Naproche-ZF

Adrian De Lon
adelon@uni-bonn.de
Mathematical Logic Group, University of Bonn

AITP'24: 9th Conference on Artificial Intelligence and Theorem Proving
5 September, 2024, Aussois, France

Formalization examples

Example: Cantor's theorem formalized in Naproche-ZF

Theorem (Cantor). There exists no surjection from A to 2^A.
Proof. Suppose not. Consider a surjection f from A to 2^A. Let $B = \{a \in A \mid a \notin f(a)\}$. Then $B \in 2^A$. There exists $a' \in A$ such that $f(a') = B$. Now $a' \in B$ iff $a' \notin f(a') = B$. Contradiction.

Example: proof tasks from proof gaps

Theorem (Cantor). There exists no surjection from A to 2^A.
Proof. Suppose not. Consider a surjection f from A to 2^A. Let $B = \{a \in A \mid a \notin f(a)\}$. Then $B \in 2^A$. There exists $a' \in A$ such that $f(a') = B$. Now $a' \in B$ iff $a' \notin f(a') = B$. Contradiction.

Negated conjecture from "Then $B \in 2^A$"	$B \notin 2^A$
Global premise from a lemma	$\forall X.\, \forall Y.\, X \subseteq Y \implies X \in 2^Y$
Global premise from a definition	$\forall X. \forall X.\, X \subseteq Y \iff (\forall x.\, x \in X \implies x \in Y)$
$\vdots$	$\vdots$
Local premise from "Let $B = \cdots$"	$\forall a.\, a \in B \iff (a \in A \wedge a \notin f(a))$
Local premise from "Consider ..."	$f \in \mathsf{Surj}(A, 2^A)$
Local premise from "Suppose not"	$\exists g.\, g \in \mathsf{Surj}(A, 2^A)$

Example: proof tasks from proof gaps

Theorem (Cantor). There exists no surjection from A to 2^A.
Proof. Suppose not. Consider a surjection f from A to 2^A. Let $B = \{a \in A \mid a \notin f(a)\}$. Then $B \in 2^A$. There exists $a' \in A$ such that $f(a') = B$. Now $a' \in B$ iff $a' \notin f(a') = B$. Contradiction.

Negated conjecture from "Then $B \in 2^A$"	$B \notin 2^A$
Global premise from a lemma	$\forall X.\ \forall Y.\ X \subseteq Y \implies X \in 2^Y$
Global premise from a definition	$\forall X. \forall X.\ X \subseteq Y \iff (\forall x.\ x \in X \implies x \in Y)$
$\vdots$	$\vdots$
Local premise from "Let $B = \cdots$"	$\forall a.\ a \in B \iff (a \in A \wedge a \notin f(a))$
Local premise from "Consider ..."	$f \in \mathsf{Surj}(A, 2^A)$
Local premise from "Suppose not"	$\exists g.\ g \in \mathsf{Surj}(A, 2^A)$

Example: proof tasks from proof gaps

Theorem (Cantor). There exists no surjection from A to 2^A.
Proof. Suppose not. Consider a surjection f from A to 2^A. Let $B = \{a \in A \mid a \notin f(a)\}$. Then $B \in 2^A$. There exists $a' \in A$ such that $f(a') = B$. Now $a' \in B$ iff $a' \notin f(a') = B$. Contradiction.

Negated conjecture from "Then $B \in 2^A$"	$B \notin 2^A$
Global premise from a lemma	$\forall X.\, \forall Y.\, X \subseteq Y \implies X \in 2^Y$
Global premise from a definition	$\forall X. \forall X.\, X \subseteq Y \iff (\forall x.\, x \in X \implies x \in Y)$
$\vdots$	$\vdots$
Local premise from "Let $B = \cdots$"	$\forall a.\, a \in B \iff (a \in A \wedge a \notin f(a))$
Local premise from "Consider ..."	$f \in \mathsf{Surj}(A, 2^A)$
Local premise from "Suppose not"	$\exists g.\, g \in \mathsf{Surj}(A, 2^A)$

