ForTheL as a Controlled Natural Language for Lean

BY PETER KOEPKE

University of Bonn, Germany

Lean Together 2020

Pittsburgh, 10 January 2020

Ordinary mathematical texts, e.g., from W. Rudin, *Principles of Mathematical Analysis*

Theorem 1. *If $x \in \mathbb{R}$, $y \in \mathbb{R}$, and $x < y$, then there exists a $p \in \mathbb{Q}$ such that $x < p < y$.*

Proof. Since $x < y$, we have $y - x > 0$, and (a) furnishes a positive integer n such that

$$m\,(y - x) > 1.$$

Apply (a) again, to obtain positive integers m_1 and m_2 such that $m_1 > n\,x$, $m_2 > -n\,x$. Then

$$-m_2 < n\,x < m_1.$$

Hence there is an integer m (with $-m_2 \leq m \leq m_1$) such that

$$m - 1 \leq n\,x < m.$$

If we combine these inequalities, we obtain

$$n\,x < m \leq 1 + n\,x < n\,y.$$

Since $n > 0$, it follows that

$$x < \frac{m}{n} < y.$$

This proves (b), with $p = m/n$. □

Ordinary mathematical texts

- Natural language with symbolic elements
- intuitive descriptions of mathematical situations, analogous to real-world situations
- exploration of situations by argumentation, with argumentative sentences and phrases

- Striving for unambiguity and exactness
- emphasizing mathematically relevant facts
- omitting routine technical details

Natural mathematical texts

- Natural language with symbolic elements
- intuitive descriptions of mathematical situations, analogous to real-world situations
- exploration of situations by argumentation, with argumentative sentences and phrases

- Striving for unambiguity and exactness
- emphasizing mathematically relevant facts
- omitting routine technical details

Approximating natural mathematical texts by ForTheL

- ForTheL = Formula Theory Language ($\sim$ sublanguage of natural mathematical language)
- 1970 *Evidence Algorithm* (Victor Glushkov)
- 1980 *System for Automated Deduction* (SAD)
- 2008 Andrei Paskevich, Haskell implementation of SAD (PhD project)
- 2017 Adoption of SAD by the Bonn Natural Proof Checking project: Naproche-SAD

- ForTheL: controlled natural language (CNL) defined by a formal grammar and implementation
- Naproche-SAD: proof-checking ForTheL texts by translations to First-Order logic and a strong ATP (eprover)

Theorem 2. *If $x \in \mathbb{R}$, $y \in \mathbb{R}$, and $x < y$, then there exists a $p \in \mathbb{Q}$ such that $x < p < y$.*

Proof. Since $x < y$, we have $y - x > 0$, and (a) furnishes a positive integer n such that

$$m\,(y - x) > 1.$$

Apply (a) again, to obtain positive integers m_1 and m_2 such that $m_1 > n\,x$, $m_2 > -n\,x$. Then

$$-m_2 < n\,x < m_1.$$

Hence there is an integer m (with $-m_2 \le m \le m_1$) such that

$$m - 1 \le n\,x < m.$$

If we combine these inequalities, we obtain

$$n\,x < m \le 1 + n\,x < n\,y.$$

Since $n > 0$, it follows that

$$x < \frac{m}{n} < y.$$

This proves (b), with $p = m/n$. □

Theorem 3. (120b) *If $x \in \mathbb{R}$ and $y \in \mathbb{R}$ and $x < y$ then there exists a rational number p such that $x < p < y$.*

Proof. Assume $x < y$. We have $y - x > 0$. Take a positive integer n such that $n \cdot (y - x) > 1$ (by 120a). Take an integer m such that $m - 1 \le n \cdot x < m$. Then

$$n \cdot x < m = (m - 1) + 1$$

$$\le (n \cdot x) + 1 < (n \cdot x) + (n \cdot (y - x))$$

$$= n \cdot (x + (y - x)) = n \cdot y.$$

$m \le (n \cdot x) + 1 < n \cdot y$. $\frac{m}{n} < \frac{n \cdot y}{n}$. Indeed $m < n \cdot y$ and $1/n > 0$. Then

$$x = \frac{n \cdot x}{n} < \frac{m}{n} < \frac{n \cdot y}{n} = y.$$

Let $p = \frac{m}{n}$. Then $p \in \mathbb{Q}$ and $x < p < y$. □

Theorem 4. *If $x\in\mathbb{R}$, $y\in\mathbb{R}$, and $x<y$, then there exists a $p\in\mathbb{Q}$ such that $x<p<y$.*

