
ELSEVIER

Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®

Electronic Notes in Theoretical Computer Science

Electronic Notes in Theoretical Computer Science 93 (2004) 102–117
www.elsevier.com/locate/entcs

Faithfully Reflecting the Structure of Informal Mathematical Proofs into Formal Type Theories

G.I. Jojgov, R.P. Nederpelt, M. Scheffer[1]

Eindhoven University of Technology
The Netherlands

Abstract

Mathematical proofs, as written in every-day Common Mathematical Language (CML), are informal and many details are left implicit. To check such proofs with a proof assistant they need to be formalized and elaborated to full detail. In order to reduce the possibility of formalization errors and therefore increase the reliability of the translation of CML texts into type theories, we use a version of Nederpelt's formal language WTT extended with logical notation that encodes the natural deduction proof steps. By using this intermediate version, the subsequent translation into a full-fledged type theory can be made such that the proof clearly reflects the structure of the original CML proof. This makes it easier to ensure that the formalization of the CML text is done correctly, and offers additional advantages over usual representations of proof terms in type theory.

Keywords: weak type theory, type theory, proof representation, formalization

1 Introduction

Proof assistants are interactive computer programs that are used to create, check and store formal theories in fields that may range from strictly theoretical as foundations of mathematics, to practical applications as proving correctness of software or hardware systems. Given a text in its formal language, the proof assistant is capable of checking its correctness with respect to a set of rules. The role of the human in this process is to provide the

[1] Email: G.I.Jojgov@tue.nl R.P.Nederpelt@tue.nl M.Scheffer@tue.nl

doi:10.1016/j.entcs.2003.12.030

formalization and to guide the proof assistant during the proof construction process.

One can ask how reliable such a method is for doing mathematics with a proof assistant. To answer this question we need to look at the two main phases in the translation of an informal proof (in the mind of a mathematician) to its fully formalized and computer-checked version (see Figure 1). At the first stage we have to phrase the informal proof/theory in the language of the prover. Then at the second stage, using the prover we create and/or check the formal proof.

The famous *de Bruijn criterion* states that in order to have a maximum reliability in the second phase, we need to have a small, manually-verifiable proof-checker to control the proofs. The proof-checker is independent of the possibly very complex machinery that actually constructs those proofs. Therefore, by using a proof assistant that conforms to the criterion of de Bruijn we can achieve a fairly high degree of reliability.

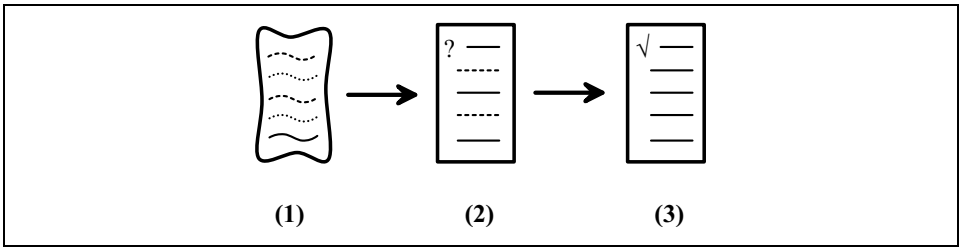

Fig. 1. The two stages of the formalization process. The initial formalization translates the informal theory (1) written in CML (e.g. an article in LaTeX) into a formulation in a formal system (2) (e.g. a WTT text) without checking logical consistency. At the second stage the proofs are created or refined if necessary. The result is a formally verified theory (3) (e.g. in the Calculus of Constructions)

As shown by practical experiences with proof assistants, the reliability of the initial formalization of the informal theory is at least as important. The problem in this case is however that there is no way to check that a formal statement correctly reflects the informal ideas that the author of the text had. In [8] the following approach to this problem is proposed: In order to increase the reliability of the initial translation of informal ideas into a formal system, introduce a new formal system that is as close as possible to the informal common mathematical language (CML). By providing a low-threshold initial formalization, we shorten the distance between the informal and the formal versions of a theory and therefore reduce the risk of making 'undetectable' formalization errors. The formal system proposed in [8] (and first introduced in [5]) is called Weak Type Theory (WTT). It imposes very little constraints (hence 'weak') on the formal text and mainly focuses on ensuring linguistic coherence.

On the other hand, proofs encoded in Type Theory are subject to tight typing rules and a well-typed term inhabiting a propositional type is an encoding of a fully formalized proof of that proposition. Since many interactive theorem provers (like Coq [9]) use the procedurally-oriented tactics-based approach, the proof terms constructed by these systems are often unreadable and the structure of the original proof is not visible.

The aim of this paper is to show by means of a tiny case study that the possibility to reflect the logical reasoning (expressed by natural deduction steps) and the possibility to reflect the informal discourse of a CML proof are actually orthogonal. We show how a mathematical proof written in CML can be presented formally in a version of WTT, extended with notation for natural deduction steps and subsequently translated into a full-fledged type theory. We will show that both versions of the proof *reflect the structure* of the original CML proof and remain very close to it. This means that it is easier for the author to make sure that his informal ideas are correctly represented in the formal version. We use a fairly standard extension of type theory (namely adding *global definitions*) to encode the formal WTT proof in type theory.

