Legible Formal Mathematics based on SAD

Peter Koepke, University of Bonn, Germany

joint work with Steffen Frerix

Cambridge, May 22, 2018

- The language of mathematics
- The System for Automated Deduction, SAD
- Improving SAD
- Examples
- Discussion

The language of mathematics

– Common mathematical language: natural language + symbolic phrases

– First-order formalizability + completeness theorem $\Rightarrow$ phrases correspond to first-order constructs

– Although mathematicians use a rich language, the language could in principle be reduced to a simple (natural) language

Structuring of theories and arguments

- Chapters, sections, subsections, paragraphs, definitions, theorems, proofs, ...; numbering

- Mathematical typesetting (LaTeX) provides such mechanisms

Sentences

– Usually correspond to formal statements

– (Common) nouns, noun phrases, adjectives, verbs, prepositions, ...

– Formal linguistics models these building blocks as logical entities

– Formal grammars can translate natural language sentences into first-order formulas

Common nouns, notions

- a *function*, *functions*, a *set*, a *triangle*, ...
- Modeling *triangle* as a predicate $T(.)$ - $T(\mathrm{ABC}) \rightarrow \ldots$
- Modeling *triangle* as a type - $\mathrm{ABC} :: T \ldots$
- Compromise: notion = soft type, sometimes with a first-order definition from more basic types
- Notions determine „small worlds“ for mathematical arguments, where definitions of notions ideally need not be expanded; sometimes, however, they have to be expanded

Adjectives and verbs

- *infinite*, *bijective*, *regular*, ...
- Adjectives are modeled by formulas $\varphi(x)$
- Can be used to modify notions (*isosceles triangle*), but do not need a type of their own
- In mathematical contexts, verbs are similar to adjectives, with arities $\geqslant 1$ (*a line A bisects an angle B*)

Patterns

- linguistic patterns define the interplay between variables

- standard meanings of words may be suggestive, but technically not essential: *group*, *ring*, *field*, *real number*, *complex number*, ...

- Words are placeholders like Hilbert's *Stuehle, Tische, Bierseidel*

- One can liberally employ syntactic patterns without attention to the meaning of components; internal meaning is regulated through syntax and axioms

Evidence Algorithm

V.M. Glushkov – 1966 – Institute of Cybernetics – Kiev, Ukraine

Task: assistance to a working mathematician

Form: mathematical text processing, proof verification

Research:

- formal languages for mathematical text's presentation
- deductive routines which determine what is «evident»
- information environment, a library of mathematical knowledge
- interactive proof search

Principles:

- closeness to a natural language
- closeness to a natural reasoning

Developed:

- languages of formal theories
- goal-driven sequent calculi
- ...

Result: System for Automated Deduction (SAD) — 1978, 2003

SAD example text

[integer/-s] [program/-s] [code/-s] [succeed/-s] [decide/-s] [halt/-s]

Signature PrgSort. A program is a notion. Let U,V stand for programs.

Signature IntSort. An integer is a notion. Let x,y,z stand for integers.

Signature CodeInt. A code of W is an integer.

Axiom ExiCode. Every program has a code.

Signature HaltRel. W halts on x is an atom.

Signature SuccRel. W succeeds on x and y is an atom.

Definition DefDH. W decides halting iff

for every z and every code x of every V

W succeeds on x and z iff V halts on z.

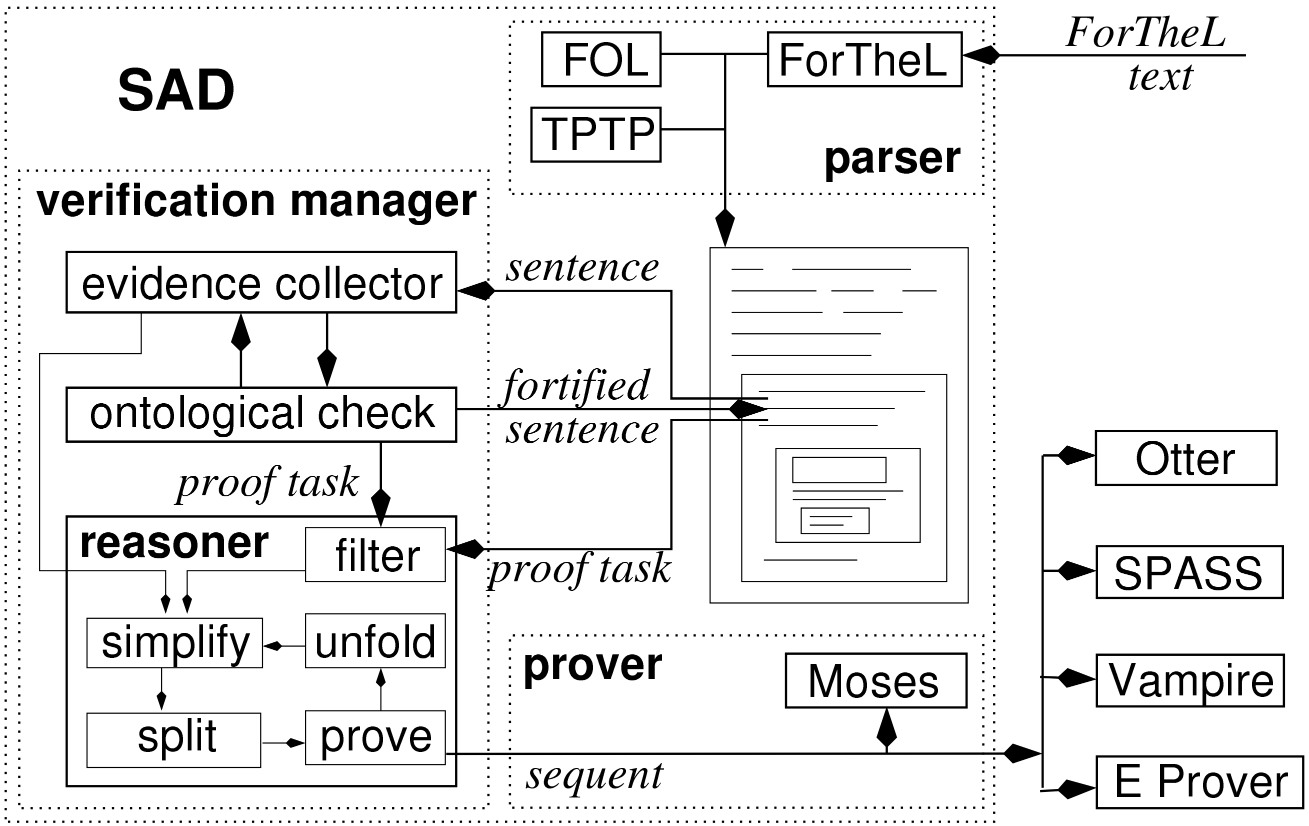
Axiom Cantor. Let W decide halting.

