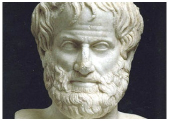
Computerisation

Fairouz Kamareddine
Heriot-Watt University
Edinburgh, UK

Data-Mining, Paris, 8 September 2017

And God created Aristotle



- Assume a problem Π,
  - If you *give* me an algorithm to solve Π, I can check whether this algorithm really solves Π.
  - But, if you ask me to *find* an algorithm to solve Π, I may go on forever trying but without success.
- But, this result was already known to Aristotle:
- Assume a proposition Φ.
  - If you *give* me a proof of Φ, I can check whether this proof really proves Φ.
  - But, if you ask me to *find* a proof of Φ, I may go on forever trying but without success.
- In fact, *programs* are *proofs*:
  - *program* = *algorithm* = *computable function* = *λ-term*.
  - By the PAT principle: *Proofs* are *λ-terms*.

Pre Computerisation

Much later than Aristotle, Leibniz (1646–1717) conceived of **automated deduction**, i.e., to find

- a language *L* in which arbitrary concepts could be formulated, and
- a method to determine the correctness of statements in *L*.

In other words, Leibniz wanted a language and a method that could carry out proof checking and proof finding. However, according to Aristotle and (later results by) Gödel and Turing, such a method can not work for every statement.

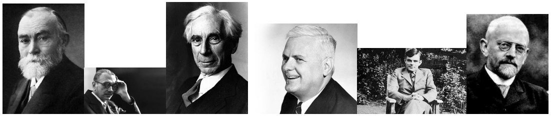
Pre Computerisation

- 1879:
  Frege was not satisfied with the use of *natural language in mathematics*:

  > *". . . I found the inadequacy of language to be an obstacle; no matter how unwieldy the expressions I was ready to accept, I was less and less able, as the relations became more and more complex, to attain the precision that my purpose required."*

  (Begriffsschrift, *Preface)*

Pre Computerisation



- *General definition of function* [8] is key to Frege's *formalisation*.
- *Self-application of functions* was at the heart of *Russell's paradox* [30].
- To *avoid paradox* Russell controled function application via *type theory* RTT.
- RTT is used in Russell and Whitehead's Principia Mathematica.
- Church's *simply typed λ-calculus* $\lambda\!\rightarrow$ [4] 1940 $=$ λ-calculus $+$ STT.
- The hierarchies of types/orders in RTT and STT are *unsatisfactory*.
- Hence, birth of *different systems of functions and types*, each with *different functional power*.

Language and Machine of the computable: By 1936

- Church: f is computable iff f can be written in λ-calculus.
  - $A ::= x \mid AB \mid \lambda x.B$
  - $(\lambda x.B)C \to_\beta B[x := C]$.
- Turing: f is computable iff f can be run on a Turing Machine.
- Size of computable functions is the size of $\mathbb{N}$.
- Size of non computable functions is the size of $\mathbb{R}$.
- $|\mathbb{N}|$ is countable (drop of water).
- $|\mathbb{R}|$ is uncountable (an ocean).
- This is impressive considering that until end of 19th century, they were still asking *What is a real number?*

The Rest of the 20th century

- Design and continue to improve the *language* and *machine* of the computable.
  - Turing machine very hard to work with and very slow.
  - λ-calculus does not have *logic* and programs written in lambda calculus would be *incomprehensible to humans.*
- We know the size of the computable, but:
  - Even if a function is computable, we may not be able to compute it. It may be so hard to compute. Work on improving *complexity.*
  - We still don't know whether some function are or are not computable. Keep working on putting these functions to the *corresponding class.*
- Computers are improving a lot faster than languages and softwares for these machines.

The Challenge

- Is to design frameworks that work well for all the features needed for expressive computation:
  - Data Types
    - Built-in: numbers, arrays, lists, etc.
    - User-defined: Records, Abstract Data Types or Symbolic Expressions (written in BNF).
  - Code: Inference rules,
  - logic and mathematics,
  - capture-free substitution within a symbolic expression,
- while having a clear syntax, semantics and the desired properties (Correctness, Termination).

Data Declaration à la Alonzo Church

Type Free

- $A ::= x \mid AB \mid \lambda x.B$
- $(\lambda x.B)C \to_\beta B[x := C]$.