In comparison with the 'standard' approach of encoding proofs as single λ-terms we see significant gains in e.g. the readability of the proof, the possibilities for its maintenance (i.e. dealing with changes in the original proof) and therefore the overall reliability of the formalization.

The contribution of this paper should be seen not in the introduction of new systems but rather in the way we can use existing ones (WTT, type-theory) to address the problems of reliability of formalization in type-theory based theorem provers. In particular, we show how the structure of informal proofs written in a WTT version extended with notation for natural deduction (ND) steps can be maintained in existing type systems.

In the next section we first give a short introduction to WTT and a version of WTT extended with logical notation (called WTT_L). In Section 3 we introduce the extensions that we make to the Calculus of Constructions (λ_C) in order to maintain the structure of the WTT_L proofs. Subsequently in Section 4 we give a sample translation of a CML proof into type theory via WTT_L and compare it with the traditional encoding in type theory. We end with a brief discussion of related work in Section 5 and conclusions in Section 6.

2 WTT and a logical extension to WTT

It is beyond the scope of this article to fully introduce Weak Type Theory, a language to express mathematical texts; here we limit ourselves to giving a first impression of the language. WTT was first introduced in [5] and is a

refinement of de Bruijn's Mathematical Vernacular [1] (MV).

- 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 important desirable properties of the WTT language such as strong normalisation, decidability of type checking and subject reduction (with respect to δ-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.
- is independent of any underlying formalism, such as type theory, set theory or category theory.

The syntax of WTT is based on linguistic categories, that form the (weak) types of the WTT terms. We mention the linguistic categories of WTT without going into details:

- on the atomic level: variables, constants and binders
- on the phrase level: terms, sets, nouns and adjectives
- on the sentence level: statements and definitions
- on the discourse level: contexts, lines and books

A WTT book represents a mathematical proof and consists of a sequence of lines, each being either a statement or a definition (in a certain context). The context 'sets the stage' for a statement/definition and is itself a list of statements (being either assumptions or typing statements, also known as declarations). On its turn, statements and definitions are build up from smaller

linguistic categories at the phrase and atomic level, among which terms, sets, nouns, binders, etc.

$$x : \mathbb{R}, x \geq 0 \rhd \sqrt{x} := \iota_{y:\mathbb{R}}(y \geq 0 \wedge y^2 = x)$$
$$x : \mathbb{N}, x = 2 \rhd \exists_{n:\mathbb{N}}(\sqrt{n^3} = x + 6)$$

Fig. 2. Example of a definition line and a statement line of a WTT book. The binder ι in the definition of $\sqrt{x}$ is Russel's definite descriptor. The expression $\iota_{y:\mathbb{R}}(\varphi(y))$ should be read as 'the real number y for which $\varphi(y)$ holds.

Figure 2 shows an example of a definition line and a statement line in WTT. The contexts consist of a declaration of a variable x and assumption it and is separated from the actual statement of the line by a symbol '$\rhd$'. The statement of the line is on its turn build up from constants like $\sqrt{\ }$, $=$ and $+$, variables x and n, and from the binder $\exists$.

The abstract syntax of WTT, which we do not give here, establishes the well-formedness conditions for these linguistic categories. All constructs obtained with the derivation system have a weak type, which corresponds to a linguistic category. The derivation rules, for which we also refer to [5], only give a subset of the well-formed constructs obtained with the abstract syntax, since the rules enforce that those constructs obey certain (weak) typing requirements.

The derivation system is syntax-driven, in the sense that for each sub(goal) in a derivation, only one rule is applicable. A book constructed with our derivation system is transparently structured and in general less ambiguous than CML texts. The derivation rules for example enforce that all parameters of a constant that is used are spelled out and that there are no free variables in a book.

WTT can be used as a starting point in the translation of mathematical texts into type theory. We show how the use of a WTT version extended with logic (i.e. WTT_L) can be used to obtain a type theoretical translation of CML texts that clearly preserve the structure. Thereto we have to add definitions for all logical operators and rules to the context of the judgement in type theory.

Examples

We consider an example translation into WTT of a proof from mathematical analysis. It contains the definitions of 'difference quotient' and of 'differentiable' and a statement using the latter definition:

For convenience we first abbreviate the following contexts by Γ_1 and Γ_2:

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

Fig. 3. A CML text from analysis.

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

$$\Gamma_2 \equiv A \subseteq \mathbb{R}, f : A \to \mathbb{R}, a : A$$

The WTT book corresponding to the text in Figure 3 is presented on Figure 4.

