The NAPROCHE Project

Linguistics and logic of common mathematical language I

Peter Koepke and Merlin Carl, Mathematical Institute
Universität Bonn

Mathematical texts are formulated in a semi-formal language, mixing natural language discourse and mathematical formulas. The meaning of a mathematical text can be described by a translation into first-order formulas; the text is correct if its first-order translation is a formal proof in a first-order proof calculus. The NAPROCHE project (NAtural language PROof CHEcking) recognizes that there is a specific, semi-formal or natural mathematical language which ought to be studied by linguistic techniques. The project aims at constructing a system which checks the correctness of texts written in a controlled but rich sublanguage of ordinary mathematical language including TEX-style typeset formulas. In our talk we demonstrate a small working prototype, explain its modular structure, and discuss future enhancements and extensions, with an emphasis on the mathematical aspects of the system and its applications.

http://www.math.uni-bonn.de/people/naproche/

Workshop *Deduction in Semantics*

Institut für Maschinelle Sprachverarbeitung,
Stuttgart, October 11, 2007

The language of mathematics, example

Euclid:

PROPOSITIO XXXII.

Pars I.

Fig. 51. *Omnis trianguli externus quivis angulus (FBC) duobus internis oppoſitis (A & C) æqualis eſt.*

(f) Per 31. l. 1. Per B duc *(f)* BL parallelam ad AC. Quia duas parallelas BL, AC ſecat FA, erit externus angulus FBL interno
(g) Per 27. l. 1. A *(g)* æqualis. Et quia eaſdem parallelas BL. AC ſecat etiam recta BC, erit LBC ſibi alterno C *(h)* æqualis. Ergo
(h) Per eand. totus FBC æquatur utrique ſimul A & C. Quod erat demonſtrandum.

Corollaria.

Fig. 51. 1. Externus angulus (FBC) quolibet internorum oppoſitorum A vel C, major eſt.

Fig. 39. 27. 2. Angulorum (C & AOB) eandem baſim (AB) habentium, major eſt (AOB,) qui intra cadit.

Pro-

The language of mathematics, example

Gauss:

10 DE NUMERORUM CONGRUENTIA

Numerorum congruentiam hoc signo, $\equiv$, in posterum denotabimus, modulum ubi opus erit in clausulis adiungentes, $-16 \equiv 9$ (mod. 5), $-7 \equiv 15$ (mod. 11)*).

3.

THEOREMA. *Propositis* m *numeris integris successivis*

$$a,\ a+1,\ a+2\ \ldots.\ a+m-1$$

alioque A, *illorum aliquis huic secundum modulum* m *congruus erit, et quidem unicus tantum.*

Si enim $\frac{a-A}{m}$ integer, erit $a \equiv A$, sin fractus, sit integer proxime maior, (aut quando est negativus, proxime *minor,* si ad signum non respiciatur) $=k$, cadetque $A+km$ inter a et $a+m$, quare erit numerus quaesitus. Et manifestum est omnes quotientes $\frac{a-A}{m}$, $\frac{a+1-A}{m}$, $\frac{a+2-A}{m}$ etc. inter $k-1$ et $k+1$ sitos esse; quare plures quam unus integri esse nequeunt.

Residua minima.

4.

Quisque igitur numerus residuum habebit tum in hac serie, $0, 1, 2, \ldots m-1$, tum in hac, $0, -1, -2, \ldots. -(m-1)$, quae *residua minima* dicemus, patetque, nisi 0 fuerit residuum, bina semper dari, *positivum* alterum, alterum *negativum*.

The language of mathematics, example

This example, de Morgan's law, could be a basic exercise in an introductory logic course; the solution is readable (by humans) and is also accepted by the NAPROCHE proof checking system. This indicates that the distance between natural proofs and formal proofs may be reduced or even eliminated.

Theorem. $\alpha \wedge \beta \leftrightarrow \neg(\neg\alpha \vee \neg\beta)$.

Proof. Assume $\alpha \wedge \beta$. Assume for a contradiction that $\neg \alpha \vee \neg\beta$.
Assume $\neg\alpha$. α. Contradiction. Thus $\neg\alpha \rightarrow \perp$.
Assume $\neg\beta$. β. Contradiction. Thus $\neg\beta \rightarrow \perp$.
Hence contradiction. Thus $\neg(\neg\alpha \vee \neg\beta)$. Thus $\alpha \wedge \beta \rightarrow \neg(\neg \alpha \vee \neg\beta)$.

