup FV(P_n) \\ FV(b_Z(E)) &= (FV(Z) \cup FV(E)) \setminus \{x\} \quad \text{if } Z \text{ is of the form } x : - \\ FV(an) &= FV(a) \cup FV(n) \\ FV(c(x_1, \dots, x_n) := u) &= FV(u) \cup \{x_1, \dots, x_n\} \quad \text{if } u \in \mathcal{T}/\mathcal{S}/\mathcal{N}/\mathcal{A}/\mathcal{S} \\ FV(x : s/n) &= FV(s/n) \\ FV(x : SET / STAT) &= \emptyset \end{aligned}$$

Free variables for contexts/lines/books are defined by:

$$(\text{Contexts}) \quad FV(\emptyset) = \emptyset, \quad FV(\Gamma, Z/S) = FV(\Gamma) \cup FV(Z/S).$$

$$(\text{Lines}) \quad FV(\Gamma \triangleright S/D) = FV(\Gamma) \cup FV(S/D).$$

$$(\text{Books}) \quad FV(\emptyset) = \emptyset \text{ and } FV(B \circ l) = FV(B) \cup FV(l).$$

DEFINITION 4.2. *Free constants for formulas are defined as follows:*

$$\begin{aligned} FC(x) &= \emptyset \\ FC(c(P_1, \dots, P_n)) &= \{c\} \cup FC(P_1) \cup \dots \cup FC(P_n) \\ FC(b_Z(E)) &= FC(Z) \cup FC(E) \\ FC(an) &= FC(a) \cup FC(n) \\ FC(c(x_1, \dots, x_n) := u) &= FC(u) \quad \text{if } u \in \mathcal{T}/\mathcal{S}/\mathcal{N}/\mathcal{A}/\mathcal{S} \\ FC(x : s/n) &= FC(s/n) \\ FC(x : SET / STAT) &= \emptyset \end{aligned}$$

Free constants for contexts/lines/books are defined by:

$$(\text{Contexts}) \quad FC(\emptyset) = \emptyset, \quad FC(\Gamma, Z/S) = FC(\Gamma) \cup FC(Z/S).$$

$$(\text{Lines}) \quad FC(\Gamma \triangleright S/D) = FC(\Gamma) \cup FC(S/D).$$

¹⁹ In the second clause, $c(P_1, \dots, P_n)$ is an instantiation, not the left hand side of a definition.

(Books) $FC(\emptyset) = \emptyset$ and $FC(B \circ l) = FC(B) \cup FC(l)$.

DEFINITION 4.3. *Subformulas of formulas are defined as follows:*

$$\begin{aligned}
\text{subfor}(x) &= \{x\} \\
\text{subfor}(c(P_1, \dots, P_n)) &= \{c(P_1, \dots, P_n)\} \cup \text{subfor}(P_1) \cup \dots \cup \text{subfor}(P_n) \\
\text{subfor}(b_Z(E)) &= \{b_Z(E)\} \cup \text{subfor}(Z) \cup \text{subfor}(E) \\
\text{subfor}(an) &= \{an\} \cup \text{subfor}(a) \cup \text{subfor}(n) \\
\text{subfor}(c(x_1, \dots, x_n) := u) &= \{c(x_1, \dots, x_n) := u\} \cup \text{subfor}(u) \\
\text{subfor}(x : s/n) &= \text{subfor}(s/n) \\
\text{subfor}(x : SET / STAT) &= \emptyset
\end{aligned}$$

Subformulas of contexts/lines/books are defined by:

(Contexts) $\text{subfor}(\emptyset) = \emptyset$, $\text{subfor}(\Gamma, Z/S) = \text{subfor}(\Gamma) \cup \text{subfor}(Z/S)$.

(Lines) $\text{subfor}(\Gamma \triangleright S/D) = \text{subfor}(\Gamma) \cup \text{subfor}(S/D)$.

(Books) $\text{subfor}(\emptyset) = \emptyset$, $\text{subfor}(B \circ l) = \text{subfor}(B) \cup \text{subfor}(l)$.

The next convention is needed (e.g., in the proof of Lemma 4.10).

CONVENTION 4.4. *We assume a version of the Barendregt Convention where names of free variables are distinct from bound ones and in the same book/context/line/formula, we use different names for bound variables. For example, if $B; \Gamma \vdash b_Z(E) :: \kappa_2$ then we assume that the declared variable in Z is different from any declared variable in Γ . If this is not the case then we rename the declared variable of Z in $b_Z(E)$ to a name of a fresh variable.*

LEMMA 4.5 (Free Variables).

- (1) If $B \vdash \Gamma :: \mathbf{I}$, then the declared variables in Γ are distinct.
- (2) If $B; \Gamma \vdash A :: \mathbf{W}$ then $FV(A) \subseteq \text{dvar}(\Gamma)$.
- (3) If $B \vdash \Gamma :: \mathbf{I}$ where $\Gamma \equiv \Gamma', (x : A)/A, \Gamma''$, then $FV(A) \subseteq \text{dvar}(\Gamma')$.

Proof: (1) By induction on the derivation $B \vdash \Gamma :: \mathbf{I}$.

(2) By induction on the derivation $B; \Gamma \vdash A :: \mathbf{W}$.

(3) By induction on the derivation $B \vdash \Gamma :: \mathbf{I}$ using (1). ⊠

DEFINITION 4.6.

(B' is a subbook of B) We say $B' \subseteq B$ if there exists $B'' : B \equiv B' \circ B''$.

(Γ' is a subcontext of Γ) We say $\Gamma' \subseteq \Gamma$ if there exists $\Gamma'' : \Gamma \equiv \Gamma', \Gamma''$.

(compatibility with a book) Let B be a book and $l = \Gamma \triangleright D$ be a line. A definition $c(x_1, \dots, x_n) := t/s/n/a/S$ is compatible with B (resp. with l) if $c \notin \text{prefcons}(B) \cup \text{defcons}(B)$ (resp. $c \neq \text{defcons}(D)$). The line l is compatible with B if $\text{defcons}(l) \notin (\text{prefcons}(B) \cup \text{defcons}(B))$.

LEMMA 4.7 (Presence of definitions). *The following hold:*

1. If $B; \Gamma \vdash A :: \mathbf{W}$ then either A is free of definitions, or A is the only definition (i.e. of the form $c(x_1, \dots, x_n) := A'$) in A .
2. If $B \vdash \Gamma :: \mathbf{I}$ then Γ is free of definitions. That is, A is free of definitions for every $x : A$ or A in Γ .

Proof: By induction on the derivations $B; \Gamma \vdash A :: \mathbf{W}$ and $B \vdash \Gamma :: \mathbf{I}$. ⊠

The next lemma studies typability inside a weakly well-typed book.

LEMMA 4.8 (Subcontext). *If $B \vdash \Gamma :: \mathbf{I}$ and $\Gamma' \subseteq \Gamma$ then $B \vdash \Gamma' :: \mathbf{I}$.*

Proof: By induction on the derivation $B \vdash \Gamma :: \mathbf{I}$. ⊠

LEMMA 4.9 (Subbook). *If $\vdash B :: \mathbf{B}$ and $B' \subseteq B$ then $\vdash B' :: \mathbf{B}$.*

Proof: Corollary of the generation lemma below. ⊠

4.2. PROPERTIES OF WEAK TYPING

LEMMA 4.10 (Thinning/Weakening).

- (1) Let $B \vdash \Gamma :: \mathbf{I}$ and $\Gamma' \subseteq \Gamma$. If $B; \Gamma' \vdash A :: \mathbf{W}$ and A is not a definition, then $B; \Gamma \vdash A :: \mathbf{W}$.
- (2) Let $\vdash B \circ l :: \mathbf{B}$, $\vdash B \circ B' :: \mathbf{B}$, and l compatible with $B \circ B'$. Then:
 - (2a) $\vdash B \circ l \circ B' :: \mathbf{B}$.
 - (2b) If $B \circ B'; \Gamma \vdash A :: \mathbf{W}$ where if A is a definition then A is compatible with l then $B \circ l \circ B'; \Gamma \vdash A :: \mathbf{W}$.
 - (2c) If $B \circ B' \vdash \Gamma :: \mathbf{I}$, then $B \circ l \circ B' \vdash \Gamma :: \mathbf{I}$.
- (3) Let $\vdash B :: \mathbf{B}$ and $B' \subseteq B$. We have:
 - (3a) If $B'; \Gamma \vdash A :: \mathbf{W}$ where if A is a definition then it is compatible with B , then $B; \Gamma \vdash A :: \mathbf{W}$,
 - (3b) If $B' \vdash \Gamma :: \mathbf{I}$, then $B \vdash \Gamma :: \mathbf{I}$.
- (4) Let $B \vdash \Gamma, \Gamma' :: \mathbf{I}$, $B \vdash \Gamma, \Gamma'' :: \mathbf{I}$, and $\text{dvar}(\Gamma') \cap \text{dvar}(\Gamma'') = \emptyset$.

(4a) $B \vdash \Gamma, \Gamma', \Gamma'' :: \mathbf{I}$.

(4b) Let $B; \Gamma, \Gamma' \vdash A :: \mathbf{W}$ and A is not a definition. $B; \Gamma, \Gamma', \Gamma'' \vdash A :: \mathbf{W}$.

Proof (1)..(4) by induction on the length of derivations. Use Lemma 4.7. \boxtimes

LEMMA 4.11 (Generation).

(1) If $B; \Gamma \vdash x :: \mathbf{W}$ then $OK(B; \Gamma)$, $x \in \text{dvar}(\Gamma)$ and

(1a) $\mathbf{W} = \mathcal{T}$ and $x \in \mathbb{V}^{\mathcal{T}}$, or

(1b) $\mathbf{W} = \mathbb{S}$ and $x \in \mathbb{V}^{\mathbb{S}}$, or

(1c) $\mathbf{W} = \mathcal{S}$ and $x \in \mathbb{V}^{\mathcal{S}}$.

