Toward an Object-Oriented Structure for Mathematical Text

Fairouz Kamareddine, Manuel Maarek and Joe Wells

ULTRA Group – Heriot-Watt University

November 14th, 2005
Theorema-ULTRA-Omega'05 Workshop
Saarland University, Germany

Computerising mathematical texts

CML

The Common Mathematical Language used by mathematicians in their everyday writings is known as *meticulous*, in comparison with other natural languages, *flexible*, in its way to accommodate many branches of mathematics, *coherent* by providing contextual justifications of statements.

Is CML reflected in current approaches
of computerising Mathematics?

Two examples

From Euclid to Bourbaki

Definition 20. *Of trilateral figures, an equilateral triangle is that which has its three sides equal, an isosceles triangle that which has two of its sides alone equal, and a scalene triangle that which has its three sides unequal.*

Euclid [The 13 Books of Euclid's Elements, Book I]

Definition 1. *A set with an associative law of composition, possessing an identity element and under which every elements is invertible, is called a group. [...] A group G is called finite if the underlying set of G is finite [...] A group [with operators] G is called commutative (or Abelian) if its group law is commutative.*

N. Bourbaki [Elements of Mathematics - Algebra, volume II, Chapter I, §4]

Two examples

From Euclid to Bourbaki

Definition 20. *Of trilateral figures, an equilateral triangle is that which has its three sides equal, an isosceles triangle that which has two of its sides alone equal, and a scalene triangle that which has its three sides unequal.*

Euclid [The 13 Books of Euclid's Elements, Book I]

Definition 1. *A set with an associative law of composition, possessing an identity element and under which every elements is invertible, is called a group. [...] A group G is called finite if the underlying set of G is finite [...] A group [with operators] G is called commutative (or Abelian) if its group law is commutative.*

N. Bourbaki [Elements of Mathematics - Algebra, volume II, Chapter I, §4]

Mathematical word processing

LaTeX

LaTeX

```
\begin{definition}
  Of trilateral figures, an equilateral triangle is that which has its
  three sides equal, an isosceles triangle that which has two of its
  sides alone equal, and a scalene triangle that which has its three
  sides unequal.
\end{definition}
```

```
\begin{definition}
  A set with an associative law of composition, possessing an identity
  element and under which every elements is invertible, is called a
  group.  [...]  A group $G$ is called finite if the underlying set of
  $G$ is finite [...]  A group [with operators] $G$ is called
  commutative (or Abelian) if its group law is commutative.
\end{definition}
```

- Visual representation of CML
- Difficult semantic recognition

Semantic markup languages

MathML, OpenMath, OMDoc

OpenMath/OMDoc

```
<!-- Euclid's example -->
<theory name="Euclid-book-1">
  <symbol id="equilateral-triangle">
    <CMP>An equilateral triangle is [...]

<!-- Bourbaki's example -->
<theory name="Group">
  <symbol id="*">
  <symbol id="E">
    <CMP>A set with  <OMOBJ>*</OMOBJ>, associative
         law of composition.
    <FMP>(a * b) * c = a * (b * c)
  <symbol id="e"> [...]
<theory name="FiniteGroup">
  <imports from="Group"> [...]
```

- Flexible
- Difficult semantic recognition due to the mixture of structural and symbolic XML and chunks of natural language.
- Extensible with embedded formal content

Full formalisation

Theorem provers

Our goal differs from full formalisation.

We want to provide a control over presentation and phrasing of the semantic structure. Most mathematical texts are unlikely to be formalized, but might well benefit from computerisation.

Procedural style – such as Coq, Isabelle

- Fully formalised
- Requires expertise
- Formalisation that may not reflect the CML text

Declarative style – such as Mizar

- Fully formalised
- Requires expertise and the Mizar Mathematical Library
- Syntax mimics natural language
- Formal Proof Sketch (a lighter version of Mizar)

Computerising the mathematical vernacular

N.G. de Bruijn's MV – WTT – MathLang-WTT – MathLang

N.G. de Bruijn's Mathematical Vernacular A language with *substantives*, *adjectives* and *flags*

The Weak Type Theory A *type system* for MV with weak types (TERM, NOUN, ADJ, SET, STAT, LINE and BOOK)

MathLang-WTT A practical evaluation of MV and WTT

- Extends WTT with FLAGS and BLOCKS
- Automates type checking
- Has been used to translate existing CML texts
- Proposes various output-views faithful to CML

MathLang's approach to computerise mathematical texts is to:

- Capture, in a first ground, the grammatical structure of the text
- Enhance this first ground language with a choice of features (semantical, logical, ...)