Assume $\neg(\neg\alpha \vee \neg\beta)$. Assume $\neg\alpha$. $\neg\alpha \vee \neg\beta$. Contradiction. Thus α.
Assume $\neg\beta$. $\neg\alpha \vee \neg\beta$. Contradiction. Thus β. $\alpha \wedge \beta$.
Thus $\neg(\neg\alpha \vee \neg\beta) \rightarrow \alpha \wedge \beta$. Qed.

Interpretations

- The *formalistic or logicistic approach*:

the common mathematical language denotes / abbreviates texts written in (first-order) logic; this language is *not* an object of study of the foundations of mathematics.

- The *naturalistic approach*:

  – treats the common mathematical language (CML) like a natural language; slogan: „take the mathematical language serious"

  – the grammar of CML is a natural language grammar for the plain text components plus a grammar for mathematical terms and formulas plus some typically mathematical constructs

  – the semantics of CML is mainly given by an adequate first-order formalization

  – the pragmatics of CML is determined by the desire to write texts and proofs, whose formalizations are formally correct

Some characteristics the *grammar of CML*:

  – combination of natural language and "mathematical formulas"

  – specific, (re-)defined words, and figures of speech

  – hypothetical constructions ("assume", "define", "let", ...)

- typography ($\alpha, \beta, ..., \frac{a}{b}, \sqrt{\ }, ...$)
- graphics (diagrams, pictures, ...)
- definitions, theorems, proofs
- ...

The *semantics of CML* can be taken in line with the logicistic approach:

- (first-order) predicate logic
- Gentzen-style natural deduction
- addition of further (implicit) assumptions and arguments
- possibility of ambiguities in the translation

The *pragmatics of CML* is guided by the mathematical practice:

- communication „about some objective mathematical reality“
- concise and unambiguous, up to some „irrelevant“ details
- language is used with a common mathematical background knowledge which follows from some foundational theory, usually Zermelo-Fraenkel set theory (ZFC)
- correct and complete proofs

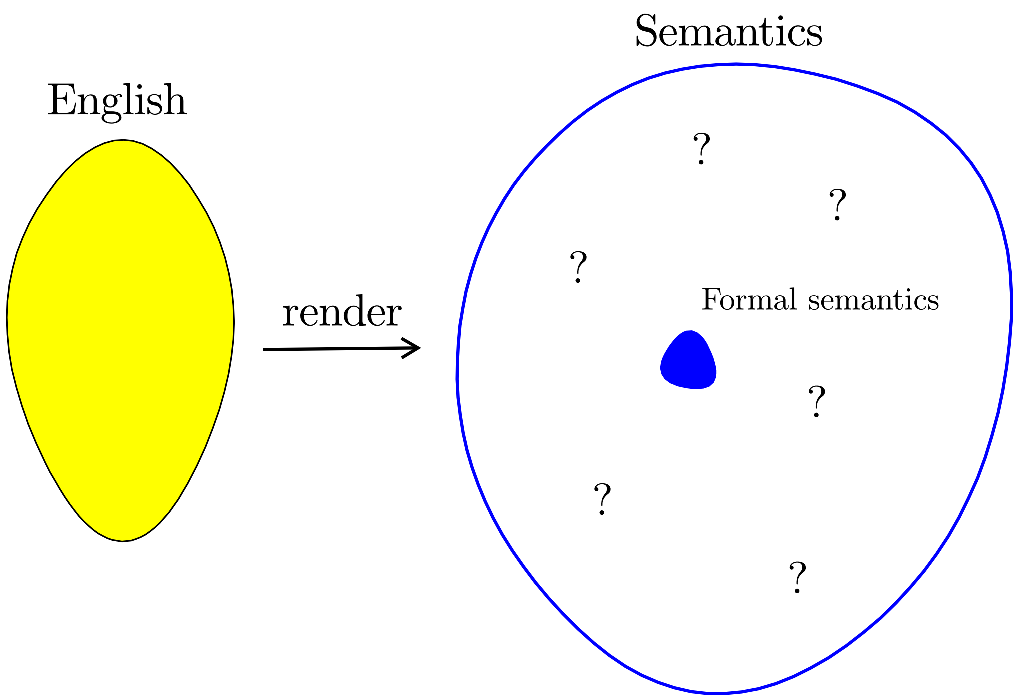
– the reader has to supply relevant and fitting background knowledge as to make mathematical texts correct; ambiguities can usually be resolved with this criterion