Example: formalizing in natural language embedded in LaTeX

Theorem (Burali-Forti antimony) There exists no set Ω such that for all α we have $\alpha \in \Omega$ iff α is an ordinal.
Proof. Suppose not. Consider Ω such that for all α we have $\alpha \in \Omega$ iff α is an ordinal. For all x, y such that $x \in y \in \Omega$ we have $x \in \Omega$. So Ω is $\in$-transitive. Thus Ω is an ordinal. Hence $\Omega \in \Omega$. Contradiction. □

```
\begin{theorem}[Burali-Forti antimony]\label{burali_forti}
    There exists no set $\Omega$ such that
    for all $\alpha$ we have $\alpha\in \Omega$ iff $\alpha$ is an ordinal.
\end{theorem}
\begin{proof}
    Suppose not.
    Consider $\Omega$ such that for all $\alpha$ we have
        $\alpha\in \Omega$ iff $\alpha$ is an ordinal.
    For all $x, y$ such that $x\in y\in\Omega$ we have $x\in\Omega$.
    So $\Omega$ is \in-transitive. Thus $\Omega$ is an ordinal.
    Hence $\Omega\in\Omega$. Contradiction.
\end{proof}
```

Example: phase transition to controlled natural language

Informal statement from T. Jech, *Set Theory*, Ex. 24.3:
If $2^{\aleph_\alpha} \leq \aleph_{\alpha+2}$ holds for all cardinals of cofinality ω,
then the same holds for all singular cardinals.

Formalized statement in controlled language:
If $2^{\aleph_\alpha} \leq \aleph_{\alpha+2}$ for all cardinals α of cofinality ω,
then $2^{\aleph_\beta} \leq \aleph_{\beta+2}$ for all singular cardinals β.

Formal translation to first-order form:
$(\forall \alpha.\, \mathrm{Card}(\alpha) \wedge \mathrm{cf}(\alpha) = \omega \rightarrow 2^{\aleph_\alpha} \leq \aleph_{\alpha+2})$
$\rightarrow (\forall \beta.\, \mathrm{Sing}(\beta) \rightarrow 2^{\aleph_\beta} \leq \aleph_{\beta+2})$

Natural language understanding

Grammar-oriented approach to NLU: grammar fragment parametrized by lexical items

Dynamic: patterns for lexical items

noun → set | group | function from *term* to *term* | *term*-ary relation | ⋯

mixfix-operator → *expr* + *expr* | ⋃ *expr* | *expr*! | ⟨*expr*, *expr*⟩ | ⋯

adjective → even | continuous | *term*-close | (*expr*, *expr*)-provable | ⋯

relator → = | ∈ | < | $\cong_{expr}$ | ⋯

Static: phrases, sentences, blocks

noun-phrase → *adjective-list noun attribute such-that-statement*

such-that-statement → such that *statement* | ε

statement → *not* | *if* | *iff* | *xor* | *nor* | *exists* | *quantified-phrase* | ⋯

atomic-statement → *formula* | *noun-statement* | *verb-statement* | *adjective-statement* | ⋯

noun-statement → *term* is a *noun-phrase* | *term* is not a *noun-phrase*

let → let *var* be a *noun phrase*. | let *var-list* ∈ *expression*.

assumption-list → suppose *statement*. *assumption-list* | *let assumption-list* | ε

theorem → *assumption-list statement*.

Extracting lexical items

Naproche-ZF extracts token patterns of lexical items from definitions. We can use the context of the definition to distinguish between nouns, verbs, adjectives, &c.

Smart paradigms à la GF (Grammatical Framework) are used to guess plural forms of nouns and verbs.

Examples:

"f is a function from X to Y iff ..."
$\rightsquigarrow$ noun: function[/s] $(-_0)$ from $(-_1)$ to $(-_2)$

"A is a matrix over k iff ..."
$\rightsquigarrow$ noun: matri[x/ces] $(-_0)$ over $(-_1)$

"x is δ-close iff ..."
$\rightsquigarrow$ adjective: $(-_1)$-close

"m divide[s/] n iff ..."
$\rightsquigarrow$ verb: divides $(-_1)$

"$m < n$ iff ..."
$\rightsquigarrow$ relation symbol: $<$

"$x \cup y = \cdots$"
$\rightsquigarrow$ infix function symbol: $\cup$

Examples: basic syntactic transformation

$a, b < c < d$
$\rightsquigarrow a < c \wedge b < c \wedge c < d$

$x \; R \; y$
$\rightsquigarrow (x, y) \in R$

For all $a < b$ such that $P(a)$ we have $Q(a)$.
$\rightsquigarrow$ For all a we have if $a < b$ and $P(a)$, then $Q(a)$.

There exists x such that ...
$\rightsquigarrow$ Consider x such that ....

Every set is an element of some Grothendieck universe.
$\rightsquigarrow$ For all sets x there exists a Grothendieck universe U such that x is an element of U.

Improvements over Naproche in NLU

Remove ambiguities from the language by distinguishing math and text mode in LaTeX (e.g. variable “a” vs article “a”) and by enforcing number agreement.