(2) If $B; \Gamma \vdash c(P_1, \dots, P_n) :: \mathbf{W}$ then $OK(B; \Gamma)$ and

(2a) either $\mathbf{W} = \mathcal{T}/\mathbb{S}/\mathcal{N}/\mathcal{A}/\mathcal{S}$, there is $D, \Gamma', x_1, \dots, x_n$ where
 $\text{dvar}(\Gamma') = \{x_1, \dots, x_n\}$, $\text{defcons}(D) = c \in \mathcal{C}^{\mathcal{T}/\mathbb{S}/\mathcal{N}/\mathcal{A}/\mathcal{S}}$,
 $\Gamma' \triangleright D \in B$, and for all $i \in \{1, \dots, n\} : \text{wt}_{B; \Gamma'}(P_i) = \text{wt}_{B; \Gamma'}(x_i)$,

(2b) or c is external to B , and there is $\kappa_1, \dots, \kappa_n$ such that

$c :: \kappa_1 \times \dots \times \kappa_n \rightarrow \mathbf{W}$ and $B; \Gamma \vdash P_i :: \kappa_i$ ($i = 1, \dots, n$).

(3) If $B; \Gamma \vdash b_Z(A) :: \kappa_2$ then $OK(B; \Gamma, Z)$ and there is κ_1 such that
 $b :: \kappa_1 \rightarrow \kappa_2$, and $B; \Gamma, Z \vdash A :: \kappa_1$.

(4) If $B; \Gamma \vdash a_n :: \mathbf{W}$ then $\mathbf{W} = \mathcal{N}$, $OK(B; \Gamma)$, $B; \Gamma \vdash n :: \mathcal{N}$ and
 $B; \Gamma \vdash a :: \mathcal{A}$.

(5) If $B; \Gamma \vdash c(x_1, \dots, x_n) := t/s/n/a/S :: \mathbf{W}$ then $\mathbf{W} = \mathcal{D}$, $OK(B; \Gamma)$,
 $c \in \mathcal{C}^{\mathcal{T}/\mathbb{S}/\mathcal{N}/\mathcal{A}/\mathcal{S}}$, $c \notin \text{prefcons}(B) \cup \text{defcons}(B)$, $\text{dvar}(\Gamma) =$
 $\{x_1, \dots, x_n\}$ and $B; \Gamma \vdash t/s/n/a/S :: \mathcal{T}/\mathbb{S}/\mathcal{N}/\mathcal{A}/\mathcal{S}$.

(6) If $B \vdash \Gamma :: \mathbf{I}$ then $\vdash B :: \mathbf{B}$ and if $\Gamma \neq \emptyset$ then

(6a) if $\Gamma = \Gamma', x : \mathbf{W}$ then $B \vdash \Gamma' :: \mathbf{I}$, $x \notin \text{dvar}(\Gamma')$ and

if $\mathbf{W} = \text{SET} / \text{STAT}$ then $x \in \mathbb{V}^{\mathbb{S}}/\mathbb{V}^{\mathcal{S}}$

else if $\mathbf{W} = s/n$ then $B; \Gamma' \vdash s/n :: \mathbb{S}/\mathcal{N}$, and $x \in \mathbb{V}^{\mathcal{T}}$.

(6b) If $\Gamma = \Gamma', S$ then $B \vdash \Gamma' :: \mathbf{I}$, and $B; \Gamma' \vdash S :: \mathcal{S}$.

(7) If $\vdash B :: \mathbf{B}$ then either $B = \emptyset$ or there is Γ, B' and S/D such that
 $B = B' \circ \Gamma \triangleright S/D$, $OK(B'; \Gamma)$, and $B'; \Gamma \vdash S/D :: \mathcal{S}/\mathcal{D}$.

Proof Take a derivation in one of the above cases. We follow the derivation until the typed construct on the left of $::$ first appears. This is done by:

(var) , (int-cons) , (ext-cons) , (bind) , (adj-noun) , (int-def) , (emp-cont) ,
 (set/stat-decl) , (term-decl) , (assump) , (emp-book) and (book-ext) for

(1), (2a), (2b), (3), (4), (5), (6) where $\Gamma = \emptyset$, (6a) and $\mathbf{W} = \text{SET} / \text{STAT}$, (6a) and $\mathbf{W} = s/n$, (6b), (7) where $B = \emptyset$ and (7) where $B \neq \emptyset$ respectively. The lemma follows by inspection of the used rule. We need Lemma 4.8 for (*adj-noun*) and Lemma 4.7 for (*int-def*). \boxtimes

LEMMA 4.12 (Free Constants).

- (1) If $\vdash B :: \mathbf{B}$, then the defined constants in B are distinct.
- (2) If $B; \Gamma \vdash \Phi :: \mathbf{W}$, then $FC(\Phi) \subseteq \text{prefcons}(B) \cup \text{defcons}(B)$.
- (3) If $B \vdash \Gamma :: \mathbf{I}$ where $\Gamma \equiv \Gamma', x : A, \Gamma''$ or $\Gamma \equiv \Gamma', A, \Gamma'$, then $FC(A) \subseteq \text{prefcons}(B) \cup \text{defcons}(B)$.

Proof (1) By induction on the size of B .

(2) We prove by induction on the length of the derivations that if $B; \Gamma \vdash A :: \mathbf{W}$ or $B \vdash \Gamma, A/x : A :: \mathbf{I}$ then $FV(A) \subseteq \text{prefcons}(B) \cup \text{defcons}(B)$.

(3) We show by induction on the derivation $B \vdash \Gamma :: \mathbf{I}$ that for any Γ', Γ'', A such that $\Gamma \equiv \Gamma', x : A, \Gamma''$ or $\Gamma \equiv \Gamma', A, \Gamma''$, we have $FC(A) \subseteq \text{prefcons}(B) \cup \text{defcons}(B)$. We use the following property we showed in (2):

- (*) if $B \vdash \Gamma, A :: \mathbf{I}$ or $B \vdash \Gamma, x : A :: \mathbf{I}$ then $FC(A) \subseteq \text{prefcons}(B) \cup \text{defcons}(B)$. \boxtimes

LEMMA 4.13 (Uniqueness of Types).

If $B; \Gamma \vdash A :: \mathbf{W}_1$ and $B; \Gamma \vdash A :: \mathbf{W}_2$, then $\mathbf{W}_1 \equiv \mathbf{W}_2$.

Proof: This is now a simple corollary of the generation lemma. \boxtimes

LEMMA 4.14 (Context). If $B; \Gamma \vdash A :: \mathbf{W}$ then $B \vdash \Gamma :: \mathbf{I}$.

Proof: By induction on the derivation $B; \Gamma \vdash A :: \mathbf{W}$ using, as $\Gamma \subseteq \Gamma, Z$, Lemma 4.8 in the (*bind*) rule where $B; \Gamma \vdash b_Z(E) :: \kappa_2$ comes from $B \vdash \Gamma, Z :: \mathbf{I}$ amongst other things. \boxtimes

LEMMA 4.15 (Subformula). The following hold:

- (1) If $B \vdash \Gamma :: \mathbf{I}$ then for every subformula A of Γ , there exists \mathbf{W} and Γ' where $\Gamma \subseteq \Gamma'$ such that $B; \Gamma' \vdash A :: \mathbf{W}$.
- (2) If $B; \Gamma \vdash A :: \mathbf{W}$ then for every subformula A' of A , there exists \mathbf{W} , and Γ' where $\Gamma \subseteq \Gamma'$ such that $B; \Gamma' \vdash A' :: \mathbf{W}$.

Proof: By simultaneous induction on the length of the derivation. \boxtimes

LEMMA 4.16 (Subformula property). If $\vdash B :: \mathbf{B}$ and $B \equiv B' \circ \Gamma \triangleright A \circ B''$:

- (1) $B' \vdash \Gamma :: \mathbf{I}$

(2) $B'; \Gamma \vdash A :: S/\mathcal{D}$.

(3) Let A' be a subformula of either Γ or of A .
There exist \mathbf{W} and Γ' such that $\Gamma \subseteq \Gamma' : B'; \Gamma' \vdash A' :: \mathbf{W}$.

Proof: (1) + (2) are shown together by induction on the length of B' .

- Case $B'' = \emptyset$ then by the generation lemma $OK(B'; \Gamma)$
(hence $B' \vdash \Gamma :: \mathbf{I}$) and $B'; \Gamma \vdash A :: S/\mathcal{D}$. We are done.
- Case $B \equiv B' \circ \Gamma \triangleright A \circ B''_1 \circ \Gamma' \triangleright A'$,
then by the generation lemma $\vdash B' \circ \Gamma \triangleright A \circ B''_1 :: \mathbf{B}$
hence by IH $B' \vdash \Gamma :: \mathbf{I}$ and $B'; \Gamma \vdash A :: S/\mathcal{D}$.

(3) : By (1) + (2) we have: $B' \vdash \Gamma :: \mathbf{I}$ and $B'; \Gamma \vdash A :: S/\mathcal{D}$. Hence, by Lemma 4.15 we have the desired (3). \square

DEFINITION 4.17. We define substitution in a formula as follows:

$$\begin{aligned}
x[x := A] &= A \\
y[x := A] &= y \quad \text{for } x \neq y \\
c(P_1, \dots, P_n)[x := A] &= c(P_1[x := A], \dots, P_n[x := A]) \\
(b_Z(E))[x := A] &= b_{Z[x:=A]}(E[x := A]) \\
(an)[x := A] &= a[x := A]n[x := A] \\
(c(x_1, \dots, x_n) := u)[x := A] &= c(x_1, \dots, x_n) := u[x := A] \text{ for } x \notin \{x_1, \dots, x_n\} \\
(y : s/n)[x := A] &= y : s[x := A]/n[x := A] \\
(y : SET / STAT)[x := A] &= y : SET / STAT
\end{aligned}$$

Substitution in contexts/lines/books is defined as follows:

(Contexts) $\emptyset[x := A] = \emptyset$ and $(\Gamma, Z/S)[x := A] = \Gamma[x := A], Z[x := A]/S[x := A]$.

(Lines) $(\Gamma \triangleright S/D)[x := A] = \Gamma[x := A] \triangleright S[x := A]/D[x := A]$.

(Books) $\emptyset[x := A] = \emptyset$ and $(B \circ l)[x := A] = B[x := A] \circ l[x := A]$.