$\Gamma_1 \quad \triangleright \quad$ *the difference quotient of* $f := \frac{f(a+h)-f(a)}{h}$

$\Gamma_2 \quad \triangleright \quad f$ *is differentiable at* $a := \lim_{h\to 0} \frac{f(a+h)-f(a)}{h}$ exists

$\emptyset \quad \triangleright \quad \neg(\lambda_{x:\mathbb{R}}(\sqrt{|x|})$ is differentiable at 0)

Fig. 4. The analysis text of Figure 3 in WTT.

To illustrate how WTT can be extended with logical notation, let us look at another example. We translate the following CML proof of one of the rules of de Morgan, that says $\neg(a \wedge b) \Rightarrow (\neg a \vee \neg b)$, into WTT. Figure 5 shows the proof of the rule in CML.

Proof. Assume $\neg(a \wedge b)$. Assume $\neg\neg a$, i.e. a. Suppose now that b holds. This is not possible, since then we would have $a \wedge b$, contradicting our assumption. Hence $\neg b$ holds. Thus we have $\neg\neg a \Rightarrow \neg b$, or $\neg a \vee \neg b$. So $\neg(a \wedge b) \Rightarrow (\neg a \vee \neg b)$ is proved. □

Fig. 5. CML version of de Morgan's proof

The constants and binders used in a WTT book are listed in the so-called *preface* of that WTT book, which lists the weak types of the constants/binders and their arguments.

Preface:

Constants	*Arguments*	*Type*

$\vee, \wedge, \Rightarrow$	$(\texttt{stat}, \texttt{stat})$	$\texttt{stat}$
$\neg$	$(\texttt{stat})$	$\texttt{stat}$
$false$	$()$	$\texttt{stat}$

We first abbreviate the following contexts:

$$\Gamma_1 \equiv a : \texttt{stat}, b : \texttt{stat}$$
$$\Gamma_2 \equiv \Gamma_1, \neg(a \wedge b)$$
$$\Gamma_3 \equiv \Gamma_2, \neg(\neg(a))$$
$$\Gamma_4 \equiv \Gamma_3, b$$

Given the preface and the contexts $\Gamma_1 \ldots \Gamma_4$ as above, the informal proof of Figure 5 can be encoded by the WTT text on Figure 6. Notice that the

1	Γ_3	$\rhd$	a
2	Γ_4	$\rhd$	$a \wedge b$
3	Γ_4	$\rhd$	$false$
4	Γ_3	$\rhd$	$\neg(b)$
5	Γ_2	$\rhd$	$\neg(\neg(a)) \Rightarrow \neg(b)$
6	Γ_2	$\rhd$	$\neg(a) \vee \neg(b)$
7	Γ_1	$\rhd$	$\neg(a \wedge b) \Rightarrow (\neg(a) \vee \neg(b))$

Fig. 6. The proof of de Morgan's law translated into a WTT book

version of Figure 6 is a faithful representation of the CML version; therefore it also contains line 5, although this is not strictly necessary to derive de Morgan's rule. The WTT text does not include the logic reasoning behind the statements. In the corresponding extended version of WTT, we include the logical reasoning (based on rules of classical natural deduction) in the rightmost column. An impression of the WTT_L version can be found on the Figure 7, for a detailed introduction to the extended version of WTT see [6].

Although this could be done, the so-called logical comments of WTT_L version are not checked for correctness: the final check for correctness is done on the level of type theory, not here. We could also decide to translate this extended WTT version to other formalisms than type theory.

Next, we show how we can adapt the calculus of constructions so that we can translate the proof of Figure 7 into type theory in such a way that the structure of the proof is maintained.

1	Γ_3	$\rhd$	a	$\neg\neg$-elim on context: $\neg(\neg(a))$
2	Γ_4	$\rhd$	$a \wedge b$	$\wedge$-intro on line 1, context : b
3	Γ_4	$\rhd$	$false$	$\neg$-elim on ctx : $\neg(a \wedge b)$, line 2
4	Γ_3	$\rhd$	$\neg(b)$	$\neg$-intro on line 3
5	Γ_2	$\rhd$	$\neg(\neg(a)) \Rightarrow \neg(b)$	$\Rightarrow$-intro on line 4
6	Γ_2	$\rhd$	$\neg(a) \vee \neg(b)$	$\vee$-introleft on line 4
7	Γ_1	$\rhd$	$\neg(a \wedge b) \Rightarrow (\neg(a) \vee \neg(b))$	$\Rightarrow$-intro on line 6

Fig. 7. The proof of de Morgan's law translated into WTT_L – an extension of WTT with ND logical comments.