Proof. Since $x<y$, we have $y-x>0$, and (a) furnishes a positive integer n such that

$$m\,(y-x)>1.$$

Apply (a) again, to obtain positive integers m_1 and m_2 such that $m_1>n\,x$, $m_2>-n\,x$. Then

$$-m_2<n\,x<m_1.$$

Hence there is an integer m (with $-m_2\leq m\leq m_1$) such that

$$m-1\leq n\,x<m.$$

If we combine these inequalities, we obtain

$$n\,x<m\leq 1+n\,x<n\,y.$$

Since $n>0$, it follows that

$$x<\frac{m}{n}<y.$$

This proves (b), with $p=m/n$. □

Theorem 5. (120b) *If $x\in\mathbb{R}$ and $y\in\mathbb{R}$ and $x<y$ then there exists a rational number p such that $x<p<y$.*

Proof. Assume $x<y$. We have $y-x>0$. Take a positive integer n such that $n\cdot(y-x)>1$ (by 120a). Take an integer m such that $m-1\leq n\cdot x<m$. Then

$$n\cdot x<m=(m-1)+1$$

$$\leq(n\cdot x)+1<(n\cdot x)+(n\cdot(y-x))$$

$$=n\cdot(x+(y-x))=n\cdot y.$$

$m\leq(n\cdot x)+1<n\cdot y$. $\frac{m}{n}<\frac{n\cdot y}{n}$. Indeed $m<n\cdot y$ and $1/n>0$. Then

$$x=\frac{n\cdot x}{n}<\frac{m}{n}<\frac{n\cdot y}{n}=y.$$

Let $p=\frac{m}{n}$. Then $p\in\mathbb{Q}$ and $x<p<y$. □

Density of the rationals in Lean-mathlib

algebra/archimedean.lean:

```
theorem exists_rat_btwn {x y : α} (h : x < y) : ∃ q : <bbb-Q>, x < q ∧
(q:α) < y :=
begin
  cases exists_nat_gt (y - x)⁻¹ with n nh,
  cases exists_floor (x * n) with z zh,
  refine ⟨(z + 1 : <bbb-Z>) / n, _⟩,
  have n0 := nat.cast_pos.1 (lt_trans (inv_pos (sub_pos.2 h)) nh),
  have n0' := (@nat.cast_pos α _ _).2 n0,
  rw [rat.cast_div_of_ne_zero, rat.cast_coe_nat, rat.cast_coe_int,
div_lt_iff n0'],
  refine ⟨(lt_div_iff n0').2 $
    (lt_iff_lt_of_le_iff_le (zh _)).1 (lt_add_one _), _⟩,
  rw [int.cast_add, int.cast_one],
  refine lt_of_le_of_lt (add_le_add_right ((zh _).1 (le_refl _)) _) _,
  rwa [← lt_sub_iff_add_lt', ← sub_mul,
       ← div_lt_iff' (sub_pos.2 h), one_div_eq_inv],
  { rw [rat.coe_int_denom, nat.cast_one], exact one_ne_zero },
  { intro H, rw [rat.coe_nat_num, ← coe_coe, nat.cast_eq_zero] at H,
subst H, cases n0 },
  { rw [rat.coe_nat_denom, nat.cast_one], exact one_ne_zero }
end
```

Can (a variant of) ForTheL be used as an input language for Lean?

- … for natural readability of Lean formalizations?
- … for a wider acceptance in the mathematical community?
- ForTheL is an input language for FOL, with natural language types represented by unary predicates and type guards
- Propositions in dependent type theory are close to FOL
- ForTheL statements can be translated to Lean propositions (see below)
- ForTheL proofs correspond to natural deduction and FOL calculi
- Lean proofs are type theoretic terms that type-check

Natural mathematical language is weakly typed

- Aarne Ranta
- Mohan Ganesalingam
- The Naproche project (Natural Proof Checking)
- Marcos Cramer

Example

THEOREM 1.1 (The Kepler conjecture). No packing of congruent balls in Euclidean three space has density greater than that of the face-centered cubic packing.

Example

Theorem 1.1 (The Kepler conjecture). No packing of congruent balls in Euclidean three space has density greater than that of the face-centered cubic packing.

```
[synonym number/-s]
Signature. A real number is a notion.
Let x,y stand for real numbers.
Signature. x is greater than y is an atom.
Signature. A packing of congruent balls
in Euclidean three space is a notion.
Signature. The face centered cubic packing is a packing
of congruent balls in Euclidean three space.
Let P denote a packing of congruent balls in
Euclidean three space.
Signature. The density of P is a real number.

Theorem The_Kepler_conjecture. No packing of congruent
balls in Euclidean three space has density greater than
the density of the face centered cubic packing.
```

Example

```
Signature. A packing of congruent balls
in Euclidean three space is a notion.
forall v0 ((HeadTerm ::
aPackingOfCongruentBallsInEuclideanThreeSpace(v0))
implies truth)

Signature. The face centered cubic packing is a packing
of congruent balls in Euclidean three space.
forall v0 ((HeadTerm :: v0 = theFaceCenteredCubicPacking) implies
aPackingOfCongruentBallsInEuclideanThreeSpace(v0))

Theorem The_Kepler_conjecture. No packing of congruent
balls in Euclidean three space has density greater than
the density of the face centered cubic packing.
forall v0 (aPackingOfCongruentBallsInEuclideanThreeSpace(v0)
implies not isGreaterThan(theDensityOf(v0),
theDensityOf(theFaceCenteredCubicPacking)))
```

Parsing of ForTheL texts

Signature and definition commands identify defining patterns of notions for further parsing and reasoning.