LEMMA 4.18 (Substitution).

If $B; \Gamma, x : A \vdash x :: \mathcal{T}/\mathcal{S}/S$ and $B; \Gamma \vdash A' :: \mathcal{T}/\mathcal{S}/S$ then:

(1) If $B; \Gamma, x : A, \Delta \vdash \Phi :: \mathbf{W}$, and Φ is not a definition, then
 $B; \Gamma, \Delta[x := A'] \vdash \Phi[x := A'] :: \mathbf{W}$.

(2) If $B \vdash \Gamma, x : A, \Delta :: \mathbf{I}$, then $B \vdash \Gamma, \Delta[x := A'] :: \mathbf{I}$.

Proof By simultaneous induction on the length of the derivations. \square

LEMMA 4.19 (Condensing). The following hold:

- (1) If $B; \Gamma, x : A, \Delta \vdash \Phi :: \mathbf{W}$, Φ is not a definition, and $x \notin \Delta, \Phi$, then $B; \Gamma, \Delta \vdash \Phi :: \mathbf{W}$.
- (2) If $B \vdash \Gamma, x : A, \Delta :: \mathbf{I}$ and $x \notin \Delta$, then $B \vdash \Gamma, \Delta :: \mathbf{I}$.

Proof Show (1) and (2) simultaneously by induction on the length of the derivation. \square

The generation lemma implies that in order to verify that a certain construct has a certain weak type, there is at most one derivation rule applicable. Hence it is easy to check weak typing. Our derivations are syntax-driven:

THEOREM 4.20 (Syntax-driven Derivations). *Derivations are syntax-driven.*

COROLLARY 4.21 (Decidability of weak type checking and weak typability).

- (1) *Weak type checking is decidable: there is a decision procedure for the question $B; \Gamma \vdash \Phi :: \mathbf{W}$?.*
- (2) *Weak typability is computable: there is a procedure deciding whether an answer exists for $B; \Gamma \vdash \Phi :: ?$ and if so, delivering the answer.*

Proof (1) By induction on the number of symbols on the right and left of \vdash using Lemma 4.11. (2) is also by induction on the number of symbols on the right and left of \vdash using (1). \square

LEMMA 4.22 (Swap). *The following hold:*

- If $B; \Gamma, x : A, y : A', \Delta \vdash \Phi :: \mathbf{W}$ and $x \notin FV(A')$ then $B; \Gamma, y : A', x : A, \Delta \vdash \Phi :: \mathbf{W}$.
- If $B \vdash \Gamma, x : A, y : A', \Delta :: \mathbf{I}$ and $x \notin FV(A')$ then $B \vdash \Gamma, y : A', x : A, \Delta :: \mathbf{I}$

Proof By induction on the length of the derivations using Lemma 4.10. \square

DEFINITION 4.23 (Context Restriction).

Let $FV(\Phi) = \{x_1, \dots, x_n\}$ and $\Gamma = \Gamma_1, x_1 : P_1, \Gamma_2, x_2 : P_2, \dots, \Gamma_n, x_n : P_n, \Gamma_{n+1}$. We define $\Gamma \downarrow FV(\Phi)$ to be $\Gamma_1, x_1 : P_1, \Gamma_2, x_2 : P_2, \dots, \Gamma_n, x_n : P_n$.

Note that $\Gamma \downarrow FV(\Phi) \subseteq \Gamma$ and if $FV(\Phi) \subseteq FV(\Phi')$ then

$$\Gamma \downarrow FV(\Phi) \subseteq \Gamma \downarrow FV(\Phi').$$

LEMMA 4.24 (Context Restriction).

If $B; \Gamma \vdash \Phi :: \mathbf{W}$ then $B; \Gamma \downarrow FV(\Phi) \vdash \Phi :: \mathbf{W}$.

Proof By induction on $B; \Gamma \vdash \Phi :: \mathbf{W}$ using Lemmas 4.8, 4.10, 4.5, 4.22 and 4.7. \square

LEMMA 4.25 (Simultaneous Substitution). *Let $\text{dvar}(\Gamma_2) = \{x_1, \dots, x_n\}$. If $B; \Gamma_1, \Gamma_2 \vdash x_i :: \mathcal{T}/\mathbb{S}/S$ and $B; \Gamma_1 \vdash P_i :: \mathcal{T}/\mathbb{S}/S$ then:*

- (1) *If $B; \Gamma_1, \Gamma_2, \Gamma_3 \vdash \Phi :: \mathbf{W}$, and Φ is not a definition, then $B; \Gamma_1, \Gamma_3[x_i := P_i] \vdash \Phi[x_i := P_i] :: \mathbf{W}$.*
- (2) *If $B \vdash \Gamma_1, \Gamma_2, \Gamma_3 :: \mathbf{I}$, then $B \vdash \Gamma_1, \Gamma_3[x_i := P_i] :: \mathbf{I}$.*

Proof We show both (1) and (2) by induction on the length of the derivation $B; \Gamma_1, \Gamma_2, \Gamma_3 \vdash \Phi :: \mathbf{W}$ and $B \vdash \Gamma_1, \Gamma_2, \Gamma_3 :: \mathbf{I}$. Use Lemmas 4.8, 4.7 and 4.10. \square

4.3. DEFINITIONAL REDUCTION

We use symbol $\xrightarrow{\delta}$ for the reductional relation of *definition unfolding*. Its definition is as expected. First, we define compatibility:

DEFINITION 4.26. *We define compatibility in the usual way as follows:*

$$\begin{array}{c}
\frac{P_i \xrightarrow{\delta} P'_i \text{ for } 1 \leq i \leq n}{c(P_1, \dots, P_i, \dots, P_n) \xrightarrow{\delta} c(P_1, \dots, P'_i, \dots, P_n)} \quad \text{for } c \in \mathbf{C} \text{ and } P_j \in \mathcal{P} \\
\frac{Z \xrightarrow{\delta} Z'}{b_Z(E) \xrightarrow{\delta} b_{Z'}(E)} \quad \frac{E \xrightarrow{\delta} E'}{b_Z(E) \xrightarrow{\delta} b_Z(E')} \quad \text{for } Z \in \mathbf{Z}, E \in \mathcal{E} \text{ and } b \in \mathbf{B} \\
\frac{a \xrightarrow{\delta} a'}{an \xrightarrow{\delta} a'n} \quad \frac{n \xrightarrow{\delta} n'}{an \xrightarrow{\delta} an'} \quad \text{for } a \in \mathcal{A} \text{ and } n \in \mathcal{N} \\
\frac{u \xrightarrow{\delta} u'}{c(x_1, \dots, x_n) := u \xrightarrow{\delta} c(x_1, \dots, x_n) := u'} \\
\frac{s/n \xrightarrow{\delta} s'/n'}{y : s/n \xrightarrow{\delta} y : s'/n'} \quad \text{for } s \in \mathbb{S} \text{ or } s \in \mathcal{N}.
\end{array}$$

DEFINITION 4.27. *Let $\vdash B :: \mathbf{B}$ and $\Gamma \triangleright c(x_1, \dots, x_n) := \Phi$ a line in B . $\xrightarrow{\delta}$ is the compatible relation on subterms of B generated by $c(P_1, \dots, P_n) \xrightarrow{\delta} \Phi[x_i := P_i]$, provided that the occurrence of the latter constant c is not the defining occurrence in the defining line mentioned. As δ -reduction depends on the book in question, we write $B \vdash c(P_1, \dots, P_n) \xrightarrow{\delta} \Phi[x_i := P_i]$.*

$\Phi[x_i := P_i]$ stands for the simultaneous substitution of x_i by P_i in Φ .

THEOREM 4.28 (Subject Reduction).

If $B; \Gamma \vdash \Phi :: \mathbf{W}$ and $B \vdash \Phi \xrightarrow{\delta} \Psi$, then $B; \Gamma \vdash \Psi :: \mathbf{W}$.

Proof Define $B \vdash \Gamma \xrightarrow{\delta} \Gamma'$ iff Γ and Γ' are exactly the same except for either

- one $x : A$ in Γ which appears as $x : A'$ in Γ' with $B \vdash A \xrightarrow{\delta} A'$, or
- one S in Γ which appears as S' in Γ' with $B \vdash S \xrightarrow{\delta} S'$.

By simultaneous induction on the derivation we show using Lemmas 4.16, 4.11, 4.7, 4.10 and 4.25 that:

- (1) If the derivation is $B; \Gamma \vdash \Phi :: \mathbf{W}$ and $B \vdash \Phi \xrightarrow{\delta} \Psi$, then $B; \Gamma \vdash \Psi :: \mathbf{W}$.
- (2) If the derivation is $B; \Gamma \vdash \Phi :: \mathbf{W}$ and $B \vdash \Gamma \xrightarrow{\delta} \Gamma'$, then $B; \Gamma' \vdash \Phi :: \mathbf{W}$
 else if the derivation is $B \vdash \Gamma :: \mathbf{\Gamma}$ and $B \vdash \Gamma \xrightarrow{\delta} \Gamma'$ then $B \vdash \Gamma' :: \mathbf{\Gamma}$. \square

In what follows we establish the Church Rosser and Strong Normalisation properties of δ -reduction by adapting the lines of [19].

The next definition (which is independent of contexts) gives for each formula, its corresponding formula where all definitions of a book are unfolded.

DEFINITION 4.29. *Take a book B , and a formula Φ . We define $|\Phi|_B$ as:*

$$\begin{aligned}
 |x|_B &= x \\
 |c(P_1, \dots, P_n)|_B &= |\Phi|_B[x_i := |P_i|_B] \text{ if } \Gamma' \triangleright c(x_1, \dots, x_n) := \Phi \text{ is a line of } B. \\
 |c(P_1, \dots, P_n)|_B &= c(|P_1|_B, \dots, |P_n|_B) \text{ if } c \text{ is external constant (in } \mathbf{pref}(B)) \\
 |b_Z(E)|_B &= b_{|Z|_B}(|E|_B) \\
 |c(x_1, \dots, x_n) := u|_B &= c(x_1, \dots, x_n) := |u|_B \\
 |an|_B &= |a|_B |n|_B \\
 |x : A|_B &= x : |A|_B
 \end{aligned}$$

We define: $|\emptyset|_B = \emptyset$, $|\Gamma, x : A|_B = |\Gamma|_B, x : |A|_B$ and $|\Gamma, S|_B = |\Gamma|_B, |S|_B$.