3 Extending λ_C with Global Parametric Definitions and Constants

To add global parametric definitions and parametric constants in the style of [11,15] to the Calculus of Constructions λ_C [10], we extend the set of pseudoterms by allowing constant instances $c(t_1, \ldots, t_n)$. This means that the set of the pseudoterms $\mathcal{T}$ is given by

$$\mathcal{T} := \mathcal{V} \mid \mathcal{C}(\mathcal{T} \ldots \mathcal{T}) \mid (\mathcal{T}\ \mathcal{T}) \mid (\lambda \mathcal{V} : \mathcal{T}\ .\ \mathcal{T}) \mid (\Pi \mathcal{V} : \mathcal{T}\ .\ \mathcal{T})$$

The pseudo-contexts are also extended by allowing the declaration and definition of global constants (respectively denoted as $\Gamma_1, c(\vec{x}:\vec{A}) : B, \Gamma_2$ and $\Gamma_1, c(\vec{x}:\vec{A}) := T : B, \Gamma_2$). The abbreviation $\vec{x}:\vec{A}$ stands for a list of declarations $x_1 : A_1, \ldots, x_n : A_n$.

The definitions can be subject to unfolding, which is usually referred to as δ-reduction:

$$\Gamma_1, c(\vec{x} : \vec{A}) := T : B, \Gamma_2 \vdash c(\vec{t}) \rightarrow_\delta T[\vec{x} := \vec{t}]$$

where $T[\vec{x} := \vec{t}]$ abbreviates $T[x_1 := t_1] \ldots [x_n := t_n]$.

In addition to the standard typing rules of λ_C, we have three extra rules for dealing with primitive parametric constants and parametric constant definitions. The first two rules are weakening rules that are needed to add primitive constants and definitions to the context. The third rule is needed for the use (i.e. instantiation) of these constants.

In addition, in the conversion rule of λ_C the β-equality is replaced by $\beta\delta$-equality. The resulting system will be referred to as λ_{CD}. It is contained in the systems described in [11,15], but here we do not use the additional possibility

to introduce local definitions.

4 Formalizing a CML proof

Let us start with a mathematical proof as it can be found in mathematics books. To clearly illustrate the technique we take a simple example and use a naive formulation of set theory here; a more elaborate example translation (of Pythagoras' proof that $\sqrt{2}$ is irrational) can be found in [6].

Consider the CML proof on Figure 8 of the statement $A \subseteq B \Rightarrow A \setminus B = \emptyset$ in set theory. Mathematicians have no problem reading this text and following

Proposition 4.1 $A \subseteq B \Rightarrow A \setminus B = \emptyset$

Proof. We prove the theorem by contradiction. Suppose $A \subseteq B$ and $x \in A \setminus B$, then $x \in A$ and $\neg(x \in B)$. Since $A \subseteq B$, we know that every element of A is an element of B. In particular this holds for x and therefore $x \in B$, which contradicts our assumption that $x \notin B$. □

Fig. 8. A proposition and its CML proof.

the idea of the proof although quite a few details are left implicit. One such detail is the set-theoretic foundation on which the proof is based, another is the precise logical reasoning justifying the steps of the proof and yet a third

8			$\triangleright$ $U : SET$	
9	*Lemma 1*	$X, Y : SET, a : U, a \in X \setminus Y$	$\triangleright$ $a \in X \wedge \neg(a \in Y)$	
10	*Lemma 2*	$X : SET$	$\triangleright$ $\forall_{x \in X}(false) \Rightarrow X =_{2s} \emptyset$	
11	Γ_1	$\triangleright$	$\subseteq(X : SET, Y : SET) := \forall_{x \in X}(x \in Y)$	
12	Γ_4	$\triangleright$	$x \in A \wedge \neg(x \in B)$	Lemma $1(X \mapsto A, Y \mapsto B, a \mapsto x)$
13	Γ_4	$\triangleright$	$x \in A$	$\wedge$-elimleft on line 12
14	Γ_4	$\triangleright$	$\forall_{y \in A}(y \in B)$	unfold $\subseteq(X \mapsto A, Y \mapsto B)$ on context: $A \subseteq B$
15	Γ_4	$\triangleright$	$x \in B$	$\forall$-elim on line 14 and line 13
16	Γ_4	$\triangleright$	$\neg(x \in B)$	$\wedge$-elimright on line 12
17	Γ_4	$\triangleright$	$false$	$\neg$-elim on line 16 and line 15
18	Γ_3	$\triangleright$	$\forall_{x \in A \setminus B}(false)$	$\forall$-intro on line 17
19	Γ_3	$\triangleright$	$A \setminus B = \emptyset$	$\Rightarrow$-elim on Lemma $2(X \mapsto A \setminus B)$ and ln 18
20	Γ_2	$\triangleright$	$A \subseteq B \Rightarrow A \setminus B = \emptyset$	$\Rightarrow$-intro on line 19

Fig. 9. The proof in WTT_L. Each line has a number, optional label, a context, the statement that is claimed or a definition of a constant, and finally a logical comment (not checked for consistency)

one is the ambiguity of natural language.

