Originally published in: Chiarcos, Chistian/Eckart de Castilho, Richard/Stede, Manfred (Eds.): Von der Form zur Bedeutung: Texte automatisch verarbeiten. From Form to Meaning: Processing Texts Automatically. Proceedings of the Biennial GSCL Conference 2009. - Tübingen: Narr, 2009. Pp. 137-145.

From Proof Texts to Logic
Discourse Representation Structures for Proof Texts in Mathematics*

Jip Veldman[1], Bernhard Fisseni[2], Bernhard Schröder[2], Peter Koepke[1]

[1] Rheinische Friedrich-Wilhelms-Universität Bonn, Mathematisches Institut
[2] Universität Duisburg-Essen, Germanistik / Linguistik

Abstract

We present an extension to Discourse Representation Theory that can be used to analyze mathematical texts written in the commonly used semi-formal language of mathematics (or at least a subset of it). Moreover, we describe an algorithm that can be used to check the resulting Proof Representation Structures for their logical validity and adequacy as a proof.

1 Introduction

The era of theorem provers (computer programs that try to prove that a mathematical statement is true) and proof checkers (computers programs that try to check the validity of a mathematical argument) started in the beginning of the sixties.

While from the beginning (Abrahams, 1964), the aim was to process proofs as written by mathematicians, this goal was not attained. All the proof checkers developed to date only accept input written in peculiar formal languages that do not have much in common with the language that mathematicians normally use and are thus quite inaccessible or unattractive to most mathematicians.

Meanwhile, computational linguistics has developed many techniques and formalisms. The aim of the Naproche project is to apply such techniques to proof checking systems; the resulting system will accept proof texts that can be read by humans as well as by machines, thus coming closer to the original goal of the pioneers in the field. The system will support wide range of applications in educational settings (teaching to write good proofs) and mathematics in general (supporting mathematicians in constructing proofs and ensuring comprehensibility).

We describe in this paper the general architecture and formalism behind our approach to processing natural language mathematical texts, which relies on an extension of Discourse Representation Theory (DRT, Kamp & Reyle, 1993; for further reference on Naproche technicalities, see Cramer, 2009; Kühlwein, 2008).[1]

* Erschienen in: C. Chiarcos, R. Eckart de Castilho, M. Stede (Hrsg.), *Von der Form zur Bedeutung: Texte automatisch verarbeiten / From Form to Meaning: Processing Texts Automatically*. Tübingen: Narr, 2009, S. 137–145.

[1] These papers and other material are available on the website, `http://www.naproche.net`

This paper is organized as follows: Section 2 provides some background on proof checking. In Section 3, we will give some of the most striking features of the language of mathematics and describe to what extent they have been implemented in the Naproche system. In Section 4, we will describe how to extend DRT in order to reflect these features. Finally, in Section 5, we will show how we can check such a representation structure for its logical validity.

2 Proof Checking

Not only computational linguistics, but also theorem proving has left infancy. During the early years, research concentrated on the development of general theorem provers. As we now know, to attain this general goal is an impossible task for reasons of computational complexity. This is one of the reasons which led to the development of proof checking; for practical purposes, the involvement of humans in proof checking makes proof checking 'easier' than fully automated proving.

At the end of the sixties, Nicolas Govert de Bruijn was the first one to successfully implement a proof checker. His Automath system (de Bruijn, 1994) was the first proof checker in which a substantial mathematical text, namely Landau (1930), could be checked – after it was appropriately formalized by human operators. Today there is a large community of computer scientists developing and applying proof checking systems. Several complex mathematical theorems like the Four Colour Theorem (Gonthier, 2005, using the Coq system) and Gödel's completeness theorem for first-order logic (Braselmann & Koepke, 2005, using the Mizar system) have been formalized and checked using a proof checking system.

Thus, these programmes are attractive in their own way, but the fact that they demand peculiar input has hampered their adoption by general mathematicians; one of the aims of our approach is to allow the use of sophisticated modern provers and checkers to mathematical natural language texts.

3 The Language of Mathematics and the Language of Naproche

Mathematicians use a small subset of written natural language enriched by a significant amount of formulaic notation to communicate mathematical theories. We call it the *semi-formal language of mathematics* (SFLM).

As part of the Naproche project, a *controlled natural language* that captures a subset of the features of SFLM has been developed. Below, we describe the most important features of SFLM and the state of the implementation in the Naproche system. To illustrate the coverage of the system so far: The first chapter of Landau (1930) has been adapted to the Naproche language. The difficulties encountered were all solvable up to now.