LEMMA 4.30. *If $B; \Gamma \vdash \Phi :: \mathbf{W}$ then $B \vdash \Phi \xrightarrow{\delta} |\Phi|_B$.*

Proof: By induction on the number of symbols in Φ . The interesting case is when Φ is $c(P_1, \dots, P_n)$ and $\Gamma' \triangleright c(x_1, \dots, x_n) := \Phi'$ is a line of B . Then $B \vdash c(P_1, \dots, P_n) \xrightarrow{\delta} \Phi'[x_i := P_i] \xrightarrow{\delta}^{IH} |\Phi'|_B[x_i := |P_i|_B] = |c(P_1, \dots, P_n)|_B$. \square

LEMMA 4.31. *For $1 \leq i \leq n$: $|\Phi[x_i := P_i]|_B = |\Phi|_B[x_i := |P_i|_B]$.*

Proof: By induction on the structure of Φ . \square

LEMMA 4.32. *If $B \vdash \Phi \xrightarrow{\delta} \Phi'$ then $|\Phi|_B = |\Phi'|_B$.*

Proof: By induction on the structure of Φ . The only interesting case is when Φ is $(c(P_1, \dots, P_n))$ where $\Gamma' \triangleright c(x_1, \dots, x_n) := \Phi'$ is a line of B and $B \vdash c(P_1, \dots, P_n) \xrightarrow{\delta} \Phi'[x_i := P_i]$. In this case use Lemma 4.31. \square

THEOREM 4.33 (Church Rosser for δ -reduction). *If $B \vdash \Phi \xrightarrow{\delta} \Phi_1$ and $B \vdash \Phi \xrightarrow{\delta} \Phi_2$ then there exists Φ_3 such that $B \vdash \Phi_1 \xrightarrow{\delta} \Phi_3$ and $B \vdash \Phi_2 \xrightarrow{\delta} \Phi_3$.*

Proof: By Lemma 4.30, $B \vdash \Phi \xrightarrow{\delta} |\Phi|_B$, $B \vdash \Phi_1 \xrightarrow{\delta} |\Phi_1|_B$, and $B \vdash \Phi_2 \xrightarrow{\delta} |\Phi_2|_B$. As $B \vdash \Phi \xrightarrow{\delta} \Phi_1$ then by Lemma 4.32 $|\Phi|_B = |\Phi_1|_B$. Similarly, as $B \vdash \Phi \xrightarrow{\delta} \Phi_2$ then $|\Phi|_B = |\Phi_2|_B$. Hence Take $\Phi_3 = |\Phi_1|_B = |\Phi_2|_B$. Now $B \vdash \Phi_1 \xrightarrow{\delta} \Phi_3$ and $B \vdash \Phi_2 \xrightarrow{\delta} \Phi_3$. \square

DEFINITION 4.34. *Let $B; \Gamma \vdash \Phi :: \mathbf{W}$. We say that Φ is in δ -normal form in the book B if there is no Φ' such that $B \vdash \Phi \xrightarrow{\delta} \Phi'$.*

THEOREM 4.35. *Let $B; \Gamma \vdash \Phi :: \mathbf{W}$. Then Φ is in δ -normal form in B iff $FC(\Phi) \cap \text{defcons}(B) = \emptyset$.*

Proof: By induction on the structure of Φ . \square

COROLLARY 4.36 (Weak Normalisation for δ -reduction).

Let $B; \Gamma \vdash \Phi :: \mathbf{W}$. Then $|\Phi|_B$ is the δ -nf of Φ .

Proof: It is easy to show that $FC(|\Phi|_B) \cap \text{defcons}(B) = \emptyset$. Hence by Theorem 4.35 $|\Phi|_B$ is in δ -normal form. But by Lemma 4.30 $B \vdash \Phi \xrightarrow{\delta} |\Phi|_B$ and by Church Rosser Theorem 4.33 the δ -normal form is unique. Hence $|\Phi|_B$ is the δ -normal form of Φ . \square

In order to establish strong normalisation of δ we introduce a measure function \mathcal{M} which decreases with δ -reduction.

DEFINITION 4.37. *Let $\mathcal{M} : \mathbf{B} \times \Phi \mapsto \mathbb{N}$ be defined as follows:*

$$\begin{aligned} \mathcal{M}_B(x) &= 1 \\ \mathcal{M}_B(c(P_1, \dots, P_n)) &= 1 + \sum_{i=1}^n \mathcal{M}_B(P_i) \\ &\quad \text{if } B = B' \circ \Gamma' \triangleright c(x_1, \dots, x_n) := u \circ B'' \\ \mathcal{M}_B(c(P_1, \dots, P_n)) &= \sum_{i=1}^n \mathcal{M}_B(P_i) \quad \text{if } c \text{ is an external constant} \\ \mathcal{M}_B(an) &= \mathcal{M}_B(a) + \mathcal{M}_B(n) \\ \mathcal{M}_B(c(x_1, \dots, x_n) := u) &= \mathcal{M}_B(u) \\ \mathcal{M}_B(b_Z(E)) &= \mathcal{M}_B(Z) + \mathcal{M}_B(E) \end{aligned}$$

LEMMA 4.38. *Let $B; \Gamma \vdash \Phi :: \mathbf{W}$. If $\text{wt}_{B; \Gamma}(x_i) = \text{wt}_{B; \Gamma}(P_i)$ for $1 \leq i \leq n$, then $\mathcal{M}_B(\Phi[x_i := P_i]) \leq \sum_{i=1}^n \mathcal{M}_B(\Phi) \mathcal{M}_B(P_i)$.*

Proof: By induction on the structure of Φ .

- Case $\Phi = x_i$, $1 \leq i \leq n$ then
 $\mathcal{M}_B(\Phi[x_i := P_i]) = \mathcal{M}_B(P_i) \leq \sum_{i=1}^n \mathcal{M}_B(P_i) = \sum_{i=1}^n \mathcal{M}_B(x_i) \mathcal{M}_B(P_i)$.
- Case $\Phi = x \neq x_i$, $1 \leq i \leq n$ then
 $\mathcal{M}_B(\Phi[x_i := P_i]) = \mathcal{M}_B(x) = 1 \leq \sum_{i=1}^n \mathcal{M}_B(P_i) = \sum_{i=1}^n \mathcal{M}_B(x) \mathcal{M}_B(P_i)$.

- Case $\Phi = an$ then $\mathcal{M}_B(\Phi[x_i := P_i]) = \mathcal{M}_B(a[x_i := P_i]n[x_i := P_i]) = \mathcal{M}_B(a[x_i := P_i]) + \mathcal{M}_B(n[x_i := P_i]) \leq^{IH} \sum_{i=1}^{i=n} \mathcal{M}_B(a) \mathcal{M}_B(P_i) + \sum_{i=1}^{i=n} \mathcal{M}_B(n) \mathcal{M}_B(P_i) = \sum_{i=1}^{i=n} \mathcal{M}_B(an) \mathcal{M}_B(P_i)$.
- Case $\Phi = c(P'_1, \dots, P'_m)$ where c is external to B and $m \geq 1$ then $\mathcal{M}_B(\Phi[x_i := P_i]) = \mathcal{M}_B(c(P'_1, \dots, P'_m)[x_i := P_i]) = \mathcal{M}_B(c(P'_1[x_i := P_i], \dots, P'_m[x_i := P_i])) = \sum_{j=1}^{j=m} \mathcal{M}_B(P'_j[x_i := P_i]) \leq^{IH} \sum_{j=1}^{j=m} \sum_{i=1}^{i=n} \mathcal{M}_B(P'_j) \mathcal{M}_B(P_i) = \sum_{i=1}^{i=n} (\sum_{j=1}^{j=m} \mathcal{M}_B(P'_j)) \mathcal{M}_B(P_i) = \sum_{i=1}^{i=n} \mathcal{M}_B(c(P'_1, \dots, P'_m)) \mathcal{M}_B(P_i)$.
- Case $\Phi = c(P'_1, \dots, P'_m)$ where $m \geq 1$ and $\Gamma' \triangleright c(x_1, \dots, x_m) := u$ is a line in B then $\mathcal{M}_B(\Phi[x_i := P_i]) = \mathcal{M}_B(c(P'_1, \dots, P'_m)[x_i := P_i]) = \mathcal{M}_B(c(P'_1[x_i := P_i], \dots, P'_m[x_i := P_i])) = 1 + \sum_{j=1}^{j=m} \mathcal{M}_B(u) \mathcal{M}_B(P'_j[x_i := P_i]) \leq^{IH} 1 + \sum_{j=1}^{j=m} \mathcal{M}_B(u) \sum_{i=1}^{i=n} \mathcal{M}_B(P'_j) \mathcal{M}_B(P_i) \leq \sum_{i=1}^{i=n} \mathcal{M}_B(P_i) + \sum_{j=1}^{j=m} \mathcal{M}_B(u) \sum_{i=1}^{i=n} \mathcal{M}_B(P'_j) \mathcal{M}_B(P_i) = \sum_{i=1}^{i=n} \mathcal{M}_B(P_i) (1 + \sum_{j=1}^{j=m} \mathcal{M}_B(u) \sum_{i=1}^{i=n} \mathcal{M}_B(P'_j)) = \sum_{i=1}^{i=n} \mathcal{M}_B(P_i) \mathcal{M}_B(c(P'_1, \dots, P'_m))$. \(\square\)

LEMMA 4.39. *If $B \vdash \Phi \xrightarrow{\delta} \Phi'$ then $\mathcal{M}_B(\Phi) > \mathcal{M}_B(\Phi')$.*

Proof: By induction on $B \vdash \Phi \xrightarrow{\delta} \Phi'$.