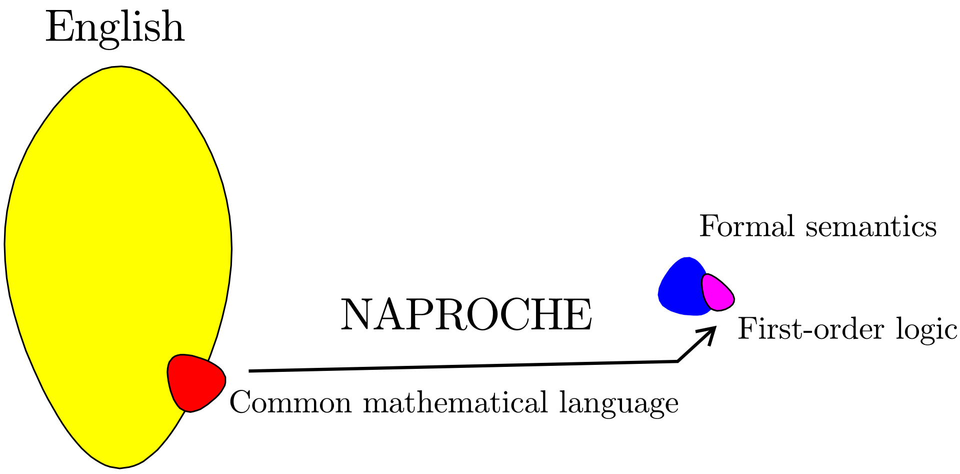
– "distances" between a text and its formalization can vary

A programmatic text from the initial phase of NAPROCHE describes this very clear cut semantic and pragmatic situation by:

... From a linguistic perspective, the Language of Mathematics is distinguished by the fact that its core mathematical meaning can be fully captured by an intelligent translation into first-order predicate logic. ...

This radical simplification of the linguistic framework can be pictured as follows:





This observation motivates the NAPROCHE (NAtural language PROof CHEcking) *project* and its principal components:

- modeling the grammar, semantics, and pragmatics of CML
- theoretical studies and practical implementations
- controlled mathematical language
- semantics using DRT; proof representation structures PRS
- proof checking using existing formal proof-checkers
- applications:
  - interfaces for formal mathematics systems
  - writing texts for humans and machines