Compare the CML proof with the text in Figure 9 where the gray text represents items in the informal text that are (for the main part) implicit. In the right column we add the logical reasoning behind the statements. These so-called logical comments are built up according to rules, specific to the chosen logic (ND for classical first order logic in this case). The contexts Γ_i abbreviate the following:

$$\begin{aligned}\Gamma_1 &\equiv X, Y : SET\\ \Gamma_2 &\equiv A, B : SET\\ \Gamma_3 &\equiv A, B : SET, A \subseteq B\\ \Gamma_4 &\equiv A, B : SET, A \subseteq B, x : U, x \in A \setminus B\end{aligned}$$

Although this text contains much more detail than the original CML proof we can still read it and follow the proof step by step. Those steps are not the steps that we make in some proof system, but the statements that the author used or meant in his/her proof. In that sense *the formal version reflects the structure of the original proof.*

The proof of Figure 9 is written in a formal language and is verifiably in accordance with its syntax and rules, but its *logical* correctness has not been verified. To do that we need to fully formalize the logical comments (in gray) that it contains. This can be done by translating this text into type theory. One way to do that is to formulate the theorem in the language of a proof assistant and use its facilities (tactics) to prove it. But then of course we are creating a different proof whose structure is dictated by the proof assistant. Eventually, when we finish the proof, we have a proof term and a proof script that generated it, but their relation to the original proof is not clear.

Suppose that the type theory we want to translate our proof into is the Calculus of Constructions (λ_C) [10]. Using the formulas-as-types embedding of logic into λ_C, we can encode proofs into lambda terms. One such term [2] is presented in Figure 10. We can check that it really encodes a proof of the statement that we wanted to prove, but it does not reflect the proof that we started with and it can barely be called human-readable.

We will show that this does not necessarily have to be the case. We need a more declarative approach than using pure λ-terms. One possibility is to represent the proof not as one single λ-term but as a sequence of terms representing each step of the original proof. This sequence can naturally be expressed by a series of definitions in a context of a typing judgement in the following way: Suppose Γ_0 is a context describing all elements of the language that we work with and the global assumptions for the proof. Then each informal step of the proof (i.e. each line in its formulation in WTT)

[2] This is the β-normal form of the term presented in Appendix A, here given in the usual λ-calculus syntax.

$$
\begin{array}{l}
\lambda A, B{:}*_s \lambda p{:}(A \subseteq B).\\
\quad (H_2\ (A \setminus B)\\
\qquad \lambda x{:}U \lambda g{:}(x \in (A \setminus B)).\\
\qquad\quad (H_1\ A\ B\ x\ g\ (x \in B) \to \mathsf{False}\ (\lambda t{:}(x \in A) \lambda l{:}((x \in B) \to \mathsf{False}).l)\\
\qquad\qquad (p\ x\ (H_1\ A\ B\ x\ g\ (x \in A)\ (\lambda k{:}(x \in A) \lambda t{:}((x \in B) \to \mathsf{False}).k)))))\\
: \Pi A, B{:}*_s.(A \subseteq B) \to ((A \setminus B) = \emptyset)
\end{array}
$$

Fig. 10. A λ-term encoding a proof of $A \subseteq B \Rightarrow A \setminus B = \emptyset$. H_1 and H_2 encode the two lemmata used implicitly in the informal proof.

is added as a parameterized definition in the context, $\Gamma_0, l_1(\Delta_1) := M_1 : A_1, \ldots, l_n(\Delta_n) := M_n : A_n$. Each definition $l_i(\Delta_i){:}A_i$ represents the encoding A_i of the statement of line i and M_i encodes the reasoning steps to conclude A_i from the previous lines (see the box below). Note that each definition has a number of parameters Δ_i describing local variables and assumptions.

number	label	context		formula	log.		definition
i	l_i	Γ_i	$\rhd$	ϕ_i	p_i	$\longrightarrow$	$l_i(\Gamma_i) := p_i : \phi_i$

Fig. 11. Encoding a line as a definition.

Extending λ_C with definitions in the context (global definitions) is not difficult (see Section 3) and one can show that the obtained λ_{CD} enjoys the usual properties of the Calculus of Constructions (Severi [11], Bloo et al. [15,13], Kamareddine [12]).

Normally, proof-checking is translated into type-checking of the term that encodes the proof. In our case the checking of a proof becomes checking the validity of the context that encodes it. To illustrate how this works we present the sample proof as a typing context in Figure 12. There we see how each of the nine lines is encoded as a definition. Let Γ be the context of Figure 12 prefixed by context Γ_0 containing declarations for the elements of the language and the logic that we formalize the proof in. If in λ_{CD} we manage to prove $\Gamma \vdash * : \Box$ then we can be sure that the proof is correct.

Compare the representation of the proof by a context with the λ-term in Appendix A that corresponds to it. We obtained it by translating the proof in Figure 12 to the proof assistant Coq [9] and unfolding all definitions. Although both the context of Figure 12 and the term in Appendix A represent the same proof, the version using definitions in the context is more comprehensible.