- Case $B \vdash c(P_1, \dots, P_n) \xrightarrow{\delta} \Phi[x_i := P_i]$ then $\mathcal{M}_B(c(P_1, \dots, P_n)) = 1 + \sum_{i=1}^{i=n} \mathcal{M}_B(\Phi) \mathcal{M}_B(P_i) > \sum_{i=1}^{i=n} \mathcal{M}_B(\Phi) \mathcal{M}_B(P_i) \geq^{\text{Lemma 4.31}} \mathcal{M}_B(\Phi[x_i := P_i])$.
- The compatibility cases are a straightforward application of IH. \(\square\)

Finally, definition unfolding inside well-typed books is well-founded:

THEOREM 4.40 (Strong Normalisation). *Let $\vdash B :: \mathbf{B}$. For all subformulas Ψ occurring in B , relation $\xrightarrow{\delta}$ is strongly normalizing (i.e., definition unfolding inside a well-typed book is a well-founded procedure).*

Proof: This is a corollary of Lemma 4.39. \(\square\)

5. Examples

5.1. EXAMPLE 1

Our first example is a simple phrase, taken from a mathematical text:

[*1] the square root of the third power of a natural number

We give two possible translations 1.1 and 1.2 into WTT:

1.1: $\mathbf{Noun}_{x:\mathbb{R}} \exists_{n:\mathbb{N}} (x = \sqrt{n^3})$ 1.2: $\mathbf{Abst}_{n:\mathbb{N}}(\sqrt{n^3})$

Note that translation 1.1 is more informative in that it gives the final type of the noun (viz. \mathbb{R}), but that translation 1.2 is more compact. It is easy to verify that both translations return a weakly well-typed noun. We check this for translation 1.2. We start with a preface incorporating all external constants:

	constant name	weak type
(i)	3	$\mathcal{T} \rightarrow \mathcal{T}$
(ii)	$\sqrt{\quad}$	$\mathcal{T} \rightarrow \mathcal{T}$
(iii)	\mathbb{N}	\mathcal{S}
(iv)	Abst	$\mathcal{T} \rightarrow \mathcal{N}$

We also give the categories of the phrase in translation 1.2:

subexp	category		subexp	category		subexp	category
n	\mathcal{T}		n	\mathcal{T}		$\mathbf{Abst}_{n:\mathbb{N}}(\sqrt{n^3})$	\mathcal{N}
n^3	\mathcal{T}		\mathbb{N}	\mathcal{S}			
$\sqrt{n^3}$	\mathcal{T}		$n:\mathbb{N}$	\mathcal{Z}			

We need to derive $B; \Gamma \vdash \mathbf{Abst}_{n:\mathbb{N}}(\sqrt{n^3}) :: \mathcal{N}$ for some B and Γ . But it is clear that $B = \Gamma = \emptyset$. We assume that n belongs to the set V^T which we postulate as: (*) $n \in V^T$. The desired derivation is given in Figure 5.1 where numbers i , ii , iii and iv refer to the above preface. This derivation is given in the format of *forward reasoning*: we start with smaller subexpressions and build larger ones. The derivation can also be developed in the format of *backward reasoning*, i.e., in the *goal-directed* manner where we start with the *goal* (line (8)) and investigate how it can be reached. The only applicable rule to get line (8) is (*bind*). This gives new goals, generated by the main symbol (**Abst**) and (iv). These goals are the judgements in lines (1), (4) and (7), and so forth. Note that our derivation system is *syntax-driven* where for each goal, *only one* rule is applicable.

We vary a bit on this example and look at the following *statement*:

[*2] 8 is the square root of the third power of a natural number

(1)		$\vdash \mathbf{0} :: \mathbf{B}$	$(emp-book)$
(2)	$\mathbf{0}$	$\vdash \mathbf{0} :: \mathbf{\Gamma}$	$(emp-cont, 1)$
(3)	$\mathbf{0}; \mathbf{0}$	$\vdash \mathbb{N} :: \mathbb{S}$	$(ext-cons, 1, 2, iii)$
(4)	$\mathbf{0}$	$\vdash n : \mathbb{N} :: \mathbf{\Gamma}$	$(term-decl, 1, 2, 3, *)$
(5)	$\mathbf{0}; n : \mathbb{N}$	$\vdash n :: \mathcal{T}$	$(var, 1, 4, *)$
(6)	$\mathbf{0}; n : \mathbb{N}$	$\vdash n^3 :: \mathcal{T}$	$(ext-cons, 1, 4, i, 5)$
(7)	$\mathbf{0}; n : \mathbb{N}$	$\vdash \sqrt{n^3} :: \mathcal{T}$	$(ext-cons, 1, 4, ii, 6)$
(8)	$\mathbf{0}; \mathbf{0}$	$\vdash \mathbf{Abst}_{n:\mathbb{N}}(\sqrt{n^3}) :: \mathcal{N}$	$(bind, 1, 4, iv, 7)$

Figure 5. Derivation that $\mathbf{Abst}_{n:\mathbb{N}}(\sqrt{n^3})$ is a noun

For the translation, the easiest thing is to use our previous example, obtaining for translations 2.1 and 2.2 the relational statements:

$$2.1: \quad \mathbf{8} : \mathbf{Noun}_{x:\mathbb{R}} \exists_{n:\mathbb{N}}(x = \sqrt{n^3}) \qquad 2.2: \quad \mathbf{8} : \mathbf{Abst}_{n:\mathbb{N}}(\sqrt{n^3})$$

But in this case, there is a shorter and more elegant translation possible, viz. the logical statement *translation 2.3*: $\exists_{n:\mathbb{N}}(\sqrt{n^3} = 8)$

REMARK 5.1. *This example shows, that the ordinary mathematical formulas (so without our extension with nouns and adjectives) are often good enough for translations of a mathematical text in formal form. In the case above, translation 2.3 is quite satisfactory, albeit that translations 2.1 and 2.2 are, in a sense, closer to the original text. However, when definitions enter the stage, then the extension with nouns and adjectives, and hence with **Noun**, **Abst**, **Adj**, etc, is more appropriate. See Examples 5.2 and 5.3.*

Now look at [*3] below for which we can use 1.1 or 1.2 to get 3.1 and 3.2:

[*3] The square root of the third power of a natural number is positive

$$\text{Translation 3.1:} \quad \forall_{y:\mathbb{R}}(y : \mathbf{Noun}_{x:\mathbb{R}} \exists_{n:\mathbb{N}}(x = \sqrt{n^3}) \Rightarrow y \geq 0)$$

$$\text{Translation 3.2:} \quad \forall_{y:\mathbb{R}}(y : \mathbf{Abst}_{n:\mathbb{N}}(\sqrt{n^3}) \Rightarrow y \geq 0)$$

$$\text{Again, a more traditional translation is translation 3.3:} \quad \forall_{n:\mathbb{N}}(\sqrt{n^3} \geq 0)$$

REMARK 5.2. *The above examples show, that the translation of a mathematical text into WTT is not compositional. Look at the indefinite article a in the phrase [*1]: the square root of the third power of a natural number. Its translation is **Noun**, **Abst** or \exists , see 1.1, 1.2 and 1.3. When embedding [*1] into statement [*2], a again can play each of these three roles. However, when embedding it in statement [*3], none of these roles can be maintained: in all three cases 3.1, 3.2 and 3.3, the translation of a is changed into \forall .*

*This non-compositionality is present in several places in the translation process. The indefinite article a is an especially versatile word. It may have roles **Noun**, **Abst**, \exists or \forall , as shown above, but also other roles, in particular:*

the introduction of a variable in the context of a definition. For the latter, see Section 5.3. The versatility of the definite article *the* is hardly less.

5.2. EXAMPLE 2

Our second example concerns a text with a definition and its application:

DEFINITION A *Fermat-sum* is a natural number which is the sum of two squares of natural numbers.

LEMMA The product of a square and a Fermat-sum is a Fermat-sum.

A WTT-translation could be the following small WTT-book B of two lines (both with an empty context), one a definition and the other a statement. So the abstract format of B is: $\emptyset \triangleright D \circ \emptyset \triangleright S$:

$$\begin{aligned} a \text{ Fermat-sum} &:= \mathbf{Noun}_{n \in \mathbb{N}} \exists k \in \mathbb{N} \exists l \in \mathbb{N} (n = k^2 + l^2) \\ \forall u: a \text{ square} \forall v: a \text{ Fermat-sum} &(uv : a \text{ Fermat-sum}) \end{aligned}$$

Note how the defined constant *a Fermat-sum*, a noun, is used in the statement following the definition. The noun *a square* is not defined in B , hence the text assumes that this definition has been given beforehand: it is an external constant to B . Hence, it has to be incorporated in the preface when applying the derivation rules in order to establish $\vdash B :: \mathbf{B}$.

5.3. EXAMPLE 3

Our third example is from analysis. It contains the definitions of *difference quotient* and of *differentiable* and a statement using the latter definition:

DEFINITION. 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 $\lim_{h \rightarrow 0} \frac{f(a+h)-f(a)}{h}$ exists. The function $\sqrt{|x|}$ is not differentiable at 0.

Both definitions require a context. We use the *flag notation* to build the context in the translation of the above text into the following WTT-book:

- (1) $\sim A \subseteq \mathbb{R}$
- (2) $\sim f : A \rightarrow \mathbb{R}$
- (3) $\sim a \in A$
- (4) $\sim h \in \mathbb{R}$
- (5) $h \neq 0$
- (6) $a + h \in A$
- (7) $\text{the difference quotient of } f := \frac{f(a+h)-f(a)}{h}$
- (8) $f \text{ is differentiable at } a := \lim_{h \rightarrow 0} \frac{f(a+h)-f(a)}{h} \text{ exists}$
- (9) $\neg(\lambda_{x:\mathbb{R}}(\sqrt{|x|}) \text{ is differentiable at } 0)$

The flag notation is a shorthand for dealing with contexts: since (parts of) contexts are frequently repeated in succeeding lines, it saves space to allow multiple use of context entries (declarations and assumptions). The flag notation also enables the structure of a WTT-book to be more easily visible.