Computerising the mathematical vernacular

MathLang-WTT – output-view (Example from F. Wiedijk's comparison)

T Terms S Sets N Nouns A Adjectives P Statements Z Declarations Γ Contexts L Lines F Flags K Blocks B Books

$Th() := irrational(\sqrt{2})$ 1

$\{1\}$

$rational(\sqrt{2})$, $a : integer$, $b : integer$, $(a, b) = 1$

$soluble(a^2 = 2 * b^2)$. 2

$even(a^2)$. 3 $even(a)$. 4

$c : integer$, $a = 2 * c$

$4 * c^2 = 2 * b^2$. 5 $2 * c^2 = b^2$. 6

$even(b)$. 7 $(a, b) = 2$. 8

$contradiction((a, b) = 1$, Line 8$)$ 9

Th 10

- Symbolic view
- CML view of symbols
- CML view of the document

Computerising the mathematical vernacular

MathLang-WTT – output-view (Example from F. Wiedijk's comparison)

T Terms S Sets N Nouns A Adjectives P Statements Z Declarations Γ Contexts L Lines F Flags K Blocks B Books

Th() := $\sqrt{2}$ is irrational 1

{1}

$\sqrt{2}$ is rational , *a* : *integer* , *b* : *integer* , (*a* , *b*) = *1*

the equation a ^{2} = 2 b ^{2} is soluble . 2

a ^{2} is even . 3

a is even . 4

c : *integer* , a = 2 c

4 c ^{2} = 2 b ^{2} . 5

2 c ^{2} = b ^{2} . 6

b is also even . 7

(*a* , *b*) = 2 . 8

contrary to the hypothesis that (a , b) = *1* 9

Th 10

- Symbolic view
- CML view of symbols
- CML view of the document

Computerising the mathematical vernacular

MathLang-WTT – output-view (Example from F. Wiedijk's comparison)

T Terms S Sets N Nouns A Adjectives P Statements Z Declarations Γ Contexts L Lines F Flags K Blocks B Books

Theorem (Pythagoras' Theorem). $\sqrt{2}$ *is irrational*. 1

Proof.

If $\sqrt{2}$ is rational, then the equation $a^2 = 2\ b^2$ is soluble 2 in integers a, b with $(a, b) = 1$.

Hence a^2 is even, 3 and therefore a is even. 4

If $a = 2\ c$, then

$4\ c^2 = 2\ b^2$, 5 $2\ c^2 = b^2$, 6

and b is also even, 7 . 8

contrary to the hypothesis that $(a, b) = 1$. 9

10 □

- Symbolic view
- CML view of symbols
- CML view of the document

MathLang-WTT

Encodings of Euclid's and Bourbaki's examples?

How to faithfully encode *a triangle* and *its sides*, *a group* and *its law* in MathLang-WTT?

MathLang-WTT

Encodings of Euclid's and Bourbaki's examples?

How to faithfully encode *a triangle* and *its sides*, *a group* and *its law* in MathLang-WTT?

- `triangle` and `side`, `group` and `law` as constants of type NOUN.

MathLang-WTT

Encodings of Euclid's and Bourbaki's examples?

How to faithfully encode *a triangle* and *its sides*, *a group* and *its law* in MathLang-WTT?

- `triangle` and `side`, `group` and `law` as constants of type NOUN.

How to encode the intrinsic relation between a triangle and its lines and between a group and its law?

MathLang-WTT

Encodings of Euclid's and Bourbaki's examples?

How to faithfully encode *a triangle* and *its sides*, *a group* and *its law* in MathLang-WTT?

- `triangle` and `side`, `group` and `law` as constants of type NOUN.

How to encode the intrinsic relation between a triangle and its lines and between a group and its law?