Note that each of the steps of the original proof is still clearly present in the context presentation. This provides greater reliability because we can trace back the definitions to the steps of the *informal* proof.

Ideally the translation from the presentation on Figure 9 to the context on Figure 12 would be done automatically. The translation from the extended WTT version to λ_{CD} can, at least for a large part, be automated. We briefly describe the procedure below.

Each book, as in Figure 9, becomes a judgement of the form $\Gamma \vdash * : \Box$ in type theory. The constants and binders of the preface of the WTT book are included in the context of the judgement in λ_{CD}. Each line of the WTT text then becomes a constant definition in the context of the λ_{CD} translation. Each element of the WTT context becomes a parameter of the defined constant in λ_{CD}. The logical comments correspond to ND rules and using the Curry-Howard embedding can be encoded as λ-terms that form the body of the definition in the typing context that codifies a WTT_L line with a statement.

Since the WTT_L text may represent logically incorrect proofs, this translation can produce an invalid context, but even then we get a reference to the original text where the error originated from, by locating the definition in the context that did not typecheck. However, the experience in [6] shows that there are problems for fully automatic translation that require further investigation.

5 Related Work

In [11] Severi and Poll describe how to introduce definitions in a Pure Type System. Their work is extended by Bloo et al. [15,13], Kamareddine [12] by considering also parametric definitions. The rules presented in Section 3 are derivable from those works. Most of the current theorem proving systems support definitions of some sort. One of the well-known application of definitions there is to simulate a form of forward reasoning. It is interesting in our case that we only need to introduce global definitions in order to reflect the declarative aspects of proof-representation.

The need for a mathematician-friendly interface to proof assistants is also the motivation behind the mathematical proof language proposed by Barendregt [7]. It is inspired by the declarative approach taken by the system Mizar [14] where one writes his/her proof statement by statement and the computer tries to infer the reasoning steps between the statements. In [16] Wiedijk proposes formal proof sketches for the language of Mizar which are also close to CML presentations of proofs.

We thank the anonymous referee of the paper for pointing out the work of

$\mathrm{Lemma}_1(X : *_s, Y : *_s, a : U, p : a \in X \setminus Y) : \ a \in X \wedge \neg(a \in Y)$,

$\mathrm{Lemma}_2(X : *_s) : \ \forall_{x \in X}(false) \Rightarrow X =_{2s} \emptyset$,

$\subseteq(X : *_s, Y : *_s) := \ \underline{\forall(X, \lambda x : X.x \in Y)}$,

$l1(A : *_s, B : *_s, p : A \subseteq B, x : U, q : x \in A \setminus B) :=$

$\mathrm{Lemma}_1(A, B, x, q) : \underline{x \in A \wedge \neg(x \in B)}$,

$l2(A : *_s, B : *_s, p : A \subseteq B, x : U, q : x \in A \setminus B) :=$

$\wedge\text{-elimleft}(x \in A, \neg(x \in B), l1(A, B, p, x, q)) : \underline{x \in A}$,

$l3(A : *_s, B : *_s, p : A \subseteq B, x : U, q : x \in A \setminus B) :=$

$p : \underline{\forall_{y:A}(y \in B)}$,

$l4(A : *_s, B : *_s, p : A \subseteq B, x : U, q : x \in A \setminus B) :=$

$\forall\text{-elim}(A, \lambda x' : A.x' \in B, l3(A, B, p, x, q), l2(A, B, p, x, q)) : \underline{x \in B}$,

$l5(A : *_s, B : *_s, p : A \subseteq B, x : U, q : x \in A \setminus B) :=$

$\wedge\text{-elimright}(x \in A, \neg(x \in B), l1(A, B, p, x, q)) : \underline{\neg(x \in B)}$,

$l6(A : *_s, B : *_s, p : A \subseteq B, x : U, q : x \in A \setminus B) :=$

$\neg\text{-elim}(x \in B, l5(A, B, p, x, q), l4(A, B, p, x, q)) : \underline{false}$,

$l7(A : *_s, B : *_s, p : A \subseteq B) :=$

$\forall\text{-intro}(U, \lambda x : U.x \in A \setminus B \Rightarrow false, \lambda x : U.\lambda y : x \in A \setminus B.l6(A, B, p, x, y))$

$: \underline{\forall_{x \in A \setminus B}(false)}$,

$l8(A : *_s, B : *_s, p : A \subseteq B) :=$

$\Rightarrow\text{-elim}(\forall_{x \in A \setminus B}(false), A \setminus B = \emptyset, \mathrm{Lemma}_2(A \setminus B), l7(A, B, p))$

$: \underline{A \setminus B = \emptyset}$,

$l9(A : *_s, B : *_s, p : A \subseteq B) :=$

$\Rightarrow\text{-intro}(A \subseteq B, A \setminus B = \emptyset, \lambda x : A \subseteq B.l8(A, B, x)) : \underline{A \subseteq B \Rightarrow A \setminus B = \emptyset}$,

$* \ : \ \Box$

Fig. 12. The λ_{CD} context captures the structure of the CML proof

Huang [4] where Gentzen's natural deduction [3,2] proofs are presented at the assertion level. This is achieved by identifying domain-specific derivable rules that are used to abbreviate tedious parts of the proof. On the logical level this allows the representation of the same reasoning steps as in our presentation. However since we use WTT as a basis, the texts written in WTT_L can be seen as natural deduction proofs provided that they are correct. Since we strive for a low threshold formalism we impose weak (i.e. linguistic) constraints and therefore allow more texts including logically incorrect ones.