The coverage of Naproche is constantly being extended. The most stable core to date permits to formulate proofs in a style that looks much more like a normal SFLM proof than proofs formallized for any other proof checker. The processing of proof metastructure is already in place and quite complete, and now the focus is on extending the linguistic form of single sentences.

The Naproche language is more formal than SFLM in some regards on which we comment below and more restricted than the language used by most mathematicians; this is also due to the fact that one of the uses for which Naproche is developed is teaching how to write proofs. As an example we give a proof of "$\sqrt{2}$ is irrational" written in Naproche style.

Theorem. $\sqrt{2}$ is irrational.

Proof. Assume that $\sqrt{2}$ is rational. Then there are integers a, b such that $a^2 = 2 \cdot b^2$ and $gcd(a, b) = 1$. Hence a^2 is even, and therefore a is even. So there is an integer c such that $a = 2 \cdot c$. Then $4 \cdot c^2 = 2 \cdot b^2$, $2 \cdot c^2 = b^2$, and b is even. Contradiction. (QED)

However, one has to bear in mind that SFLM is not a uniform phenomenon. The language used in textbooks is much more explicit and much more formal than the language of advanced journal articles. Therefore, Naproche language and the accepted textual structure ist much closer to the language of algebraically written textbooks than this language sometimes is to the language of highly specialised articles. An example for this are geometric argumentations (as in knot theory) and coarse-grained argumentations, which latter are typical for specialised papers. Both defy formalisation at the current state of the art.

Cultural background. SFLM is considered so important by the community of mathematicians that mathematics students will spend a lot of time during the first months at university learning to communicate in it. However, it has not seen much attention from linguists (Eisenreich, 1998; Ranta, 1999; Zinn, 2006 being some of the few exceptions). As an example of SFLM we cite two small fragments from page 11 and 12 of Kunen (1980), a standard textbook in set theory. The first quotation shows that Zermelo-Fraenkel set theory proves that there is no universal set.

Theorem. $\neg\exists z \forall x (x \in z)$.

Proof. If $\forall x (x \in z)$, then, by Comprehension, form $\{x \in z : x \notin x\} = \{x : x \notin x\}$, which would yield a contradiction by the Russell paradox discussed above. (QED)

The second quotation gives the definition of an ordered pair.

... $\langle x, y \rangle = \{\{x\}, \{x, y\}\}$ is the ordered pair of x and y.

SFLM incorporates parts of the syntax and semantics of natural language, so that it takes over its complexity and some of its ambiguities. However, SFLM texts are distinguished from common natural language texts by several characteristics:

Mixing formulas and natural language. Proofs combine natural language expressions with mathematical symbols and formulas, which can syntactically function like noun phrases, as grammatical predicates, as predicatives, as modifiers, or even as whole sentences.

At the moment, Naproche supports such mixed language unless the formulaic part has scope over several sentences (which is often considered 'bad style') or, conversely, inside a formulaic expression, natural language fragments are used.

Avoiding ambiguities. In SFML, constructions which are hard to disambiguate are generally avoided. An important example is coreference: Mathematical symbols ensure unambiguous reference, most importantly variables fulfill the task of anaphoric pronouns in other natural language. Therefore, referents are generally labelled explicitly if ambiguity could arise, as in the beginning of

the first example, where it is assumed that the universal set exists and at the same time it is labelled with the variable name "z".

Therefore, Naproche treats mathematical referents differently from other referents, and also identifies free variables in formulas, as these are available as antecedents in natural language fragments. It is thus not sufficient to only semantically process formulas, but the formulaic representation must be taken into account because it retains important information for the proof representation.

Variables are treated differently by Naproche depending on the place of introduction. For variables introduced explicitly (by *There is an* x), Naproche uses the dynamic semantics of DRS. Remember that using existentially quantified variables on the left side of an implication DRS leads to implicit universal quantification.

Variables implicitly introduced in a proof are assumed to be existentially quantified, while variables introduced implicitly in theorems, lemmas and definitions are treated as universally quantified, even if the linguistic form is not that of an implication. In this, Naproche follows mathematical custom.

Assumption management. It is characteristic of SFML texts that assumptions are introduced and retracted in the course of the argument. Contrary to general texts, in SFLM the scopes of assumptions are deeply nested.

To take an example, the proof cited above is a proof by contradiction: At the beginning (in the *if* clause), it is assumed that the universal set z exists. All subsequent claims are relativised to this assumption. Finally, the assumption leads to a contradiction, and is retracted, concluding that the universal set does not exist.