- By parametrising `triangle` and `group` with `sides` and `law`
  → **Constraining & not flexible**
- By using a statement "`has`".

```
has(triangle,line1); has(triangle,line2); has(triangle,line3)
has(group,law)
```

→ **Verbose & not reliable**

Obviously, this kind of fundamental description of mathematical objects was not satisfactory and needed improvement

Abstraction with nouns and adjectives

- **Back to N.G. de Bruijn's informal definitions.**

MV's substantives (MathLang-WTT's nouns)

MV's adjectives (MathLang-WTT's adjectives)

MV's names (MathLang-WTT's terms)

Abstraction with nouns and adjectives

- Back to N.G. de Bruijn's informal definitions.
- **Analogy with Object-oriented programming.**

MV's substantives (MathLang-WTT's nouns)
Classes

MV's adjectives (MathLang-WTT's adjectives)
Mixins (functions from classes to classes)

MV's names (MathLang-WTT's terms)
Objects

Abstraction with nouns and adjectives

- Back to N.G. de Bruijn's informal definitions.
- Analogy with Object-oriented programming.
- **New design of MathLang with object-oriented aspects.**

MV's substantives (MathLang-WTT's nouns)

Classes

Nouns as classes

MV's adjectives (MathLang-WTT's adjectives)

Mixins (functions from classes to classes)

Adjectives as mixins

MV's names (MathLang-WTT's terms)

Objects

Terms as objects

Abstraction with nouns and adjectives

Euclid's example

Definition 20. *Of trilateral figures, an equilateral triangle is that which has its three sides equal, an isosceles triangle that which has two of its sides alone equal, and a scalene triangle that which has its three sides unequal.* *Euclid [The 13 Books of Euclid's Elements, Book I]*

Figure and triangle defined as nouns. Trilateral and equilateral defined as adjectives.

```
{   figure := Noun { sides:= set(line);
                     contained_by(self,self.sides) };
    trilateral := Adj (figure) { card(self.sides) = 3 };
    triangle := trilateral figure;
    equilateral := Adj (triangle) {
                     forall (side1:self.sides,
                       forall (side2:self.sides,
                         side1.length = side2.length)) } }
```

Abstraction with nouns and adjectives

Euclid's example

Definition 20. *Of trilateral figures, an equilateral triangle is that which has its three sides equal, an isosceles triangle that which has two of its sides alone equal, and a scalene triangle that which has its three sides unequal.* *Euclid [The 13 Books of Euclid's Elements, Book I]*

Figure and triangle defined as nouns. Trilateral and equilateral defined as adjectives.

```
{   figure := Noun { sides:= set(line);
                     contained_by(self,self.sides) };
    trilateral := Adj (figure) { card(self.sides) = 3 };
    triangle := trilateral figure;
    equilateral := Adj (triangle) {
                     forall (side1:self.sides,
                       forall (side2:self.sides,
                         side1.length = side2.length)) } }
```

Abstraction with nouns and adjectives

Bourbaki's example

Definition 1. *A set with an associative law of composition, possessing an identity element and under which every elements is invertible, is called a group. [...] A group G is called finite if the underlying set of G is finite [...] A group [with operators] G is called commutative (or Abelian) if its group law is commutative.* N. Bourbaki *[Elements of Mathematics - Algebra, volume II, Chapter I, §4]*

Group defined as a noun. Finite and Abelian defined as adjectives.

```
{   group := Noun { E:set;
                    { a:E; b:E } |> * (a,b) :E;
                    e:E;
                    forall (a:E, forall (b:E, forall(c:E,
                      *(*(a,b),c) = *(a,*(b,c)) )));
                    forall (x:E, invertible(e,x)) };
    finite := Adj (group) { finit_set(self.E) };
    Abelian := Adj (group) {
      forall (x:self.E, forall (y:self.E,
        self.* (x,y) = self.* (y,x) )) } }
```

Abstraction with nouns and adjectives

Bourbaki's example

Definition 1. *A set with an associative law of composition, possessing an identity element and under which every elements is invertible, is called a group. [...] A group G is called finite if the underlying set of G is finite [...] A group [with operators] G is called commutative (or Abelian) if its group law is commutative. N. Bourbaki [Elements of Mathematics - Algebra, volume II, Chapter I, §4]*