6 Conclusions

In our tiny case study we have shown that within currently available systems it is possible to choose an alternative representation of proofs that faithfully represents the structure of the informal arguments formalized in the proof. This gives us an increased level of confidence that the formal proof is the one the author meant. It also allows tracing back which parts of the formal proof correspond to which part of the informal one and therefore increases the maintainability of the proof. By starting the formalization as close as possible to the informal representation, we reduce the chance of formalization errors.

In recent years, we have seen a successful implementation of several different proof checking systems (e.g. COQ, Mizar, PVS, Lego, Nuprl, Agda, HOL, Isabelle, etc.). Although many of them have evolved to mature systems, wide acceptance of proof assistants by mathematicians outside the theorem proving community is still to come.

Now that their foundations have been sufficiently developed, the time has come for the proof assistants to become an accepted research tool for mathematicians. To achieve this goal we need to lower the threshold for using them. Therefore the formalization process needs to be made easier so that also mathematicians who are not type theory experts can use a proof assistant. We hope that with this paper we have shown how some of these goals could be achieved within the existing systems.

References

[1] Bruijn, N.G. de. *The Mathematical Vernacular, a Language for Mathematics with Typed Sets*, Proceedings of the Workshop on Programming Languages, Marstrand, Sweden, 1987, p. 865-936

[2] Andrews, P.B. *An Introduction to Mathematical Logic and Type Theory: To Thruth through Proof*, Academic Press, Inc., 1986

[3] Pravitz, D. *Natural Deduction: A Proof Theoretic Study*, Almquist & Wiksel, 1965

[4] Huang, X. *Reconstructing Proofs at the Assertion Level*, 12th International Conference on Automated Deduction, 1994

[5] Nederpelt, Rob. *Weak Type Theory: A formal language for mathematics*, May, 2002, CSR-02-05, Technische Universiteit Eindhoven

[6] Scheffer, Mark. *Formalizing Mathematics Using Weak Type Theory*, 2003, M.Sc. Thesis, Technische Universiteit Eindhoven, Available via `http://zax.mine.nu`

[7] Barendregt, Henk. *Towards an Interactive Mathematical Proof Language*, Thirty Five Years of Automath, F. Kamareddine ed., Kluwer, 2003

[8] Kamareddine, Fairouz and Nederpelt, Rob. *A derivation system for a formal language of mathematics*, July, 2001, Submitted for publication

[9] The Coq Development Team. *The Coq Proof Assistant Reference Manual – Version V7.4*, Feb 2003, `http://coq.inria.fr`

[10] Coquand, Thierry and Huet, Gérard. *The Calculus of Constructions*, Information and Computation, p. 95-120, 1988, number 76(2/3)

[11] Severi, P. and Poll, E. *Pure Type Systems with Definitions*, 1994, Proc. of LFCS'94, St. Petersburg, Russia, p. 316–328, LNCS series, number 813, Springer Verlag, Berlin

[12] Kamareddine, F. and Nederpelt, R. *Refining the Barendregt Cube using Parameters*, Proceedings of FLOPS 2001, p. 375–389, LNCS 2024, Springer Verlag, 2001

[13] Bloo, R. and Kamareddine, F. and Nederpelt, R. *The Barendregt Cube with Definitions and Generalized Reduction*, Information and Computation, 126(2), pp. 123–143, 1996

[14] Mizar. The System Mizar, `http://www.mizar.org`

[15] Bloo,R. and Kamareddine, Fairouz and Laan, Twan and Nederpelt, Rob. *Parameters in Pure Type Systems*, Proceedings of LATIN'02, 2002, Springer, volume LNCS 2286,

[16] Wiedijk, Freek. *Formal proof sketches*, Fokkink, Wan and Pol, Jaco van de, 7th Dutch Proof Tools Day, CWI, 2003

A Proof Term

The term below (in Coq syntax) is obtained by unfolding all definitions from Figure 12. λ-abstraction is incoded in Coq by square brackets and Π-abstraction by round brackets.