"`Signature. A packing of congruent balls in Euclidean three space is a notion.`" registers the following:

- a new linguistic pattern `[Wd [‘‘packing’’], Wd [‘‘of’’], Wd [‘‘congruent’’], Wd [‘‘balls’’], Wd [‘‘in’’], Wd [‘‘euclidean’’], Wd [‘‘three’’], Wd [‘‘space’’]]`
- a new unary relation symbol `aPackingOfCongruentBallsInEuclideanThreeSpace`
- a new introductory axiom
  `forall v0 ((HeadTerm :: aPackingOfCongruentBallsInEuclideanThreeSpace(v0`
  `implies truth)`

Notion parsing

- parsing in Naproche-SAD uses a hierarchy of (sub-)parsers which are combined by parser combinators
- new notions are introduced by a host of parsers, including

```
sigNotion = do
  ((n,h),u) <- wellFormedCheck (ntnVars . fst) sig
  uDecl <- makeDecl u
  return $ dAll uDecl $ Imp (Tag HeadTerm n) h
 ...
```

- `sigNotion` produces the introductory axiom

```
forall v0 ((HeadTerm :: aPackingOfCongruentBallsInEuclideanThreeSpace(v0
implies truth)
```

- `sigNotion` also produces

```
forall v0 ((HeadTerm :: v0 = theFaceCenteredCubicPacking) implies
aPackingOfCongruentBallsInEuclideanThreeSpace(v0))
```

Notion parsing

Quantified notions are parsed by parsers like

```
quNotion = label "quantified notion" $
  paren (fa <|> ex <|> no)
  where
    fa = do ...
    ex = do ...
    no = do
      wdToken "no"; (q, f, v) <- notion
      vDecl<- mapM makeDecl v
      return (q . flip (foldr dAll) vDecl . blImp f . Not, map pVar v)
```

and lead to first-order fragments like

```
forall v0 (aPackingOfCongruentBallsInEuclideanThreeSpace(v0) implies not
```

Correspondences between languages

Semantics	classes of objects	mathematical objects	
Natural language	common noun	proper noun	quantifier
Example	real number	3	for all real numbers ...
Type theory	type	element	
	$\mathbb{R}$	$3\colon \mathbb{R}$	$\forall v\colon \mathbb{R}, \ldots$
Lean	constant	constant	
	$\mathbb{R}$: Type	$3\colon \mathbb{R}$	$\forall v\colon \mathbb{R}, \ldots$
First-order logic	unary predicate	constant	
	$\mathbb{R}(.)$	$\mathbb{R}(3)$	$\forall v\, (\mathbb{R}(v) \rightarrow \cdots)$
ForTheL	notion	term	
	real number	3 is a real number	for all real numbers ...
	aRealNumber	$\forall v (v = 3 \rightarrow \mathrm{aRealNumber}(v))$	...
Set/class theory	set/class	element	bounded quantification
	$\mathbb{R}$	$3 \in \mathbb{R}$	$\forall v \in \mathbb{R}, \ldots$

For everyday mathematics, these differences often are only notational. Here is an experiment with a simple text from mathlib

ForTheL texts and Lean texts

Signature. A *type* is a set. Let α stand for a type. Let $a\colon t$ stand for a is an element of t.

...

class monoid (α : Type u) extends semigroup α, has_one α := (one_mul : ∀ a : α, 1 * a = a) (mul_one : ∀ a : α, a * 1 = a)

Definition 6. *A* monoid *is a semigroup α such that α is a type with one and $\forall a\colon \alpha\ 1^{\alpha} *^{\alpha} a = a$ and $\forall a\colon \alpha\ a *^{\alpha} 1^{\alpha} = a$.*

class comm_monoid (α : Type u) extends monoid α, comm_semigroup α

Definition 7. *A* commutative monoid *is a monoid that is a commutative semigroup.*

class group (α : Type u) extends monoid α, has_inv α := (mul_left_inv : ∀ a : α, a^{-1} * a = 1)