Group defined as a noun. Finite and Abelian defined as adjectives.

```
{   group := Noun { E:set;
                    { a:E; b:E } |> * (a,b) :E;
                    e:E;
                    forall (a:E, forall (b:E, forall(c:E,
                      *(*(a,b),c) = *(a,*(b,c)) )));
                    forall (x:E, invertible(e,x)) };
    finite := Adj (group) { finit_set(self.E) };
    Abelian := Adj (group) {
      forall (x:self.E, forall (y:self.E,
        self.* (x,y) = self.* (y,x) )) } }
```

Abstraction with nouns and adjectives

Multi adjective refinements

$$
\begin{tikzcd}
\texttt{group} \arrow[r, "\texttt{finite}"] \arrow[d, "\texttt{Abelian}"'] & \texttt{finite group} \arrow[d, "\texttt{Abelian}"] \\
\texttt{Abelian group} \arrow[r, "\texttt{finite}"'] & \begin{array}{l}\texttt{Abelian finite group} \\ \texttt{finite Abelian group}\end{array}
\end{tikzcd}
$$

- Combine the adjectives *finite* and *Abelian* to obtain either `Abelian finite group` or `finite Abelian group`.
- In MathLang both expressions share the same type. Their meaning may differ as the statements introduced by the adjectives may overlap.
- It is possible to define an `isosceles equilateral scalene triangle`.
- But not a `Abelian triangle` (with these current definitions).

Syntax

Sets, category expressions and identifiers

$$
\begin{array}{lcl}
\mathit{ident},\ i & = & \text{denumerably infinite set of identifiers} \\
\mathit{label},\ l & = & \text{denumerably infinite set of labels} \\
\mathit{cvar},\ v & = & \text{denumerably infinite set of category variables} \\
 & & \\
\mathit{category},\ c & ::= & \mathbf{term}(\mathit{exp}) \mid \mathbf{set}(\mathit{exp}) \mid \mathbf{noun}(\mathit{exp}) \mid \mathbf{adj}(\mathit{exp}, \mathit{exp}) \\
 & \mid & \mathbf{stat} \mid \mathbf{dec}(\mathit{category}) \mid \mathit{cvar} \\
\mathit{cident},\ \mathit{ci} & ::= & \mathit{ident} \mid \mathit{exp}.\mathit{cident}
\end{array}
$$

Syntax

Steps

step, *s*	::=	*phrase*	Basic unit
	|	**label** *label step*	Labelling
	|	*step* $\triangleright$ *step*	Local scoping
	|	$\{\overrightarrow{step}\}$	Block

(an arrow on top of a meta-variable represents a sequence of 0 or more meta-variables)

```
Block: sequence of reasoning statements
{ x*(y+1) = x*y';
  x*y' = x*y+x;
  x*y+x = x*y+x*1 }

Blocks and sub-blocks
{ --A proof of P by induction--
  { --Proof of the base-- [...]; P(0) };
  { --Proof of the induction-- { n:N; P(n) } |> { [...]; P(n+1) } } }

Local scoping: contextualises one reasoning step
{ --Proof of the contradiction-- [...] }
      |> { --Statement proved by contradiction-- [...] }
```

Syntax

Phrases and expressions

phrase, *p*	$::=$	*exp*	
	$\mid$	$\mathit{cident}(\overrightarrow{\mathit{ident}}) := \mathit{exp}$	Definition
	$\mid$	$\mathit{ident}(\overrightarrow{\mathit{exp}}) := \mathit{exp}$	Definition by matching case
	$\mid$	$\mathit{ident} \ll \mathit{cident}$	Sub-noun and adjective statement
exp, *e*	$::=$	$\mathit{cident}(\overrightarrow{\mathit{exp}})$	Instance
	$\mid$	$\mathit{ident}(\overrightarrow{\mathit{category}}) : \mathit{exp}$	Elementhood declaration
	$\mid$	$\mathit{ident}(\overrightarrow{\mathit{category}}) : \mathit{category}$	Declaration
	$\mid$	**Noun** $\{\mathit{step}\}$	Noun
	$\mid$	**Adj**$(\mathit{exp})\ \{\mathit{step}\}$	Adjective
	$\mid$	*exp exp*	Refinement
	$\mid$	**self** $\mid$ **super**	Self and super
	$\mid$	**ref** *label*	Referencing