With simple types:

- $\sigma ::= T \mid \sigma \to \tau$
- $A ::= x \mid AB \mid \lambda x{:}\sigma.B$
- $(\lambda x : \sigma.B)C \to_\beta B[x := C]$.

With dependent types:

- $A ::= x \mid * \mid \Box \mid AB \mid \lambda x{:}A.B \mid \Pi x{:}A.B$
- $(\lambda x : A.B)C \to_\beta B[x := C]$.

Code à la Alonzo Church

With dependent/polymorphic types

$$(\lambda) \qquad \frac{\Gamma, x{:}A \vdash b : B \qquad \Gamma \vdash \Pi_{x:A}.B : s}{\Gamma \vdash \lambda_{x:A}.b : \Pi_{x:A}.B}$$

$$(\mathrm{app}_{\Pi}) \qquad \frac{\Gamma \vdash F : \Pi_{x:A}.B \qquad \Gamma \vdash a : A}{\Gamma \vdash Fa : B[x{:=}a]}$$

With simple types:

$$(\lambda_{\rightarrow}) \qquad \frac{\Gamma, x{:}\sigma \vdash b : \tau}{\Gamma \vdash \lambda_{x:A}.b : \sigma \rightarrow \tau}$$

$$(\mathrm{app}) \qquad \frac{\Gamma \vdash F : \tau \rightarrow \sigma \qquad \Gamma \vdash a : \tau}{\Gamma \vdash Fa : \sigma}$$

Even a minor change can mean big things

For example, simply changing the order of functions and arguments, and restructuring parenthesis, enable us to:

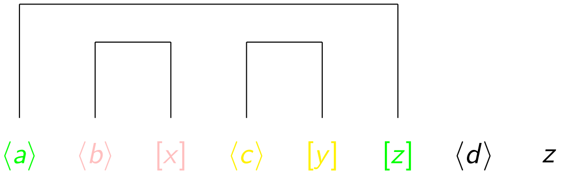
- Express things that would hard to do in the old notation.
- Reduce proofs of strong normalisation to proofs of weak normalisation.
- Make computations more efficient.
- Avoid unnecessary/redundant computations and allow for free lazy, local, or global reductions.

Lambda Calculus à la de Bruijn

- $A := x \mid AB \mid \lambda x.B$ $\qquad A := x \mid\ < B > A \mid [x]B$
- $(\lambda x.\lambda y.xy)z$ $\qquad \langle z\rangle[x][y]\langle y\rangle x.$
- $(\lambda x.A)B \to_\beta A[x := B]$ $\qquad \langle B\rangle[x]\ A \to_\beta [x := B]A$
- $((\ \lambda_x.(\lambda_y.\lambda_z.\text{–})c)ba)$ $\qquad [_1\quad [_2\quad [_3\quad]_2\quad]_1$
- $\langle a\rangle\langle b\rangle[x]\langle c\rangle[y][z]\ \langle d\rangle$ $\qquad [\ [\ ][\ ]].$

Redexes in de Bruijn's notation

Classical Notation	de Bruijn's Notation
$(\underline{(\lambda_x.(\lambda_y.\lambda_z.zd)c)b})a$	$\langle a\rangle\underline{\langle b\rangle[x]}\langle c\rangle[y][z]\langle d\rangle z$
$\downarrow_\beta$	$\downarrow_\beta$
$(\underline{(\lambda_y.\lambda_z.zd)c})a$	$\langle a\rangle\underline{\langle c\rangle[y]}[z]\langle d\rangle z$
$\downarrow_\beta$	$\downarrow_\beta$
$\underline{(\lambda_z.zd)a}$	$\underline{\langle a\rangle[z]}\langle d\rangle z$
$\downarrow_\beta$	$\downarrow_\beta$
ad	$\langle d\rangle a$



- This maks it easy to study local/global/mini reductions into the λ-calculus, Kamareddine etal [17, 18]