REMARK 5.3. *The flag notation is no more than sugaring, flags do neither exist in the WTT-syntax, nor in its derivation system. Flags provide for a certain view on WTT-texts which can be helpful for a human reader, e.g. when inspecting a formalized mathematical text in an electronic library.*

The above flag-text is the *sugared* version of the book given below. For convenience we first abbreviate:

$$\Gamma_1 = A \subseteq \mathbb{R}, f : A \rightarrow \mathbb{R}, a : A, h : \mathbb{R}, h \neq 0, a + h \in A$$

$$\Gamma_2 = A \subseteq \mathbb{R}, f : A \rightarrow \mathbb{R}, a : A$$

The book matching the above flag-text consists of the \circ -concatenation of the following three lines:

- $$\begin{array}{ll} \Gamma_1 & \triangleright \text{the difference quotient of } f := \frac{f(a+h)-f(a)}{h} \\ \Gamma_2 & \triangleright f \text{ is differentiable at } a := \lim_{h \rightarrow 0} \frac{f(a+h)-f(a)}{h} \text{ exists} \\ \emptyset & \triangleright \neg(\lambda_{x:\mathbb{R}}(\sqrt{|x|}) \text{ is differentiable at } 0) \end{array}$$

Note how context administration works in the flag notation. For example, the part $A \subseteq \mathbb{R}, a : A, f : A \rightarrow \mathbb{R}$ needs not to be repeated for the definition in line (8), since it is still *open* (the three flagpoles of lines (1) to (3) are still present in line (8)). We now discuss the details of the above WTT-book.

- Each context element is separately placed in a *flag*. The attached *flagpole* registers how long the context element is supposed to be present. Nesting of the flags fixes the order of the context elements.

- Context elements can be either declarations or assumptions. In order to visualize the difference, we start declarations with symbol \rightsquigarrow . (The symbol \rightsquigarrow *points at* the declared variable.)
- Note that the indefinite article *a* occurring in the noun *a function* in the original text, is translated here into the flag in line (2).
- In the WTT-book there are only *bound* variables. Notice for example that all free variables in line (8) are bound in the context lines (1) to (3).
- In translating the original text into (flag-style) WTT-format, we added the declaration of A , which was not explicit in the original text.
- $a \in A$ is translated as a declaration, whereas $a + h \in A$ is an assumption. (Both statements are treated similarly in the original mathematical text.)
- The context elements containing h are arbitrarily arranged in the original mathematical text. We concentrate these elements in lines (4) to (6) to use context administration smoothly: for the definition in line (8) we just skipped –i.e. cut the flagpoles of– the context elements (4) to (6). In fact, h is a bound variable in $\lim_{h \rightarrow 0} \frac{f(a+h)-f(a)}{h}$ and should not be *declared* in the context. Moreover, the assumptions $h \neq 0$ and $a + h \in A$ are not desired for the definition in line (8). It is important to find a *minimal* context for a definition or statement.
- In line (7) we have a *phrase definition* of the form $\mathfrak{C}^{\mathcal{T}}(\vec{V}) := \mathcal{T}$. The parameter list of the constant *the difference quotient* is (A, f, a, h) , but only f is accounted for in the WTT-book. A better formulation is: *the difference quotient of f in a with difference h (where f has domain A)*.
- The definition in line (8) is a *statement definition*, corresponding to $\mathfrak{C}^{\mathcal{S}}(\vec{V}) := \mathcal{S}$. Of the parameter list (A, f, a) of the constant *is differentiable at*, we only find f and a in the book. The definition can also be given as a phrase definition (namely as the definition of the adjective *differentiable*), in the format $\mathfrak{C}^{\mathcal{A}}(\vec{V}) := \mathcal{A}$. This case is studied below.
- The limit-binder $\lim_{h \rightarrow p}$ can be considered to be a (non-binding!) constant of three variables, $\lim(p, X, g)$, defined beforehand in a context consisting of the three declarations $p \in \mathbb{R}, X \subseteq \mathbb{R}, g : X \rightarrow \mathbb{R}$. That is, we consider $\lim_{h \rightarrow p} g(h)$ to be an alternative notation for $\lim(p, X, g)$, with X the domain of function g . In line (8) we have the *instantiation* $\lim(0, X, \lambda_{h \in X} \frac{f(a+h)-f(a)}{h})$, with $X = \{x \in \mathbb{R} \mid a + x \in A \wedge x \neq 0\}$.
- The *existence* of the limit as required in line (8) should also have been defined beforehand, in a piece of mathematical theory stating:

(1) The definition of the *limit-property* of a function $g : X \rightarrow \mathbb{R}$ with respect to a point p on the real x -axis. This property is given as an existential statement (with ε 's and δ 's), e.g. in the statement definition:

$$g \text{ has the limit-property in } p := \exists l \in \mathbb{R} \forall \varepsilon \in \mathbb{R}^+ \exists \delta \in \mathbb{R}^+ \forall x \in X \setminus \{p\} (|x - p| < \delta \Rightarrow |g(x) - l| < \varepsilon)^{20}$$

(2) The theorem that *if* the limit-property holds, the existing l is unique.

(3) The definition of $\lim(p, X, g)$ as being this unique l , again under the assumption that the limit-property holds.

Hence $\lim(p, X, g)$ exists is $g : X \rightarrow \mathbb{R}$ has the limit-property in p .

- In line (9), parameters A , f and a of the statement *f is differentiable at a* are instantiated with \mathbb{R} , $\lambda_{x:\mathbb{R}}(\sqrt{|x|})$ and 0, respectively. The resulting statement (and its negation with the logical constant \neg) needs no context.

We give an alternative translation where *differentiable (at)* is defined as an adjective. Since $f : A \rightarrow \mathbb{R}$ becomes a subscript declaration, it should be left out of the context for statement (8'). (Lines (1) to (7) are the same as before, in line (9') we employ the typing symbol “.” instead of the verb “is”.)

$$\begin{array}{l} (1) \quad \boxed{\sim A \subseteq \mathbb{R}} \\ (2) \quad \boxed{\sim f : A \rightarrow \mathbb{R}} \\ \quad \quad \quad \vdots \\ (8') \quad \boxed{\text{differentiable at } a := \text{Adj}_{f:A \rightarrow \mathbb{R}}(\lim_{h \rightarrow 0} \frac{f(a+h) - f(a)}{h} \text{ exists})} \\ (9') \quad \neg(\lambda_{x:\mathbb{R}}(\sqrt{|x|}) : \text{differentiable at } 0) \end{array}$$

Derivations leading to either of the two WTT-books given above, need many small steps, but can be constructed straightforwardly. We omit the derivations. We only give a number of remarks regarding these derivations:

- In line (1) of both WTT-books we type A by declaring it to be a *subset* of \mathbb{R} . This is not according to the rules. However, we may consider the *declaration* $A \subseteq \mathbb{R}$ to be shorthand for the declaration $A : \text{SET}$ followed by the *assumption* $A \subseteq \mathbb{R}$. Another option is to rewrite $A \subseteq \mathbb{R}$ as $A : \wp(\mathbb{R})$.
- Equality, addition, subtraction and division should also obtain weak types in the preface. Their common weak type is $\mathcal{T} \times \mathcal{T} \rightarrow \mathcal{T}$. The weak type of $\sqrt{}$ is $\mathcal{T} \rightarrow \mathcal{T}$. The logical operator \neg has weak type $\mathcal{S} \rightarrow \mathcal{S}$.
- *Function application* as in $f(a)$ and $f(a+h)$ can be treated as a binary external constant `app1` with weak type $\mathcal{T} \times \mathcal{T} \rightarrow \mathcal{T}$.

²⁰ This is a statement definition $\mathcal{C}^S(\vec{v}) := \mathcal{S}$. It needs a context to declare X , p and g .

- In lines (9) and (9') we use an internally defined notion (*differentiable*). In both translations, either as part of a constant for a statement or as a constant for an adjective, we can apply rule (*int-cons*) for establishing well-typedness.

5.4. EXAMPLE 4

Finally we give an example from elementary algebra. We consider the following theorem together with its proof:

THEOREM. Let G be a set with a binary operation \cdot and left unit element e . Let H be a set with binary operation $*$ and assume that φ is a homomorphism of G onto H . Then H has a left unit element as well.

PROOF. Take $e' = \varphi(e)$. Let $h \in H$. There is $g \in G$ such that $\varphi(g) = h$. Then $e' * h = \varphi(e) * \varphi(g) = \varphi(e \cdot g) = \varphi(g) = h$, hence e' is left unit element of H . \square

A translation into WTT is:

- | | |
|------|--|
| (1) | $\sim G : \text{SET}$ |
| (2) | $\sim \cdot : (G \times G) \rightarrow G$ |
| (3) | $\sim e : G$ |
| (4) | $e : \text{a left unit element of } G$ |
| (5) | $\sim H : \text{SET}$ |
| (6) | $\sim * : (H \times H) \rightarrow H$ |
| (7) | $\sim \varphi : G \rightarrow H$ |
| (8) | $\varphi : \text{a surjective homomorphism}$ |
| (9) | $e' := \varphi(e)$ |
| (10) | $\sim h : H$ |
| (11) | $\exists_{g \in G} (\varphi(g) = h)$ |
| (12) | $\sim g : G$ |
| (13) | $\varphi(g) = h$ |
| (14) | $e' * h = \varphi(e) * \varphi(g) = \varphi(e \cdot g) = \varphi(g) = h$ |
| (15) | $e' * h = h$ |
| (16) | $e' : \text{a left unit element of } H$ |
| (17) | $H \text{ has a left unit element}$ |

This book is a good example of the gains due to flags. The book consists of six lines with contexts overlapping largely. Schematically, the lines are:

- (1) – (8) ▷ (9)
- (1) – (8), (10) ▷ (11)
- (1) – (8), (10), (12), (13) ▷ (14)
- (1) – (8), (10) ▷ (15)
- (1) – (8) ▷ (16)
- (1) – (8) ▷ (17)

The first of these lines is a definition, the second to fifth are intermediate results (statements), being part of the proof, and the last line expresses the theorem (a statement, as well). We translated the proof first, and put the theorem at the end in order to facilitate a further translation into type theory (this is not necessary for the translation into WTT). We note the following:

- Lines (1) to (4) could be concentrated in the single declaration (extending WTT slightly):

$$(1') \quad \boxed{\rightsquigarrow (G, \cdot, e) : \text{a groupoid with left unit element}}$$

and something similar for lines (5) and (6). WTT (or type theory!) in practice often *asks for* such a kind of abbreviations for dependent parts of a context (also called *telescopes*, see [23]).