- tutorial applications
- interactions with the philosophy of mathematics

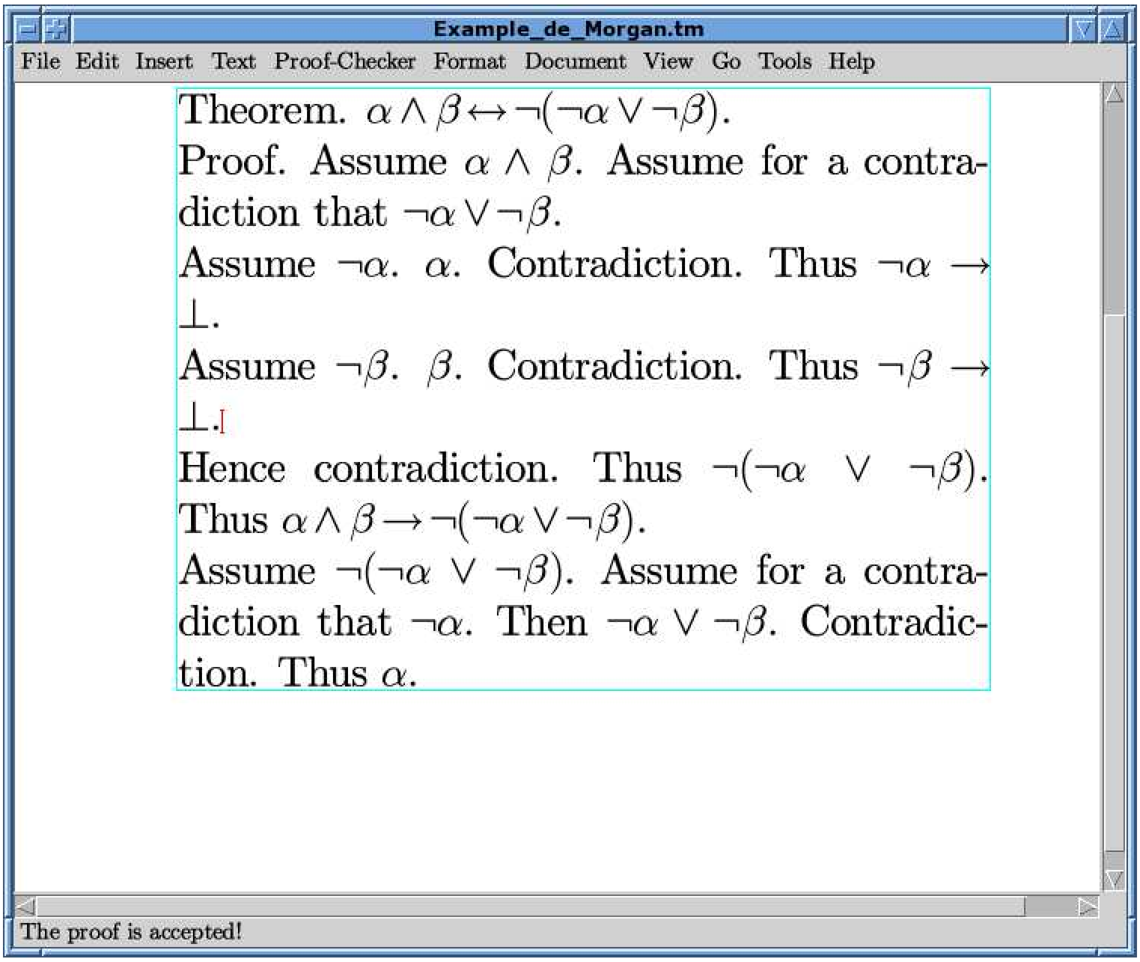
In particular, the project aims at developing the NAPROCHE *system*:

- The NAPROCHE project is centered around the NAPROCHE system, a practical implementation of semantics for parts of CML
- combines standard tools for writing mathematical texts and for checking formal proofs with a DRT-orientated grammar, adapted to CML and proof checking
- linguistic issues
- mathematical issues

A quote from the initial phase of the project:

The ... project NAPROCHE aims at constructing a system which accepts a controlled but rich subset of ordinary mathematical language including TeX-style typeset formulas and transforms them into formal statements. We adapt linguistic techniques to allow for common grammatical constructs and to extract mathematically relevant implicit information about hypotheses and conclusions. Combined with proof checking software we obtain NAtural language PROof CHEckers.

A screenshot from the NAPROCHE system so far:



This is a standard $\mathrm{T_EX_{MACS}}$ window with an incorporated proof checker. The button „Proof-Checker“ initiates the checking of the current editing buffer, the result of the check is output on the status line. In the example, „The proof is accepted!“ because the argument so far is correct. Whether the argument proves the Theorem is only checked after the proof has been closed by a „Qed.“

The NAPROCHE system is a “mathematical Curt” (Blackburn and Bos, 2005). On the left-hand side is the Curt architecture, the NAPROCHE architecture on the right-hand side has an adding typesetting layer at the top.

Input text (“`Every man dies`”)

↕ Parser, Tokenizer (readline.pl)

Tokenized format (`[all, man, die]`)

↕ NLP (natural language processing)

Internal representation (DRS)

↕

FOL format (`all(A,imp(man(A),die(A)))`)

↕ (fol2otter.pl, fol2mace.pl)

Input format for theorem prover / model builder

Typeset mathematical text

↕

LaTeX-style format / XML format

↕

Tokenized format

↕

Internal representation

↕

First-order logic format

↕

Input format for theorem prover/checker

The first NAPROCHE prototype realized this concept as follows:

