# A Controlled Natural Language for Type Theory[*]

Thomas Hales

University of Pittsburgh
Pittsburgh, PA, USA

## 1 Introduction

This abstract describes the design and development of a controlled natural language for mathematics that has the Lean theorem prover as intended target. We call this language Colada (short for *Co*ntrolled *la*nguage *da*ta). Documents in our dialect are written in a specially prepared LaTeX file. Our aim is to capture definitions and theorem statements from the published mathematical literature in our dialect, but checking mathematical proofs is beyond the scope of our project.

Our design grows out of previous controlled natural languages for mathematics (specifically, Forthel-Naproche-SAD), as described in Peter Koepke's AITP 2019 talk, which exhibited some short proofs written in fluent English that can be read and checked by their software [FK19], [Pas07], [LPV04].

We use Forthel as the generic name for any of the dialects inspired by Forthel (a controlled natural language developed by various researchers starting with Glushkov's *Evidence Algorithm*, and implemented in Paskevich's PhD thesis), including Colada. We refer to the Colada language as *our dialect*. Our dialect differs from others in that our semantic target is the logical Calculus of Inductive Constructions (CiC) as implemented in the Lean theorem prover, instead of first-order logic [dMKA+15]. Our dialect can be viewed as a fusion of three different syntactic traditions: Forthel syntax, LaTeX syntax, and Lean theorem-prover syntax. From another perspective, our dialect might be viewed as a mountain of syntactic sugar for Lean.

## 2 Controlled Natural Languages (CNL)

By controlled natural language for mathematics (CNL), we mean an artificial language for the communication of mathematics that is (1) designed in a deliberate and explicit way with precise computer-readable syntax and semantics, (2) based on a single natural language (which for us is English), and (3) broadly understood at least in an intuitive way by mathematically literate speakers of the natural language.

CNLs can achieve a much higher degree of English fluency than other proof-checking languages.

Our basic aim is to develop a technology that lies somewhere between the current practice of research mathematicians and the current practice within the proof assistant community.

[*]I thank Peter Koepke for introducing me to the field and Jesse Han for his contributions. Research is supported by Sloan Foundation grant G-2018-10067. This work is part of the Formal Abstracts project, which aims to capture all the major definitions and theorems of mathematics in a format that is both human and computer friendly. Source code and examples are found at github.com/formalabstracts/CNL-CIC.

## 3 Research to Date

Our specific research contributions to date are as follows.

We have a design and specification of a controlled natural language. Like other Forthel dialects, our grammar is not context-free. However, it is similar to a context-free grammar by being specified through production rules on terminal and nonterminal symbols. Users may extend the grammar with new mathematical notation and constructs: the language contains syntax for the extension of its own syntax.

The lexical structure of our dialect is specified in sedlex, a lexical generator tool for OCaml. Our dialect has been specified in menhir, an OCaml-based parser-generator tool for LR(1) grammars. (Although our dialect is not an LR(1) grammar, which prevents menhir from automatically generating a parser, the software checks that our grammar is well-formed.)

We believe that some complexity is justified (and even required) to capture widespread mathematical idioms and formulas, the syntax of type theory, and their interactions. Our grammar is recursive to an extraordinary degree. The grammar has about 350 nonterminals and about 550 production rules. The grammar contains about 150 English words (such as *all, any, are, case, define, exists, if, iff, is, no, not, of, or, over, proof, the, theorem*, etc.) with a fixed grammatical function. User syntax extensions build on that base.

We keep most features of Forthel, such as its handling of synonyms, noun phrases, verbs, and adjectives; and its grammar extension mechanisms. We have added many additional features such as plural formation for nouns and verbs, operator precedence parsing (with user-specified precedence levels and associativities); scoping of variables; syntax for LaTeX macros; and dependent type theory including inductive and mutual inductive types, structures, and lambda terms.

A parser for our grammar has been implemented in OCaml, building substantially on the parser combinator library that John Harrison wrote to parse HOL Light.