Some notions of reduction studied in the literature

Name	*In Classical Notation*	*In de Bruijn's notation*
(θ)	$((\lambda_x.N)P)Q$ $\downarrow$ $(\lambda_x.NQ)P$	$\langle Q\rangle\langle P\rangle[x]N$ $\downarrow$ $\langle P\rangle[x]\langle Q\rangle N$
(γ)	$(\lambda_x.\lambda_y.N)P$ $\downarrow$ $\lambda_y.(\lambda_x.N)P$	$\langle P\rangle[x][y]N$ $\downarrow$ $[y]\langle P\rangle[x]N$
(γ_C)	$((\lambda_x.\lambda_y.N)P)Q$ $\downarrow$ $(\lambda_y.(\lambda_x.N)P)Q$	$\langle Q\rangle\langle P\rangle[x][y]N$ $\downarrow$ $\langle Q\rangle[y]\langle P\rangle[x]N$
(g)	$((\lambda_x.\lambda_y.N)P)Q$ $\downarrow$ $(\lambda_x.N[y:=Q])P$	$\langle Q\rangle\langle P\rangle[x][y]N$ $\downarrow$ $\langle P\rangle[x][y:=Q]N$
(β_e)	? $\downarrow$?	$\langle Q\rangle\overline{s}[y]N$ $\downarrow$ $\overline{s}[y:=Q]N$

A Few Uses of these reductions/term reshuffling

- Regnier [27] uses θ and γ in analyzing perpetual reduction strategies.
- Term reshuffling is used by Kfoury, Tiuryn, Urzyczyn, Wells in [22, 20] in analyzing typability problems.
- Nederpelt [24], de Groote [7], Kfoury+ Wells [21], and Kamareddine [16] use generalised reduction and/or term reshuffling in relating SN to WN.
- Ariola etal [1] uses a form of term-reshuffling in obtaining a calculus that corresponds to lazy functional evaluation.
- Kamareddine etal [17, 14, 19, 3] show that they could reduce space/time needs in computation.

Even more: de Bruijn's generalised reduction has better properties

$$\begin{array}{lll}
(\beta) & (\lambda_x.M)N \to M[x := N] & \\
(\beta_I) & (\lambda_x.M)N \to M[x := N] & \text{if } x \in FV(M) \\
(\beta_K) & (\lambda_x.M)N \to M & \text{if } x \notin FV(M) \\
(\theta) & (\lambda_x.N)PQ \to (\lambda_x.NQ)P & \\
(\beta_e) & (M)\overline{s}[x]N \to \overline{s}\{N[x := M] & \text{for } \overline{s} \text{ well-balanced.}
\end{array}$$

- Kamareddine [16] shows that β_e satisfies *Church Rosser*, *PSN*, postponment of K-contraction and conservation (latter 2 properties fail for β-reduction).
- *Conservation of β_e*: If A is $\beta_e I$-normalisable then A is β_e-strongly normalisable.
- Postponment of K-contraction : Hence, discard arguments of K-redexes after I-reduction. This gives flexibility in implementation: *unnecessary work can be delayed, or even completely avoided.*

- Attempts have been made at establishing some reduction relations for which postponement of K-contractions and conservation hold.
- The picture is as follows (-N stands for normalising and $r \in \{\beta_I, \theta_K\}$).

(β_K-postponement for r)	If $M \to_{\beta_K} N \to_r O$ then $\exists P$ such that $M \twoheadrightarrow^+_{\beta_I\theta_K} P \twoheadrightarrow_{\beta_K} O$	
(Conservation for β_I)	If M is β_I-N then M is β_I-SN	Barendregt's book
(Conservation for $\beta + \theta$)	If M is $\beta_I\theta_K$-N then M is β-SN	[7]

- De Groote does not produce these results for a single reduction relation, but for $\beta + \theta$ (this is more restrictive than β_e).
- β_e is the first single relation to satisfy β_K-postponement and conservation.
- Kamareddine [16] shows that:

(β_{eK}-postponement for β_e)	If $M \to_{\beta_{eK}} N \to_{\beta_{eI}} O$ then $\exists P$ such that $M \to_{\beta_{eI}} P \twoheadrightarrow^+_{\beta_{eK}} O$
(Conservation for β_e)	If M is β_{eI}-N then M is β_e-SN

Church's Simply Typed λ-calculus in modern notation

- Terms $A ::= x \mid AB \mid \lambda x{:}\sigma.B$
- Types $::= T \mid \sigma \to \tau$
- Γ is an environment (set of declaration).
- Rules:

$$\begin{array}{ll}
\text{(start)} & \dfrac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma} \\[2ex]
(\lambda) & \dfrac{\Gamma, x{:}\sigma \vdash A : \tau}{\Gamma \vdash \lambda_{x:\sigma}.A : \sigma \to \tau} \\[2ex]
(\text{app}_\Pi) & \dfrac{\Gamma \vdash A : \sigma \to \tau \qquad \Gamma \vdash B : \sigma}{\Gamma \vdash AB : \tau}
\end{array}$$