– TeXmacs + all other layers implemented in home-grown PROLOG

– simple keyword language

– Theorem / Proof / Qed construct

– no explicit references to assumptions and lemmas

– only simple proof rules

The currently developed version of NAPROCHE will use standard tools as follows:

Typeset mathematical text

↕ TeXmacs

adapted XML format

$\updownarrow$ Parser, Tokenizer

Tokenized format

$\updownarrow$ XML → PRS Grammar

Proof representation structures

$\updownarrow$ PRS → FOL

First-order logic format

$\updownarrow$ (fol2otter.pl)

Input format for theorem prover Otter

$\updownarrow$ Otter

accepted / not accepted

Discussion of possible provers and proof checkers:

- proof checking can be done e.g. MIZAR or a home-grown Prolog checker
- proof checking amounts to proving every statement from available premises and methods, with e.g. strong provers like Otter, Bliksem
- Problem: how to determine the available premises
    - explicit declaration of premises: *By Theorem 5.7 ...*
    - underspecified declarations which can be resolved in context: *By induction hypothesis ...*

- closely preceding statements

- Solution (?): define a metric between statement in text and
  background knowledge, use premises with small distance

The crucial device for the complete system will be an extended DRT format: PRS = proof representation structures. A formal grammar transforms XML texts into the PRS semantics. This follows Blackburn-Bos, with mathematical features added.

The current system interface is given by the mathematical text editor TeXmacs:

- WYSIWYW LaTeX-quality text editor

- uses the TeX and LaTeX algorithms and font handling

- developed since 1999 by Joris van der Hoeven

- `www.texmacs.org`

- extendable system with scheme/guile as extension language

- can be used as an interface to other programs and for NAPROCHE

The TeXmacs format is a sort of markup language:

Theorem. $(\neg\varphi \vee \psi) \rightarrow (\varphi \rightarrow \psi)$.
Proof.

Let $(\neg\varphi\vee\psi)$.
Let $\neg\varphi$. Let φ. Contradiction. ψ. Thus $\varphi\rightarrow\psi$. Thus $\neg\varphi\rightarrow(\varphi\rightarrow\psi)$.
Let ψ. Let φ. ψ. Thus $\varphi\rightarrow\psi$. Thus $\psi\rightarrow(\varphi\rightarrow\psi)$.
$\varphi\rightarrow\psi$. Thus $(\neg\varphi\vee\psi)\rightarrow(\varphi\rightarrow\psi)$.
Qed.

Internal representation (.tm file)

```
<TeXmacs|1.0.6>
<style|generic>
<\body>
  Example:
  <\quotation>
    Theorem. <with|mode|math|(\<neg\>\<varphi\>\<vee\>\<psi\>)\<rightarrow\>
             (\<varphi\>\<rightarrow\>\<psi\>)>.\
    Proof.
    Let <with|mode|math|(\<neg\>\<varphi\>\<vee\>\<psi\>)>.
    Let <with|mode|math|\<neg\>\<varphi\>>. Let <with|mode|math|\<varphi\>>.
    Contradiction. <with|mode|math|\<psi\>>. Thus
    <with|mode|math|\<varphi\>\<rightarrow\>\<psi\>>. Thus

<with|mode|math|\<neg\>\<varphi\>\<rightarrow\>(\<varphi\>\<rightarrow\>\<psi\>)>.
    ...
```

The further plans for the development of the NAPROCHE system include:

- declaration of premises („By Lemma ...“)
- definition of new symbols („Define a function ...“)
- set theoretic approach: formulas and abstraction terms $\{x|\varphi\}$; efficient handling of terms using ”lazy expansions”
- ellipses („the sequence $x_1, ..., x_n$“)
- providing background knowledge to the checker, e.g., on natural numbers or finite sequences

- formalization of interesting theories

This involves many logical aspects like:

- identify deduction rules actually used in CML; these rules might constitute a truely natural deduction calculus

- "for all $i \in I$ *choose* a_i such that ..."

- deal with the dynamic phenomenon that brackets are usually explicitely opened but often not closed

There is a range of possible applications like:

- formalization of basic/interesting mathematical domains:
  "*Logic for man and machines*"

- tutorial applications

- natural language interfaces to provers and proof checkers

- distinguishing explicit and implicit knowledge in mathematical practice

- ...