Then there exists V such that for every y

V halts on y iff W does not succeed on y and y.

Proposition. No program decides halting.

System for Automated Deduction



- **manager**: decompose input text into separate proof tasks
- **reasoner**: big steps of reasoning, heuristic proof methods
- **prover**: inference search in a sound and complete calculus

Pakevich's SAD 2.3

- prototype for PhD project; not continued thereafter
- complexity problems; only short texts („miniatures“) checkable
- sets and functions only partially implemented
- compact Haskell code
- ....

Formalizations in „SAD 3.0“

Let q^2 stand for $q * q$.

Theorem 1. *$p = q^2$ for* no *rational number q.*

Proof. Assume the contrary. Take a rational number q such that $p = q^2$. Take coprime m, n such that $m * q = n$. Then $p * m^2 = n^2$. Therefore p divides n. Take a natural number k such that $n = k * p$. Then $p * m^2 = p * (k * n)$. Therefore $m * m$ is equal to $p * k^2$. Hence p divides m. Contradiction. $\square$

Recent formalizations

Theorem 2. κ^+ *is regular.*

Proof. Assume the contrary. Take a cofinal subset x of κ^+ such that $\mathrm{card}(x) \neq \kappa^+$. Then $\mathrm{card}(x) \leq \kappa$. Take a function f that is surjective from κ onto x. Define

$$g[i] = \text{Choose a function } h \text{ that is surjective from } \kappa \text{ onto } i \text{ in } h$$

for $i \in \kappa^+$. Define

$$h[(\xi, \zeta)] = g[f[\xi]][\zeta]$$

for $(\xi, \zeta) \in \kappa \times \kappa$. Let us show that h is surjective from $\kappa \times \kappa$ onto κ^+.

$\mathrm{Dom}(h) = \kappa \times \kappa$. Every element of κ^+ is an element of $h\,[\![\kappa \times \kappa]\!]$.

Proof. Let α be an element of κ^+. Take an element ξ of κ such that $\alpha < f[\xi]$. Take an element ζ of κ such that $g[f[\xi]][\zeta] = \alpha$. Then $\alpha = h[(\xi, \zeta)]$. Therefore the thesis. Indeed (ξ, ζ) is an element of $\kappa \times \kappa$. end.

Every element of $h\,[\![\kappa \times \kappa]\!]$ is an element of κ^+.

Proof. Let α be an element of $h\,[\![\kappa \times \kappa]\!]$. We can take elements ξ, ζ of κ such that $\alpha = h[(\xi, \zeta)]$. Therefore the thesis (by Transitivity). end. end.

Therefore $\kappa^+ \leq \kappa$. Contradiction. $\square$

Improvements

- Eliminated severe inefficiencies in the reasoning process; „chapter-sized“ texts can now be checked
- Improved handling of notions: using notions as types, if possible (no typeguards); „elimination of types“ reminiscent of Sledgehammer
- Improved inbuilt notions for sets and functions; these notions correspond to ZFC set theory
- Exchanged parts of the code by own commented code which should be easier to maintain
- Initial processing allows $\LaTeX$-input

Proof-checked SAD *texts*

- Texts in a *controlled natural language* (CNL)
- Logical meaning of a proof-checked text:

$$\bigwedge \text{premises} \Longrightarrow \bigwedge \text{conclusions}$$

- A text captures a sequence of mathematical arguments
- Texts may be linked by insertion or other processes
- A text may, e.g., construct notions that are used axiomatically in another text
- Interlinked collections of texts that may be founded on the initial axiomatic assumptions of SAD, like ZFC

Further work

- Implementing more standard notions within the system: ordered pairs and tuples, structures („a group consists of ...“), inductive data types, ..., natural numbers, ...

- Automatic „notion derivation“, like type derivation in Haskell

- Development environment for SAD like the Isabelle environment

- Formalizations: miniatures and chapters from textbooks

Text-orientated formal mathematics

- Embedding SAD into usual mathematical documents through some `forthel` environment; partial proof-checking of documents

- Enriching the ForTheL/SAD language by further constructs, based on analyzing standard mathematical texts

- Replacing the *ad hoc* parsing by proper parser library with linguistically sound grammars and dictionary

- Expanding the metalanguage within SAD; making „statements about statements“

- ...

Issues

- Relating to other formal mathematics systems
- Difficulties of interpretations of texts: the human interpretation of the natural language SAD text might differ from the interpretation by the SAD system
- What is a correct formalization of something?
- ...

Thank You!