Earley’s algorithm guarantuess better asymptotic behaviour than backtracking monadic parser combinators (cubic vs. exponential).

Declarative grammar specification is easier to extend.

Literate formalization

Only content of *formal environments* such as `definition`, `theorem`, and `proof` is checked by the system. Everything else is treated as informal commentary.

One can freely mix informal and formal material in the same document.

The formal material is already readable and does not need to be restated in informal language (less clutter and no issues with syncing).

Grammar-oriented approach to NLU: why not use machine learning?

A lot of expressivity of mathematical writing is in coining terminology and notational patterns, which are easy to capture in a parametrized grammar.

Restrictions by the grammar are mainly felt when writing, not when reading.

Clear semantics in the translation to FOL or HOL.

Clear failure conditions.

Potential for better error messages and editor tooling.

Takeaways

Formalizations in controlled natural language still function as mathematical texts even for those unfamiliar with the system.

Language extensions are tempting, but a small and strict language suffices. It also avoids an *uncanny valley effect*.

Don't have quasi-synonymous words mean different things ("thus", "hence", "so", &c).

A controlled natural language is at least as easy to learn as a semantically equivalent formal language, but it is less intimidating to novice users and one has a higher chance of intuitively using the right syntax.

LaTeX embedding is nice to have.

Natural proof vernacular and foundations

Proof gaps: what counts as a proof?

"I have discovered a truly marvellous proof of this, which this margin is too narrow to contain."

"Left as an exercise to the reader."

"Follows by induction."

"Follows from an easy diagram chase."

"Easy (using 1.5 of course)."

"Check (part 3 is like 3.6)."

"Think."

Proof rules in Naproche-ZF: some terminals

Γ is the set of global premises, containing all previous theorems and definitions.

Λ is the set of local premises, containing local assumptions and claims from previous proof steps.

Γ^{sel} is a subset of Γ after optional premise selection.

$$\frac{\Gamma^{\mathsf{sel}}; \Lambda \vdash^{\mathsf{ATP}} \varphi}{\Gamma; \Lambda \vdash \varphi}\ \square$$

$$\frac{\Lambda, \gamma_1, \ldots, \gamma_k \vdash^{\mathsf{ATP}} \varphi}{\Gamma; \Lambda \vdash \varphi}\ \text{by } \gamma_1, \ldots, \gamma_k \in \Gamma$$

$$\frac{\Lambda \vdash^{\mathsf{ATP}} \varphi}{\Gamma; \Lambda \vdash \varphi}\ \text{by assumption}$$

$$\frac{\Gamma^{\mathsf{sel}}; \Lambda \vdash^{\mathsf{ATP}} \forall x.\ x \in X \longleftrightarrow x \in Y}{\Gamma; \Lambda \vdash X = Y}\ \text{setext}$$

Proof rules in Naproche-ZF: some intermediate steps

$$\frac{\Gamma; \Lambda, \varphi \vdash \psi \qquad \Gamma^{\mathsf{sel}}; \Lambda \vdash^{\mathsf{ATP}} \varphi}{\Gamma; \Lambda \vdash \psi}\ \text{have } \varphi \qquad \frac{\Gamma; \Lambda \vdash \varphi \qquad \Gamma^{\mathsf{sel}}; \Lambda, \varphi \vdash^{\mathsf{ATP}} \psi}{\Gamma; \Lambda \vdash \psi}\ \text{suffices } \varphi$$

$$\frac{\Gamma; \Lambda, \varphi \vdash \psi}{\Gamma; \Lambda \vdash \varphi \to \psi}\ \text{assume } \varphi \qquad \frac{\Gamma; \Lambda, \varphi \vdash \bot}{\Gamma; \Lambda \vdash \neg\varphi}\ \text{assume } \varphi \qquad \frac{\Gamma; \Lambda, \neg\varphi \vdash \bot}{\Gamma; \Lambda \vdash \varphi}\ \text{suppose not}$$

$$\frac{\Gamma; \Lambda, \varphi_1 \vdash \psi \quad \cdots \quad \Gamma; \Lambda, \varphi_k \vdash \psi \qquad \Gamma^{\mathsf{sel}}; \Lambda \vdash^{\mathsf{ATP}} \bigvee_i \varphi_i}{\Gamma; \Lambda \vdash \psi}\ \text{cases } \varphi_1, \ldots, \varphi_k$$

$$\frac{\Gamma; \Lambda, a \vdash \varphi(a)}{\Gamma; \Lambda \vdash \forall a. \varphi(a)}\ \text{let } a \qquad \frac{\Gamma \vdash \forall a. (\forall b. b \in a \to \varphi(b)) \to \varphi(a)}{\Gamma; \Lambda \vdash \forall c. \varphi(c)}\ \in\text{-induction}$$

$$\frac{\Gamma; \Lambda, a, \psi(a) \vdash \varphi \qquad \Gamma^{\mathsf{sel}}; \Lambda \vdash^{\mathsf{ATP}} \exists a. \psi(a)}{\Gamma; \Lambda \vdash \varphi}\ \text{consider } a \text{ such that } \psi(a)$$

Differences between Naproche and Naproche-ZF: foundations

Naproche uses Kelley–Morse class theory with urelements in first-order logic.