In Naproche, management of assumptions has been implemented. At the moment, assumptions can be introduced explicitly in various ways, e. g. by *Assume*. The scope of assumptions extends to the end of a proof (marked by *QED*), unless they are explicitly discharged using *Thus*.

Extending the vocabulary. Definitions in SFLM texts add new symbols and expressions to the vocabulary and fix their meaning.

Naproche supports definitions; if an expression or symbol is used before its definition, the user is informed of this mistake.

Text structure and intratextual references. Mathematical texts are highly structured. At a global level, they are commonly divided into *building blocks* like definitions, lemmas, theorems and proofs. Inside a proof, assumptions can be nested into other assumptions, so that the scopes of assumptions define a hierarchical proof structure.

Proof steps are then commonly justified by referring to results in other texts, or previous passages in the same text. In the first example two intratextual references are used: "Comprehension", referring to the Axiom of Comprehension, and "the Russell paradox discussed above"; special identifiers representing the structure of the mathematical text, such as "Theorem 1.5" are also common.

Proof structure is therefore an inherent feature of the Naproche language, including intratextual references and the corresponding reference markers ("Lemma 42" and later "by Lemma 42").

Contradiction. Proofs by contradiction are supported in Naproche; this is signalled by *Contradiction* at the end of the (sub)proof.

Notation support. Syntactically, Naproche supports a wide range of mathematical formula notation, the semantics of which must be defined in the text or in a module containing background knowledge.

Linguistic form of sentences. The linguistic form of sentences in SFLM is quite restricted.

Naproche accepts simple declarative SFLM sentences consisting of a mix of natural language terms and formulaic notation (see above); furthermore, relative sentences with *such that*, which are very typical of SFLM, can be processed, as well as intratextual references (*by Lemma 42*). Natural language terms can be noun phrases, unary and binary predicates, consisting of a verb or alternatively a copula.

At the moment, we are working toward support for adjectives and nouns with complements and argument alternation (*A is equivalent to B* and *A and B are equivalent*).

Guiding the reader. Some linguistic forms used in SFLM can be safely excluded from the coverage of Naproche. The most important case are comments that guide the reader but do not contribute to the semantics of the proof. While limited coverage for such commentaries as "by Induction" in (1-1a) has been implemented, as the information can also guide the proof checking process, general coverage of such commentary is not planned because of the immense semantic complications: the commentary in (1-1b) speaks about the structure of the proof, temporarily suspending previously established variable bindings.

(1) (a) Thus by induction, if $x \neq 1$ then there is a u such that $x = u'$.[2]

(b) We now prove that x is unique in the formula given above.

4 From Discourse to Proof Representation Structures

We use techniques based on Discourse Representation Theory (DRT, Kamp & Reyle, 1993) to analyze texts written in SFLM. Claus Zinn pioneered such an approach in Zinn (2003), Zinn (2004) and Zinn (2006). We call a discourse representation structure (DRS) whose structure is extended to suit mathematical discourse *Proof Representation Structure* (PRS).[3]

A PRS has five constituents: [1] An identification number (i), [2] a list of discourse referents (d_1, ..., d_m), [3] a list of mathematical referents ($m_1, \ldots, m_n$), [4] a list of textual referents ($r_1, \ldots, r_p$) and [5] an *ordered* list of conditions ($c_1, \ldots, c_l$). Similar to DRSes, we can display PRSes as boxes. The following box illustrates the general structure of a PRS:

i

$d_1, \ldots, d_m$	$m_1, \ldots, m_n$
c_1	
$\vdots$	
c_l	
$r_1, \ldots, r_p$	

The *identification number* (i above) can later be used in textual referents in intratextual and intertextual references (which are stored in the 'drawer' containing $r_1, \ldots, r_p$ above, and assigned using

[2] This example is from the translation of Landau (1930) into the Naproche language.

[3] The name, if not the structure itself, is due to Zinn's work (Zinn, 2006).

the $use()$ condition, see below). If a PRS represents one of the building blocks mentioned above, we add a marker qualifying the type (theorem, proof, etc.) to its identification number.

As in DRSes, *discourse referents* ($d_1, \ldots, d_m$ above) are used to identify objects in the domain of the discourse. However, the domain contains three kinds of objects: first, mathematical objects like numbers or sets, secondly, symbols and formulas that are used to make claims about mathematical objects, and finally, textual referents. Discourse referents can refer to all three kinds of objects.