Definition 8. *A* group *is a monoid α such that α is a type with inverses and for all $a\colon \alpha\ a^{-1,\alpha} *^{\alpha} a = 1^{\alpha}$.*

class comm_group (α : Type u) extends group α, comm_monoid α

Definition 9. *A* commutative group *is a group that is a commutative monoid.*

lemma mul_assoc [semigroup α] : ∀ a b c : α, a * b * c = a * (b * c) := semigroup.mul_assoc

instance semigroup_to_is_associative [semigroup α] : is_associative α (*) := ⟨mul_assoc⟩

Lemma 10. [mul assoc] *Let α be a semigroup. Then for all $a, b, c\colon \alpha\ a *^{\alpha} (b *^{\alpha} c) = a *^{\alpha} (b *^{\alpha} c)$.*

lemma mul_comm [comm_semigroup α] : ∀ a b : α, a * b = b * a := comm_semigroup.mul_comm

Lemma 11. [mul comm] *Let α be a commutative semigroup. Then for all $a, b\colon \alpha\ a *^{\alpha} b = b *^{\alpha} a$.*

Translating from ForTheL to Lean

Modifying the FOL output of Naproche-SAD we could produce valid Lean code (FOL typeguards $\mapsto$ type restrictions) (with Adrian De Lon and Daniel Kollert)

Signature. A prime number is a natural number.

Let p denote a prime number.

Axiom PrimeIrred. If p | (b * c) then p | b or p | c.

Axiom ClearDenom. There exist coprime b,c such that b * q = c.

Proposition PrimeNoSquare. q * q = p for no rational number q.

Proof. Assume the contrary. Take a rational number q such that p = q * q. Take coprime a,b such that a * q = b. Then p * (a * a) = b * b. Therefore p divides b. Take a natural number c such that b = c * p.

Then p * (a * a) = p * (c * b).

Therefore a * a is equal to p * (c * c). Hence p divides a. Contradiction.
qed.

axiom isPrimeNumber : NaturalNumber → Prop

notation `PrimeNumber` := {x : NaturalNumber // isPrimeNumber x}

axiom PrimeIrred : ∀ p : PrimeNumber, ∀ b c : NaturalNumber, ((Div p) (m b c)) → Div p b ∨ Div p c

axiom ClearDenom : ∀ q : RationalNumber, (∃ v0 v1 : NaturalNumber, Coprime v0 v1 ∧ (eq (m v0 q)) (v1))

theorem PrimeNoSquare : ∀ p : PrimeNumber, ∀ v0 : RationalNumber, not ((eq (m v0 v0)) (p)) := omitted

Parsing ForTheL to type theory

The command

`Signature. A packing of congruent balls in Euclidean three space is a notion.`

should register

- a new linguistic pattern `[Wd [“packing”], Wd [“of”], Wd [“congruent”], Wd [“balls”], Wd [“in”], Wd [“euclidean”], Wd [“three”], Wd [“space”]]`
- a new type constant `aPackingOfCongruentBallsInEuclideanThreeSpace`
- a new introductory axiom `aPackingOfCongruentBallsInEuclideanThreeSpace(v0)) : Type`

```
sigNotion = do
  ((n,h),u) <- wellFormedCheck (ntnVars . fst) sig
  uDecl <- makeDecl u
  return $ dAll uDecl $ Imp (Tag HeadTerm n) h
  -- type theoretic return like (n `member` "Type")
...
```

Problems and perspectives

- identify function terms and function types in natural language mathematics
- declarative proofs (ForTheL) versus procedural proofs (Lean)
- concentrate on declarative statements (definitions, statements of theorems)
- this corresponds to the fabstracts approach
- CNLtoLean: divide Naproche-SAD parsing into 1. linguistic parsing + 2. translation into formal logic (FOL + type theory)
- investigate declarative proofs for Lean, corresponding to Isar proofs for Isabelle
- natural language statements motivate to work towards naturally structured proofs in type theory, similar to the natural proofs in Naproche-SAD
- natural proofs require strong automatic proving for implicit simple proof obligations

General aspects

- Mathematics uses natural language with specific mathematical phrases
- The language of mathematics can be modeled by different logics
- The language of mathematics serves the communication between mathematicians and thus possesses a high degree of universality, independent of logical modeling
- Development of controlled general mathematical languages that can be projected out to several logics

General aspects

- Mathematics uses natural language with specific mathematical phrases
- The language of mathematics can be modeled by different logics
- The language of mathematics serves the communication between mathematicians and thus possesses a high degree of universality, independent of logical modeling
- Development of controlled general mathematical languages that can be projected out to several logics
- Richard Montague, 1970: *I reject the contention that an important theoretical difference exists between formal and natural languages.*
- Provocation 1: *I reject the contention that an important theoretical difference exists between formal and natural mathematical languages.*
- Provocation 2: *I reject the contention that an important theoretical difference exists between formal and natural mathematics.*

Thank you for your attention!