Naproche-ZF uses a Zermelo–Fraenkel-like theory of pure sets, such as ZF, ZFC, or Tarski–Grothendieck, but in higher-order logic.

Naproche-ZF drops ontological checking, which had performance problems.

Simpler ontology does not need typeguards for classes and urelements anymore, keeping trivial statements about sets trivial for the ATP.

The more standard ontology makes the system easier to teach since mathematics students and mathematicians are often already familiar with Zermelo–Fraenkel set theory.

Practical concerns

Naproche-ZF as a scalable continuation of Naproche/SAD

Naproche-ZF is a continuation of Naproche (which is a continuation of SAD) and has the same main goal: check proofs that resemble natural language mathematics as closely possible. It...

...scales beyond chapter-sized formalizations with ad hoc axiomatic preliminaries by using concurrency, avoiding redundant checking of shared imports, having better asymptotic parsing behaviour, &c.

...adds new features: structures, inductive definitions, support for higher-order logic, &c.

...has various tweaks and improvements to the controlled language, making it a more accurate model of mathematical English with fewer ambiguities.

...has a grammar-oriented implementation that makes the controlled language easier to extend.

...uses smart paradigms instead of manual specification of synonyms.

Performance matters (a natural proof assistant is still a proof assistant)

You should make a compromise between naturalness and performance when formalizing.

Proper caching makes the biggest difference and Naproche-ZF adds caching between runs.

Keeping checking times fast while formalizing makes it easier to upgrade to newer versions of ATPs (because you have more wiggle room within a given time limit).

Transparency

You cannot (and should not) hide everything under a natural language interface.

Naproche-ZF has a clearer separation of stages.

The translation to formal logic should be straightforward/unsurprising (e.g. it's better to report an ambiguity error than to arbitrarily disambiguate).

The resulting formulas should be easy to inspect (no more names like "`zpzlzuzs`").

Hard problem: how to present ATP proofs to the user?

Outlook

Early experiments with controlled language as target for autoformalization with LLMs

Has the potential to work better than autoformalizations to other formal languages, since LLMs have seen much more natural language than formal mathematics (even if you only count mathematical English).

Restricting to controlled language via prompting works OK.

Non-controlled sentences generated by the LLM could indicate where to extend the grammar.

One could also use grammar augmentation (via syntactically constrained sampling) to force controlled natural language output.

Other difficulties of autoformalization remain: global coherence, implicitness, high-level informal reasoning, &c.

Outlook

Experiment with integrating Naproche ideas into Mizar.

Formalizations in set theory, general topology, and algebra.

Bug fixes, IDE features, more library work, &c.

Preview: bidirectional translation between Mizar and controlled natural language

```
definition let X,Y;
    pred X c= Y means for x being object holds x in X implies x in Y;
```

Definition. Let X, Y be sets. $X \subseteq Y$ iff for all objects x such that $x \in X$ we have $x \in Y$.

```
theorem for C being countable Language, phi wff string of C, X being set st
    X c= AllFormulasOf C & phi is X-implied holds phi is X-provable
```

Theorem (Completeness Theorem). Let L be a countable language, φ a wellformed L-formula, and Γ a set of L-formulas such that $\Gamma \vDash \varphi$. Then $\Gamma \vdash \varphi$.

```
theorem Th19:
  for T being non empty normal TopSpace, A,B being closed Subset of T st
  A <> {} & A misses B holds ex F being Function of T,R^1 st
  F is continuous & for x being Point of T holds 0 <= F.x & F.x <= 1 &
  (x in A implies F.x = 0) & (x in B implies F.x = 1)
```

Theorem (Urysohn). Let T be a non-empty normal space. Let A, B be closed subsets of T such that $A \neq \emptyset$ and $A \cap B = \emptyset$. Then there exists a continuous function f from T to $\mathbb{R}$ such that for all points x of T we have $0 \leq f(x) \leq 1$ and $x \in A \implies f(x) = 0$ and $x \in B \implies f(x) = 1$.

Thank you!