Mathematical referents ($m_1, \ldots, m_n$) are the terms and formulas which appear in the text; they must be available as their syntactic structure is exploited in mathematical reasoning and are bound using $mathid()$ conditions (see below). In the current PRSes, free variables contained in a mathematical formula are identified and can thus be referred to in later discourse.

Finally, there is an *ordered* list of conditions. Just as in the case of DRSes, PRSes and PRS conditions are defined recursively: let A, B be PRSes, $X, X_1, \ldots, X_n$ discourse referents, Y a mathematical referent, and Z a textual referent. Then

1. for any n-ary predicate p (e.g. expressed by adjectives and noun phrases in predicative use and verbs in SFLM), $p(X_1, \ldots, X_n)$ is a condition.

2. $holds(X)$ is a condition representing the claim that the formula referenced by X is true.

3. $mathid(X, Y)$ is a condition which binds a discourse referent X to a mathematical referent Y (a formula or a term).

4. $use(Z)$ is a condition representing that the source of the textual referent Z explains the previous or the next proof step.

5. A is a condition, i. e. conditions can be nested.

6. $\neg A$ is a condition representing the negation of A.

7. $A \Rightarrow B$ is a condition representing an implication, a universal quantifier, or an assumption and its scope.

8. *contradiction* is a condition.

9. $A := B$ is a condition representing a definition.

Note that contrary to the case of DRSes, a bare PRS can be a direct condition of a PRS, and PRSes are not merged in a way that the provenance of conditions becomes intransparent. This allows to represent in a PRS the structure of a text divided into building blocks (definitions, lemmas, theorems, proofs) by structure markers. The hierarchical structure of assumptions is represented by nesting conditions of the form $A \Rightarrow B$: A contains an assumption, and B contains the representation of all claims made inside the scope of that assumption.

The algorithm creating PRSes from a text in SFLM proceeds sequentially: It starts with the empty PRS. Each sentence or structure marker in the discourse updates the PRS according to an algorithm similar to a standard DRS construction algorithm, but taking the nesting of assumptions into account. The following figures show simplified versions of the PRSes constructed from the examples from Kunen (1980) given above.

theorem.1

goal.2

0	$\neg\exists z\forall x(x \in z)$
$mathid(0, \neg\exists z\forall x(x \in z))$	
$holds(0)$	

body.3

id.4

1,2	$z, \forall x(x \in z)$
$mathid(1, z)$	
$mathid(2, \forall x(x \in z))$	
$holds(2)$	

$\Rightarrow$

id.5

3,4,5	$\{x \in z : x \notin x\}, \{x : x \notin x\}$ $\{x \in z : x \notin x\} = \{x : x \notin x\}$
use(comprehension)	
$mathid(3, \{x \in z : x \notin x\})$	
$mathid(4, \{x : x \notin x\})$	
$mathid(5, \{x \in z : x \notin x\} = \{x : x \notin x\})$	
$holds(5)$	
$contradiction$	
use(Russell paradox)	
comprehension, Russell paradox	

id.0

$\langle x, y\rangle$	

:=

id.1

1,2,3,4,5	$x, y, \langle x, y\rangle, \{\{x,\}, \{x, y\}\}, \langle x, y\rangle = \{\{x,\}, \{x, y\}\}$
$mathid(1, x)$	
$mathid(2, y)$	
$mathid(3, \langle x, y\rangle)$	
$mathid(4, \{\{x,\}, \{x, y\}\})$	
$mathid(5, \langle x, y\rangle = \{\{x,\}, \{x, y\}\})$	
$holds(5)$	
$alternative.name(3,$ ordered pair of $)$	

Even in these easy examples there are various interesting phenomena in the PRS constructions:

1. While parsing the statement "$\forall x(x \in z)$", we compute the list of free variables of this formula. We add a new discourse referent for the free variable z and quantify this new variable existentially, because this variable is not bound be any assumption made before.
2. While parsing "$\{x \in z : x \notin x\} = \{x : x \notin x\}$", we also extract the mathematical objects used in this formula and give them a discourse referent.
3. the PRS condition $alternative.name$ codes a alternative description in natural language for the object defined formally.