- Not all parameters are accounted for in the WTT-book. E.g., one is inclined to make lines (4) and (8) more specific in the following manner:

(4) e : a left unit element of G with respect to \cdot ,

(8) φ : a homomorphism of G onto H with respect to \cdot and $*$, resp.

- We left out the parameter list for the newly defined constant ℓ in line (9). As we said before, this list can be reconstructed since it is equal to the list of the declared variables occurring in the context of the definition. Hence, the *official* format of line (9) can be: $\ell(G, \cdot, e, H, *, \varphi) := \varphi(e)$.

Moreover, in the subsequent *uses* of the constant ℓ in lines (14) to (16), there should be parameter lists as well behind each occurrence of constant ℓ . In this case, however, the instantiations for the variables in the variable lists of ℓ in lines (14) to (16) are *exactly the same* as in the definition itself (so variable G is instantiated with G , variable \cdot with \cdot , etc.). This shows once more that it can be very economical to allow a shorter notation for parameter lists such that reconstructable or unchanged heads of parameter lists may be omitted. This is only sugar and can always be undone. (In Automath [7] this is a syntactic feature.)

- The context for the definition $e' := \varphi(e)$ in line (9) of the example is larger than necessary. In fact, we could do with context $G : \text{SET}, e : G, H : \text{SET}, \varphi : G \rightarrow H$ and corresponding parameter list (G, e, H, φ) . In our example, however, we keep close to the original CML-text, in which the local definition $e' = \varphi(e)$ is made in the full context of the theorem.
- Lines (12) and (13) are a direct consequence of our wish to *avoid free variables* in a WTT-book. Note that g is a *bound* variable in line (11), but without line (12) it would be a *free variable* in line (14).

It is, in fact, a free variable in the original mathematical text. This is due to the habit in mathematics to extend the scope of an existentially bound variable outside the formula in which it is introduced:

*There is $g \in G$ such that $\varphi(g) = h$. Then ... $\varphi(e) * \varphi(g) = \varphi(e \cdot g) = \varphi(g) = \dots$*

The occurrences of g after *Then. ...* are free! What is actually happening in such cases is that the *existence* of such a g automatically induces the (silent) introduction of such a g (for convenience *called g* again) outside the scope of the \exists -binder. Repair of this habit is straightforward, by using the corresponding logical rule of \exists -elimination:

$$\frac{\exists_{x \in U}(P(x)), \forall_{x \in U}(P(x) \Rightarrow r)}{r}$$

This rule is the background for lines (12) and (13), since lines (12) to (14) prove that $\forall_{g \in G}(\varphi(g) = h \Rightarrow e' * h = h)$, which, together with (11) and \exists -elimination justifies the conclusion (15). Hence, the apparent *detour* via lines (12) to (14) is necessary in order to mend a frequent short cut in mathematical texts. This complication is not the fault of our translation.

- Line (14) is a chain of statements: there are as many statements in this line as there are =-signs, and maybe even one more: the *implicit conclusion* $e' * h = h$, which is repeated in (15) (but in a smaller context).
- The translation is as near as possible to the original text (but for the mending of the consequences of the existential quantifier, see above). This implies that intermediate results which one would expect in the formal version, are nevertheless left out. E.g., one could expect the line $\forall_{h \in H}(e' * h = h)$ preceding line (16), in the same context (1) – (8).²¹
- Note that we *omit all justifications* in WTT-books. Our reason for doing this is, that we want to have a rather simple syntax for WTT, which is

²¹ This is the logical consequence of lines (10) and (15), due to the \forall -introduction rule.

neither concerned with *meta-arguments* about the logical or mathematical correctness, nor with the interdependence of statements induced by these correctness arguments. Hence, we find no words like *Since*, *Hence* or *Because* in WTT. The drawback of this is, that the argumentation structure of a book can become unclear. This is mended when we make the next step: translating WTT into type theory.

However, as we said in Remark 3.6, we may choose instead to add to the WTT-book above, two columns, one with labels and one with comments. In the *label* column, to the left of the sample WTT-text, we could write

- At line (9): *Proof start* and *Definition*,
- At line (16): *Proof end*,
- At line (17): *Theorem*.

In the *comments* column, to the right of the text, we could write

- At line (11): *From (8) or Because of (7), (8) and (10)*,
- At line (15): \exists -*elimination on (11) and (12) to (14)*,
- At line (16): *From (10) to (15), \forall -introduction and definition of left unit element*.

We leave it as an easy exercise to give a derivation of the WTT-book above.

6. Final remarks

Presently, there is a great variety of proof checkers and theorem provers. We mention Automath [7], Mizar [17], NuPrl [5], Coq [2], Isabelle [15] Ω MEGA [3] and PVS [18]. These systems provide help for the users, e.g. by offering a friendly user interface or by enabling the use of *tactics*. These tactics are special assignments to the computer, in order to simplify or develop the actual *proof goal* during the construction or check of a proof. Some of these systems are based on a fully formalized language for mathematics. Others like PVS only provide tactics to the user, a completed proof is not a text with mathematical content, but only a listing of the tactics used. Some theorem provers offer a mathematical vernacular in the WTT-sense, i.e. an incomplete, textual rendering of a mathematical content which has resemblance with informal proofs. Below we discuss some of these.

- The theorem proving system Coq [2] incorporates in its documentation a kind of intermediate specification language, called *Gallina*²². This

²² See <http://coq.inria.fr/doc/node.0.0.html>.

vernacular is a formally defined language meant for the development of mathematical theories and to prove specifications of programs. The intention is that it is usable as a *language of commands* for Coq, helping the Coq-user to stay closer to *normal* intuition when proving a mathematical proposition. It is, however, not meant as a *first step in formalization* as WTT is. It has a rather specific form which does only distantly reflect a mathematical discourse and it is also not very adequate for the purposes exposed in this paper.

- The mathematical vernacular proposed in [9] relies on Coq. It provides instructive *labels* (like Axiom, Definition, Hypothesis, Statement, Proof). Moreover, a number of natural deduction rules are replaced with more intuitive alternatives (e.g., \exists -elimination). The aims are clearly different from ours: labels as the ones above are out of scope in our formalisation and so are *comments* about the (e.g. logical) structure, as is discussed in Remark 3.6. Moreover, since WTT is not concerned with validity, the proof structure itself (e.g., natural deduction) has no formal counterpart in our system. Consequently, a Proof Synthesis Algorithm as in [10] has no direct application in WTT. However, when WTT is translated into more complete formal languages (see Section 6.1), the ideas of Dowek will most probably be fruitful and inspiring.
- The aim of the project Ω MEGA [3] is to develop a software environment for the support of a scala of theorem provers. The built-in version of the *mathematical vernacular* is meant to give the user on request a mathematics-like computer *view* of an already checked proof. It has the same drawbacks as the mathematical vernacular of Coq.
- The basic languages of Mizar [17] and Isar [20] are close to the reliability criterion and have proven to be suited for expressing large corpora of mathematical content. Their syntax is, however, rather complicated and requires much of an ordinary user to become acquainted with it. [20] compares both Isar and Mizar listing their weak and strong points.
- In the *Theorema project* [4] computer algebra systems are extended with facilities for mathematical proving. The provers are designed to imitate the proof style humans employ in their proving attempts. The proofs can be produced in human-readable style. However, this is done by *post-processing* a formal proof in natural language. This deviates from our approach, for which no formal proof needs to be present. The natural language part of Theorema has little structure and restricts itself to comments on the logical steps employed (a part that does not appear in WTT). The linguistic facilities of WTT are absent in Theorema's natural language. Moreover, the text style of Theorema (insofar as we could see) is not based on a formal grammar for the textual language.

- In [16], a large amount of work has been done on a typed functional programming language GF whose purpose is to define languages such as fragments of natural languages, programming languages and formal calculi. GF is an extension of logical frameworks which are implementations of type theory. GF is based on Martin-Löf’s type theory. Our work is different and complementary to that of Ranta. We do not at all assume/prefer one type theory instead of another. WTT is completely independent from any particular type theory. We believe that the formalisation of a language of mathematics should separate the questions of *which type theory is necessary for which part of mathematics* and *which language should mathematics be written in*. Moreover, mathematicians don’t usually know or work with type theories. Mathematicians usually *do* mathematics (manipulations, calculations, etc), but are not interested in general in reasoning *about* mathematics.

As far as we know, no theorem provers provide an independent language for describing mathematical content in such a manner that the reliability criterion is sufficiently accounted for. Existing mathematical vernaculars are moreover (to our knowledge) not ready for immediate use, and if accessible for a mathematical user, then with great difficulty.²³

6.1. FUTURE WORK

We list a number of items concerning possible future work.

- The syntax and derivation rules of WTT must be *tested* on a corpus of mathematical texts from various areas in mathematics and with considerable size. Former tests (cf. Section 1), are not enough to allow conclusions about WTT. Forthcoming tests should enlist a potential *users group*, ranging from students to all sorts of mathematicians: both in the theoretical and applied field, and working in either teaching or research.
- In this testing stage, it should be considered whether certain forms of *sugaring* can safely be added to WTT. This sugaring is probably desirable in order to make WTT an acceptable language tool for the users. For example, the possibility of infix notation makes the text more user-friendly. This also holds for other sugaring devices, such as the possibility of omitting empty parameter lists.

Another possible sugaring which may be advantageous is the use of *flags* as employed in Section 5. In the examples of that section, the benefits of the flag notation were explained.

²³ See <http://www.cs.kun.nl/~freek/digimath/bycategory.html> for an extensive overview of systems implementing mathematics in the computer.

For enabling a consistent use of certain forms of sugaring, the syntax of WTT must be extended. It may be preferable to do this in a kind of *shell*, built on top of WTT, to maintain the reliability criterion for WTT. The development of such a shell is user-driven and may require arbitrary decisions, without much coherence. In this respect, the sugaring-shell differs from the underlying WTT syntax, which has a tighter structure and a more mathematically-driven motivation.