Type system

Rules for steps

$$\frac{\vdash s_1 : Step \qquad s_1 \vdash s_2 : Step \qquad \{s_1; s_2\} \vdash \{\overrightarrow{s}\} : Step}{s_1 \vdash \{s_2; \overrightarrow{s}\} : Step}\ \text{STEP-COMPOSITION}$$

$$\frac{\vdash s : Step \qquad s \vdash s' : Step \qquad \{s; s'\} \vdash s'' : Step}{s \vdash s' \rhd s'' : Step}\ \text{LOCAL-SCOPING}$$

$$\frac{\vdash s : Step \qquad s \vdash p : Stat/Dec(t)/Def(t)}{s \vdash p : Step}\ \text{ATOMIC-STEP}$$

$$\frac{}{\vdash \{\} : Step}\ \text{EMPTY-STEP}$$

Type system

Rules for noun and adjective expressions

$$\frac{\vdash s : Step \qquad \{s;\ self : Term(T)\} \vdash s' : Step \quad \forall i \in I(s'), \{s;\ self : Term(T);\ s'\} \vdash i : T(i)}{s \vdash \textbf{Noun}\ \{s'\} : Noun(T)}\ \text{NOUN}$$

$$\frac{\vdash s : Step \qquad s \vdash e : Noun(T) \quad T \leq T' \qquad \{s; super : Term(T); self : Term(T')\} \vdash s' : Step \quad \forall i \in I(s'), \{s; super : Term(T); self : Term(T'); s'\} \vdash i : T'(i)}{s \vdash \textbf{Adj}\ (e)\ \{s'\} : Adj(T, T')}\ \text{ADJ}$$

$$\frac{\vdash s : Step \qquad s \vdash e_1 : Adj(T_1, T_1') \quad s \vdash e_2 : Noun(T_2)/Set(T_2)/Term(T_2) \qquad T_1 \leq T_2}{s \vdash e_1 e_2 : Noun(T_1' \uplus T_2)/Set(T_1' \uplus T_2)/Term(T_1' \uplus T_2)}\ \text{REFINEMENT}$$

Type system

Example of typing – Euclid's example

Term Terms *Set* Sets *Noun* Nouns *Adj* Adjectives *Stat* Statements

Def Definition *Step* Local scopings ▷ *Step* Blocks { }

Definition 20. Of trilateral figures,

an equilateral triangle is that which has its three sides equal,

an isosceles triangle that which has two of its sides alone equal,

and a scalene triangle that which has its three sides unequal.

Type system

Example of typing – Bourbaki's example

Term Terms *Set* Sets *Noun* Nouns *Adj* Adjectives *Stat* Statements

Def Definition *Step* Local scopings ▷ *Step* Blocks { }

Definition 1.

A set

with an associative law of composition,

possessing an identity element

and under which every elements is invertible, is called a group.

[...]

A group G is called finite if the underlying set of G is finite

[...]

A group [with operators] G is called *commutative* (or *Abelian*) if its group law is commutative.

Other works

Krzysztof Retel

In his research work, Krzysztof Retel investigates ways to

- Bridge MathLang with existing systems for formalising mathematics.
- Design MathLang features that would extend the semantical knowledge contained in MathLang documents and provide opportunities for verification.

To target these two points he experienced translations of CML documents into Mizar via MathLang. He compared this translation path with a direct translation into Mizar and proposed guidance for such gradual translations.

Future works

MathLang ongoing works

- Development of a user interface for MathLang based on the scientific editor $\text{T}_{\text{E}}\text{X}_{\text{MACS}}$.
- Design of MathLang extension features.
- Bridging existing systems and languages (Mizar, OpenMath, OMDoc) with features combinations.
- Orienting MathLang development with translation of mathematical documents.
- Refinement of the object-oriented aspect of MathLang with *traits*.

Conclusion

MathLang

- We saw how the experience-driven development of MathLang led us to extend our ground language by turning nouns into classes and adjectives into mixins.
- MathLang provides an expressive encoding for computerising the symbolic and natural language parts of mathematical text.
- Our current work is aimed to demonstrate the utility of decomposing the path towards full formalisation.