5 Checking PRSes

There is a natural translation from a PRS to first-order logic. So the correctness of the text represented in a PRS can be checked using a theorem prover. We proceed by first translating a PRS into first-order logic, and then a theorem prover for first-order logic can be used to check the result of this translation. For the first step we use a variant of the translation from DRSes to first-logic described by Blackburn & Bos (2003). We list the most important changes that have to be made to the algorithm in order to parse PRSes correctly:

1. $mathid$ conditions are not translated into subformulas of the first-order translation. A $mathid$-condition, which binds a discourse referent n to a term t, triggers a substitution of n, by t, in the translation of subsequent conditions. A $mathid$-condition, which binds a discourse referent n to a formula ϕ, causes a subsequent $holds(n)$-condition to be translated to ϕ.
2. Definition conditions are also not translated into subformulas of the translation. Instead, a condition of the form $A := B$ triggers a substitution of the relation symbol it defines, by the first-order translation of B, in subsequent conditions.
3. The $holds$-conditions that occur in a $goal$-PRS are translated after parsing the following $body$-PRS.
4. $use(Z)$ is not translated, but should be used as a hint for the proof checker.

For example, the PRS of the first example can be translated as follows into first-order logic enriched with class terms:

$$[\forall z(\forall x(x \in z)) \Rightarrow (\{x \in z : x \notin x\} = \{x : x \notin x\} \wedge \bot)] \Rightarrow \neg\exists z \forall x(x \in z)$$

The checking algorithm used in Naproche is based on, but more complex than the algorithm described above. For example we did not explain how use is used as a hint for the proof checker.

6 Conclusion

We have sketched how a number of phenomena characteristic for the semi-formal language of mathematics are treated in our proof checking system Naproche. We extend the DRS approach to discourse semantics in a number of ways to deal with phenomena like variables, mathematical formula, explicit references to text parts, definitions and assumptions with wide scope. All extensions preserve the direct interpretability in first-order logic of standard DRT.

References

Abrahams, P. W. (1964). Application of lisp to checking mathematical proofs. In E. C. Berkeley & D. G. Bobrow, editors, *The Programming Language LISP: Its Operations and Applications*, pages 137–160. The MIT Press, Cambridge (USA), London.

Blackburn, P. & Bos, J. (2003). *Working with Discourse Representation Theory: An Advanced Course in Computational Linguistics*. CSLI, Stanford.

Braselmann, P. & Koepke, P. (2005). Gödel's completeness theorem. *Formalized Mathematics*, 13, 49–53.

Cramer, M. (2009). *Mathematisch-logische Aspekte von Beweisrepräsentationsstrukturen*. Master's thesis, Rheinische Friedrich-Wilhelms-Universität Bonn.

de Bruijn, N. G. (1994). Reflections on automath. In R. B. N. et al., editor, *Selected Papers on Automath*, volume 133 of *Studies in Logic*, pages 201–228. Elsevier.

Eisenreich, G. (1998). Die neuere Fachsprache der Mathematik seit Carl Friedrich Gauß. In L. Hoffmann, H. Kalverkämper, H. E. Wiegand, C. Galinski, & W. Hüllen, editors, *Fachsprachen / Languages for Special Purposes: ein internationales Handbuch zur Fachsprachenforschung und Terminologiewissenschaft / An International Handbook of Special-Language and Terminology Research, 1. Halbband*, number 14 in Handbücher zur Sprach- und Kommunikationswissenschaft, chapter 136, pages 1222–1230. de Gruyter, Berlin, New York.

Gonthier, G. (2005). A computer-checked proof of the four colour theorem. unpublished ms. `http://research.microsoft.com/en-us/people/gonthier/4colproof.pdf`.

Kamp, H. & Reyle, U. (1993). *From Discourse to Logic*. Kluwer, Dordrecht.

Kühlwein, D. (2008). A calculus for proof representation structures. Diploma Thesis. Rheinische Friedrich-Wilhelms-Universität Bonn.

Kunen, K. (1980). *Set theory*, volume 102 of *Studies in Logic and the Foundations of Mathematics*. North-Holland Publishing Co., Amsterdam. An introduction to independence proofs.

Landau, E. (1930). *Grundlagen der Analysis. Das Rechnen mit ganzen, rationalen, irrationalen, komplexen Zahlen*. Akademische Buchgesellschaft, Leipzig.

Ranta, A. (1999). Structures grammaticales dans le Français mathématique. *Mathématiques, informatique et Sciences Humaines*, pages 138:5–56; 139:5–36.

Zinn, C. (2003). A computational framework for understanding mathematical discourse. *Logic Journal of the IGP*, 11(4), 457–484.

Zinn, C. (2004). *Understanding Informal Mathematical Discourse*. Ph.D. thesis, Institut für Informatik, Universität Erlangen-Nürnberg.

Zinn, C. (2006). Supporting the formal verification of mathematical texts. *Journal of Applied Logic*, 4(4), 592–621.