- It is desirable to *implement* WTT. A parser should be able to parse WTT-expressions and to check whether such expressions obey to the grammar of Section 2. Next, a (grammatical) type checker must be implemented which establishes the weakly well-typedness of contexts, books etc. according to the derivation rules of Section 3. User-friendliness of such a type checker is of importance, in order to make the tool acceptable for mathematicians who wish to write (or translate) their text in(to) WTT. This may ask for things as *flags*, but also for other aids such as pop-up windows, appropriate input-output handling, interactive communication, possible storage and retrieval of texts and contexts.
- It is interesting to investigate how WTT can be *enriched* in the direction of either type theory, or another acceptable complete mathematical language, such as that of Zermelo-Fraenkel Set Theory. A first step in that direction is to incorporate labels, proofs and proof methods. This encapsulates the *label* and *comments* columns mentioned in Remark 3.6.
- In this stage of the research, it is conceivable that *a chain of intermediate languages* between WTT and – for example – full fledged type theory is the best manner to bridge the gap. In this case, translation protocols should be devised for each link of the chain. For different transitions in this translation process, different specialists may be the preferred executives: mathematicians, computer scientists or type theorists. It may be worth while to compare this research with work on (hierarchies of) specification languages as has been done in computer science.
- It should be investigated how *computer assistance* can be invoked in either the full transition process or in one of the translation stages from WTT to a completely formalized version of a mathematical text.
- Finally, it can be investigated how the results obtained can be made profitable for the community of mathematicians, both in developing and in using mathematics, in several degrees of precision. For example, easy access to WTT technology can be useful for computer help in writing mathematics. On the other side of the spectrum, we have the complete formalization of a certain text, which is suitable for a complete check

on correctness and subsequently for storage of correct mathematical knowledge in a – publicly accessible – data base.

6.2. CONCLUSION

The famous mathematician Frege was frustrated by the informalities of the common mathematical language CML: . . . *I found the inadequacy of language to be an obstacle; no matter how unwieldy the expressions I was ready to accept, I was less and less able, as the relations became more and more complex, to attain precision...* (*Begriffsschrift*, Preface, see [11]). In 1879, he wrote the *Begriffsschrift* (see [11]), whose *first purpose is to provide us with the most reliable test of the validity of a chain of inferences* (again, see *Begriffsschrift*, Preface). Then he wrote the *Grundlagen und Grundgesetze der Arithmetik* [11] where he argued that mathematics is a branch of logic and described arithmetic in *Begriffsschrift*. Russell wrote a letter to Frege [11] informing him of a paradox in Frege's work and his own (see [12]). To avoid the paradox, Russell used *type theory* in the famous *Principia Mathematica* [21] where mathematics was founded on logic. Advances were also made in set theory [22], category theory [13], etc., each being advocated as a better foundation for mathematics. But, none of the logical languages of the 20th century satisfies the criteria expected of a language of mathematics. A logical language does not have mathematico-linguistic categories, is not universal to all users of mathematics, and is not a satisfactory communication medium:

- Logical languages make fixed choices (first versus higher order, predicative versus impredicative, constructive versus classical, types or sets, etc.). But different parts of mathematics need different choices and there is no universal agreement as to which is the best formalism.
- A logician writes in logic their understanding of a mathematical-text as a formal, complete text which is structured considerably unlike the original, and is of little use to the *ordinary* mathematician.
- Mathematicians do not want to use formal logic and have for centuries done mathematics without it.

So, mathematicians kept to CML. In this paper, we gave WTT, an alternative to CML which avoids some of the features of the logical languages which made them unattractive to mathematicians. We hope that WTT will open a new useful era of collaboration between mathematicians and logicians:

- WTT- and CML-texts are related by the reliability criterion (a WTT-text covers what its CML-version intended). A mathematician can check this.

- Although both the CML-text and its translation into WTT are incomplete, WTT has additional *levels* supporting more rigor. One can define further translations into more and more logically-complete versions. Since these translations are from formal to formal texts, it is easy to check reliability between a text at level $_i$ and its more complete version at level $_{i+1}$.
- As the path from a CML- to a logically-complete text is divided in clearly connected phases, it can be built with or without the help of the mathematician.

The above bridging between mathematics and logic can also reach computer science and proof checking. In 1967 the famous mathematician de Bruijn began work on logical languages for complete books of mathematics that can be checked by machine. People are prone to error, so if a machine can do proof checking, we expect fewer errors. Most mathematicians doubted de Bruijn could achieve success, and computer scientists had no interest at all. However, he persevered and built Automath [14] (AUTOMated MATHe-matics). Today, there is much interest in many approaches to proof checking for verification of computer hardware and software. Many theorem provers have been built to mechanically check mathematics and computer science reasoning (e.g. Isabelle, HOL, Coq, etc.). In practice, a CML-text is structured very differently from a computer-checked text proving the same facts. Making the latter involves extensive knowledge and many choices:

- First, the needed choices include:
 - The choice of the underlying logical system.
 - The choice of how concepts are implemented (equational reasoning, equivalences and classes, partial functions, induction, etc.).
 - The choice of the formal system: a type theory (dependent?), a set theory (ZF? FM?), etc.
 - The choice of the proof checker: Automath [14], Isabelle [15], Coq [2], PVS [18], Mizar [17], etc.
- Any informal reasoning in a CML-text will cause headaches as it is hard²⁴ to turn a big step into a (series of) syntactic proof expressions.
- Then the CML-text is *reformulated* in a fully *complete* syntactic formalism where every detail is spelled out. Very long expressions replace a clear CML-text. The new text is useless to ordinary mathematicians.

Thus, automation is user-unfriendly for the ordinary mathematician/computer scientist. It is the hope that WTT may help in dividing the jump from informal mathematics to a fully formal one into smaller more informed steps.

²⁴ *Tactics* help but give a *track* for the final proof which is not informative nor accessible.

References

1. Bancerek, G.: 2003, 'On the Structure of Mizar Types'. *ENTCS* **85.7**.
2. Barras, B. et al.: 1999, 'The Coq Proof Assistant, Reference Manual'. INRIA.
3. Benzmüller, C. and M. Kohlhase: 1997, 'ΩMEGA: Towards a Mathematical Assistant'. In: W. McCune (ed.): *Proceedings of Conference on Automated Deduction (CADE-14)*, Vol. 1249 of *Lecture Notes in Artificial Intelligence*. pp. 252–255.
4. Buchberger, B. et al.: 1997, 'A survey of the Theorema project'. In: *Proceedings of ISSAC'97 (International Symposium on Symbolic and Algebraic Computation)*, Maui, Hawaii (1997). pp. 384–391.
5. Constable, R. L. et al.: 1986, *Implementing Mathematics with the Nuprl Proof Development System*. Prentice Hall.
6. de Bruijn, N.: 1990, *Reflections on Automath*. Eindhoven University of Technology. Also in [14], pages 201–228.
7. de Bruijn, N. G.: 1970, 'The mathematical language Automath, its usage and some of its extensions'. In: *Proceedings of Symposium on Automatic Demonstration*, Vol. 125 of *Lecture Notes in Mathematics*. pp. 29–61. Also in [14], pages 73–100.
8. de Bruijn, N. G.: 1979/1980, 'Grammatica van WOT'. *Euclides* **55**, 66–72.
9. Dowek, G.: 1990a, 'Naming and scoping in a mathematical vernacular'. Technical Report 1283, INRIA (Institut National de Recherche en Informatique et en Automatique, Rocquencourt).
10. Dowek, G.: 1990b, 'A Proof Synthesis Algorithm for a Mathematical Vernacular'. In: G. Huet and G. Plotkin (eds.): *Proceedings of the First Workshop on Logical Frameworks, Antibes, France*.
11. Heijenoort, J. v. (ed.): 1967, *From Frege to Gödel: A Source Book in Mathematical Logic, 1879–1931*. Cambridge, Massachusetts: Harvard University Press.
12. Kamareddine, F., L. Laan, and R. Nederpelt: 2002, 'Types in logic and mathematics before 1940'. *Bulletin of Symbolic Logic* **8**(2), 185–245.
13. MacLane, S.: 1972, *Categories for the Working Mathematician*. Springer.
14. Nederpelt, R. P., J. H. Geuvers, and R. C. de Vrijer: 1994, *Selected papers on Automath*. Amsterdam: North-Holland.
15. Nipkow, T., L. Paulson, and M. Wenzel: 2002, *Isabelle/HOL, A proof assistant for higher-order logic*. Springer, LNCS 2283.
16. Ranta, A.: 2001, 'The GF language: Syntax and Type System'. Technical report, Chalmers. <http://www.cs.chalmers.se/~aarne/GF>.
17. Rudnicki, P.: 1992, 'An overview of the MIZAR project'. In: B. Nordström, K. Pettersson, and G. Plotkin (eds.): *Proceedings of the 1992 Workshop on Types for Proofs and Programs, Båstad*. pp. 311–332.
18. S. Owre, J. R. and N. Shankar: 1992, 'PVS: A prototype verification system'. In: D. Kapur (ed.): *Lecture Notes in Artificial Intelligence*, Vol. 607. pp. 748–752.
19. Severi, P. and E. Poll: 1994, 'Pure Type Systems with Definitions'. In: A. Nerode and Y. Matiyasevich (eds.): *Proceedings of LFCS'94 (LNCS 813)*. New York, pp. 316–328.
20. Wenzel, M. and F. Wiedijk: 2002, 'A comparison of Mizar and Isar'. *Automated Reasoning* **29**, 389–411.
21. Whitehead, A. and B. Russell: 1910¹, 1927², *Principia Mathematica*, Vol. I, II, III. Cambridge University Press.
22. Zermelo, E.: 1908, 'Untersuchungen über die Grundlagen der Mengenlehre'. *Math. Annalen* **65**, 261–281.
23. Zucker, J.: 1975, 'Formalization of classical mathematics in Automath'. In: *Colloque Internationale de Logique, Clermont-Ferrand, France*. pp. 135–145.

