ForTheL for Type Theory

Adrian De Lon, Peter Koepke, and Anton Felix Lorenzen

University of Bonn

Naproche-SAD [Nap] accepts mathematical input texts in the ForTheL language, translates them into first-order logic and checks for logical correctness. The controlled natural language ForTheL allows statements close to the language of mathematical textbooks. The system supports natural structurings of proofs, using strong ATPs to deal with tedious details. Altogether, it is possible to write proof documents which are *natural* and immediately readable by mathematicians. Naproche-SAD has been presented at the two previous editions of AITP.

Other proof assistants have realized that broader adoption in the mathematical community would require a *readable* input language. Mizar and Isabelle have declarative proof languages that mimic ordinary proof structures and provide some readability. It seems, however, that even more naturality is required for wider acceptance [Hal19].

Since most popular and successful proof systems are based on type theories this calls for a CNL for type theory, with additional automation to reach acceptable proof structures and granularities. Our presentation will be based on the project described in the final section below.

1. The ForTheL language. ForTheL is a (subset of) the natural language of mathematics and therefore amenable to a type-theoretical analysis [Gan13,Ran94]. Common nouns like *man* or *number* can be viewed as types, and proper nouns like *Aristotle* or 15 as inhabitants of those types. Predicates or adjectives like *mortal* or *odd* can be used to modify types. Natural language quantification commonly uses nouns with implicit variables as in *"Every man is mortal"*.

Similarly, ForTheL's parser categorizes phrases as nouns, verbs, and adjectives and provides subparsers to identify specific phrases (= patterns) in the input. The ForTheL language centers around *notions*, which can be viewed as weak types. Notions are introduced as noun phrases by `Signature` and `Definition` commands:

```
Signature. A natural number is a notion.
Definition. Assume that n is a natural number.
A divisor of n is a natural number m such that ...
```

2. First-order parsing in Naproche-SAD. Linguistically, this ForTheL text introduces a type `naturalNumber` and a dependent type `divisorOf(x)`. The patterns `[natural, number]` and `[divisor, of, ?]` are stored as assignments `[natural, number] → naturalNumber(x)` and `[divisor, of, ?] → divisorOf(x,y)` of word patterns to first-order predicate symbols and are used in the further parsing of the text. A parser `notion` then looks for previously defined notions, allowing us to parse `0 is a natural number` as its first-order equivalent `naturalNumber(0)`. Similarly, the statement `Every natural number has a divisor` leads to `forall x (naturalNumber(x) -> exists y divisorOf(x,y)`.

3. Towards a type-theoretic parsing. First-order parsing views notions as unary predicates and dependent notions as n-ary predicates with $n \geq 2$, corresponding to standard first-order renderings of type-theoretic formulas. By a simple syntactic reification we turn the predicates into a type `naturalNumber` and a dependent type `divisorOf(x)`. Now `0 is a natural number` can be translated to the judgment `0 : naturalNumber` and `Every natural number has a divisor` to `(forall x : naturalNumber) (exists y : divisorOf(x))`.

4. Type-theoretic foundations and axiomatic style. Type theory ideally builds up the mathematical edifice from first principles. This creates types modelling common mathematical structures in an intuitionistic and constructive framework. Many mathematical properties and

proofs can however be understood axiomatically, proceeding from assumed given types and their properties to theorems and proofs. Locally this is also true of type-theoretic libraries like LEAN's mathlib [Com].

5. ForTheL, mathlib and fabstracts. We have reformulated typical entries of mathlib in ForTheL: the introduction of groups and commutative groups, e.g., in the mathlib file `groups.lean` reads as follows:

```
class group (A : Type u) extends monoid A, has_inv A :=
    mul_left_inv : forall a : A, inv a * a = 1
class comm_group (A : Type u) extends group A, comm_monoid A
```

and can be reformulated in ForTheL, using some LaTeX pretty printing:

Definition 1. *A* group *is a monoid* α *such that* α *is a type with inverses and for all* $a : \alpha$ $a^{-1,\alpha} *^{\alpha} a = 1^{\alpha}$.

Definition 2. *A* commutative group *is a group that is a commutative monoid.*

The ForTheL translation of such statements can straightforwardly be modified to generate valid LEAN input. Interestingly, the ForTheL text describing some elementary lemmas on groups and similar structures immediately proof-checks in the first-order Naproche-SAD system with E as an external ATP.

A CNL for CiC.

We have recently started working on a new ForTheL-like CNL as a front-end for the calculus of inductive constructions (CiC). We have reconsidered some architectural decisions, which should allow for faster performance, more flexibility and maintainability, and improved error messages. Unlike the current implementation of Naproche-SAD, we will have a clear data structure representing the parse tree, so that the translation process becomes more transparent. This CNL will feature new phrases that will allow the user to interact with the underlying type theory in a natural manner, departing from the mostly first-order semantics of ForTheL. One of our goals is to write documents that translate to LEAN code equivalent to some fragments of mathlib and fabstracts.

Our presentation will discuss relevant mathematical linguistics and demonstrate a prototype implementation of the new CNL. We expect to be able to show a new parser and a translator from a mostly declarative fragment of natural mathematical language to CiC. We will also discuss details of implementation and design of such a language, and show some mathematical examples.

References

[Com] LEAN Community. mathlib. https://github.com/leanprover-community/mathlib.

[Gan13] Mohan Ganesalingam. *The language of mathematics.* Springer, 2013.

[Hal19] Thomas Hales. Mathematical Definitions, Formally Speaking. `https://www.icms.org.uk/downloads/bigproof/Hales.pdf`, 2019.

[Nap] Naproche-SAD in Isabelle-jEdit. https://files.sketis.net/Isabelle_Naproche-20190611/.

[Ran94] Aarne Ranta. Type theory and the informal language of mathematics. In *Types for Proofs and Programs. Types 1993*, pages 352–365. Springer, 1994.