Future work will transform parsed output to type-checked terms in Lean: our system will output Lean-pre-expressions, which will then be processed by Lean's elaboration and type-checking procedures. Another future project is syntax highlighting and auto-completion tools for our dialect in editors such as emacs and VSCode. We also plan to develop large mathematical libraries in our dialect.

We have written software that takes a specially prepared LaTeX file as input and strips away the non-semantic content (such as headers, spaces and other layout, graphics, remarks, and dollar signs) and outputs raw CNL. The key to beautifully typeset TeX documents is a dual expansion system for macros. The TeX engine expands macros in the usual way, but the CNL engine expands macros according to an independent semantic specification.

We believe our language will find novel applications to search, document analysis, and document transformation.

## 4 Example

Examples will be given during the AITP presentation to show that English fluency is obtained without loss of semantic content. The presentation will include a discussion of dependent types, structures, inductive types, type coercions, and implicit arguments.

Here is one example that assumes a context in which a binary relation $(R, \le)$ has been defined.

### 4.1 pdf

Here is a sample text, as viewed by the mathematician reading the document.

**Definition 1** (greatest element)**.** *We say that $y$ is a* **greatest element** *in $R$ iff for all $x$, $x \le y$.*

Let $x < y$ stand for $x \le y$ and $x \ne y$.

### 4.2 source

The LaTeX source file for the pdf is similar to documents prepared every day by mathematicians.

```
\def\deflabel#1{\begin{definition}[#1]\label{#1}}

\deflabel{greatest element} We say that $y$ is a
\df{greatest\~element} in $R$ iff for all\ $x,\ x \le y$.
\end{definition}

Let $x < y$ stand for $x \le y$ and $x \ne y$.
```

### 4.3 CNL

The CNL is generated from the source file by stripping formatting.

```
Definition Label_greatest_element . We say that y is a
greatestelement in R iff for all x , x \le y .

Let x < y stand for x \le y and x \ne y .
```

### 4.4 parse tree

This parse tree is generated from the CNL. We only display the first definition and have pruned the tree for simplicity. Nonterminals are typeset in smallcaps and literals are in teletype. At hierarchical levels above those shown in the outline, we have TEXT_ITEM → DECLARATION → DEFINITION. A further transformation not shown would produce a Lean pre-expression from the tree.

- DEFINITION_PREAMBLE
  - LIT_DEF ........................................ `Definition`
  - LABEL ........................................... `Label_greatest_element`
  - PERIOD .......................................... `.`
- list(ASSUMPTION) ....................................
- DEFINITION_AFFIRM
  - DEFINITION_STATEMENT $\to$ PREDICATE_DEF
    * OPT_SAY ................................... `We say that`
    * PREDICATE_HEAD $\to$ PREDICATE_WORD_PATTERN $\to$ NOTION_PATTERN
      · TVAR ................................... $y$
      · LIT_IS .................................. `is`
      · LIT_A .................................. `a`
      · WORD_PATTERN ......................... `greatestelement in` $R$
    * IFF_JUNCTION .............................. `iff`
    * STATEMENT ................................ `for all` $x$`,` $x$ `\le` $y$
  - PERIOD .......................................... `.`

## References

[dMKA+15] Leonardo de Moura, Soonho Kong, Jeremy Avigad, Floris Van Doorn, and Jakob von Raumer. The Lean theorem prover (system description). In *International Conference on Automated Deduction*, pages 378–388. Springer, 2015.

[FK19] Steffen Frerix and Peter Koepke. Making set theory great again: The Naproche-SAD project. 2019.

[LPV04] Alexander Lyaletski, Andrey Paskevich, and Konstantin Verchinine. Theorem proving and proof verification in the system SAD. In *International Conference on Mathematical Knowledge Management*, pages 236–250. Springer, 2004.

[Pas07] Andrei Paskevich. The syntax and semantics of the ForTheL language, 2007.