$$
\begin{array}{l}
[A, B{:}*_s;\ p{:}(A \subseteq B)] \\
\ ([A_0, B_0{:}*_s;\ p_0{:}(A_0 \subseteq B_0)] \\
\quad (H_2\ (A_0 \setminus B_0) \\
\quad\ ([A_1, B_1{:}*_s;\ p_1{:}(A_1 \subseteq B_1)] \\
\quad\quad ([A_2{:}*_s;\ P{:}(U \to *_p);\ p_2{:}((x{:}U)(x \in A_2) \to (P\ x));\ x{:}U](p_2\ x) \\
\quad\quad\ (A_1 \setminus B_1)\ [\_{:}U]\mathsf{False} \\
\quad\quad\ [x{:}U;\ g{:}(x \in (A_1 \setminus B_1))] \\
\quad\quad\ ([A_2, B_2{:}*_s;\ p_2{:}(A_2 \subseteq B_2);\ x_0{:}U;\ q{:}(x_0 \in (A_2 \setminus B_2))] \\
\quad\quad\quad ([A_3{:}*_p;\ p_3{:}([A{:}*_p]A \to \mathsf{False}\ A_3);\ q_0{:}A_3](p_3\ q_0) \\
\quad\quad\quad\ (x_0 \in B_2) \\
\quad\quad\quad\ ([A_3, B_3{:}*_s;\ p_3{:}(A_3 \subseteq B_3);\ x_1{:}U;\ q_0{:}(x_1 \in (A_3 \setminus B_3))] \\
\quad\quad\quad\quad ([A_4, B_4{:}*_p; \\
\quad\quad\quad\quad\ p_4{:}([A, B{:}*_p](\gamma{:}*_p)(A \to B \to \gamma) \to \gamma\ A_4\ B_4)] \\
\quad\quad\quad\quad\ (p_4\ B_4\ [\_{:}A_4;\ l{:}B_4]l)\ (x_1 \in A_3)\ (x_1 \in B_3) \to \mathsf{False} \\
\quad\quad\quad\quad\ ([A_4, B_4{:}*_s;\ \_{:}(A_4 \subseteq B_4);\ x_2{:}U; \\
\quad\quad\quad\quad\quad q_1{:}(x_2 \in (A_4 \setminus B_4))](H_1\ A_4\ B_4\ x_2\ q_1)\ A_3\ B_3\ p_3 \\
\quad\quad\quad\quad\quad x_1\ q_0))\ A_2\ B_2\ p_2\ x_0\ q) \\
\quad\quad\quad\ ([A_3, B_3{:}*_s;\ p_3{:}(A_3 \subseteq B_3);\ x_1{:}U; \\
\quad\quad\quad\quad q_0{:}(x_1 \in (A_3 \setminus B_3))] \\
\quad\quad\quad\quad ([A_4{:}*_s;\ P{:}(U \to *_p); \\
\quad\quad\quad\quad\ f_0{:}([A{:}*_s;\ P{:}(U \to *_p)](x{:}U)(x \in A) \to (P\ x)\ A_4\ P); \\
\quad\quad\quad\quad\ t{:}U;\ q_1{:}(t \in A_4)](f_0\ t\ q_1)\ A_3\ [y{:}U](y \in B_3) \\
\quad\quad\quad\quad\ ([A_4, B_4{:}*_s;\ p_4{:}(A_4 \subseteq B_4);\ x_2{:}U; \\
\quad\quad\quad\quad\quad \_{:}(x_2 \in (A_4 \setminus B_4))]p_4\ A_3\ B_3\ p_3\ x_1\ q_0)\ x_1 \\
\quad\quad\quad\quad\ ([A_4, B_4{:}*_s;\ p_4{:}(A_4 \subseteq B_4);\ x_2{:}U;\ q_1{:}(x_2 \in (A_4 \setminus B_4))] \\
\quad\quad\quad\quad\quad ([A_5, B_5{:}*_p; p_5{:}([A, B{:}*_p](\gamma{:}*_p)(A \to B \to \gamma) \to \gamma\ A_5\ B_5)] \\
\quad\quad\quad\quad\quad\quad (p_5\ A_5\ [k{:}A_5;\ \_{:}B_5]k)\ (x_2 \in A_4) \\
\quad\quad\quad\quad\quad\ (x_2 \in B_4) \to \mathsf{False} \\
\quad\quad\quad\quad\quad\ ([A_5, B_5{:}*_s;\ \_{:}(A_5 \subseteq B_5);\ x_3{:}U; \\
\quad\quad\quad\quad\quad\quad q_2{:}(x_3 \in (A_5 \setminus B_5))](H_1\ A_5\ B_5\ x_3\ q_2)\ A_4 \\
\quad\quad\quad\quad\quad\quad B_4\ p_4\ x_2\ q_1))\ A_3\ B_3\ p_3\ x_1\ q_0))\ A_2\ B_2\ p_2\ x_0\ q)) \\
\quad\quad A_1\ B_1\ p_1\ x\ g))\ A_0\ B_0\ p_0))\ A\ B\ p) \\
\ : (A, B{:}*_s)(A \subseteq B) \to ((A \setminus B) = \emptyset)
\end{array}
$$