The Barendregt Cube

- Syntax: $A ::= x \mid * \mid \Box \mid AB \mid \lambda x{:}A.B \mid \Pi x{:}A.B$
- Formation rule:

$$\frac{\Gamma \vdash A : s_1 \qquad \Gamma, x{:}A \vdash B : s_2}{\Gamma \vdash \Pi x{:}A.B : s_2} \qquad \textit{if } (s_1, s_2) \in \boldsymbol{R}$$

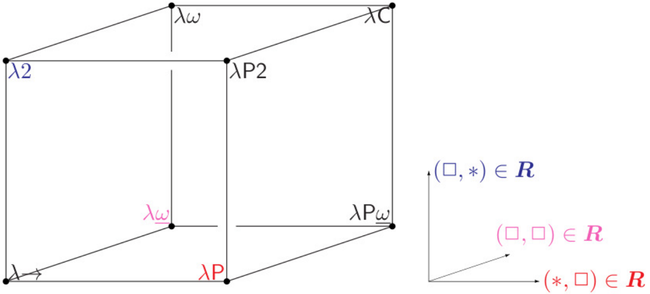
The β-cube: $\to_\beta$ + conv$_\beta$ + app$_\Pi$

(axiom)	$\langle\rangle \vdash * : \Box$
(start)	$\dfrac{\Gamma \vdash A : s \qquad x \notin \mathrm{DOM}(\Gamma)}{\Gamma, x{:}A \vdash x : A}$
(weak)	$\dfrac{\Gamma \vdash A : B \qquad \Gamma \vdash C : s \qquad x \notin \mathrm{DOM}(\Gamma)}{\Gamma, x{:}C \vdash A : B}$
(Π)	$\dfrac{\Gamma \vdash A : s_1 \qquad \Gamma, x{:}A \vdash B : s_2 \qquad (s_1, s_2) \in \boldsymbol{R}}{\Gamma \vdash \Pi_{x:A}.B : s_2}$
(λ)	$\dfrac{\Gamma, x{:}A \vdash b : B \qquad \Gamma \vdash \Pi_{x:A}.B : s}{\Gamma \vdash \lambda_{x:A}.b : \Pi_{x:A}.B}$
(conv$_\beta$)	$\dfrac{\Gamma \vdash A : B \qquad \Gamma \vdash B' : s \qquad B =_\beta B'}{\Gamma \vdash A : B'}$
(app$_\Pi$)	$\dfrac{\Gamma \vdash F : \Pi_{x:A}.B \qquad \Gamma \vdash a : A}{\Gamma \vdash Fa : B[x{:=}a]}$

	Simple	Poly-morphic	Depend-ent	Constr-uctors	Related system	Refs.
$\lambda\!\to$	$(*,*)$				λ^τ	[4, 2, 13]
$\lambda 2$	$(*,*)$	$(\square,*)$			F	[10, 29]
λP	$(*,*)$		$(*,\square)$		AUT-QE, LF	[6, 11]
$\lambda\underline{\omega}$	$(*,*)$			$(\square,\square)$	POLYREC	[28]
λP2	$(*,*)$	$(\square,*)$	$(*,\square)$			[23]
$\lambda\omega$	$(*,*)$	$(\square,*)$		$(\square,\square)$	Fω	[10]
λP$\underline{\omega}$	$(*,*)$		$(*,\square)$	$(\square,\square)$		
λC	$(*,*)$	$(\square,*)$	$(*,\square)$	$(\square,\square)$	CC	[5]

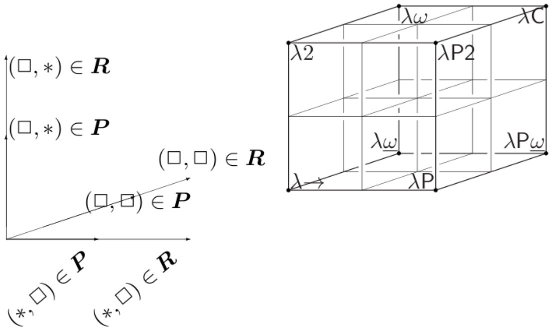
The 8 Systems

The Barendregt Cube



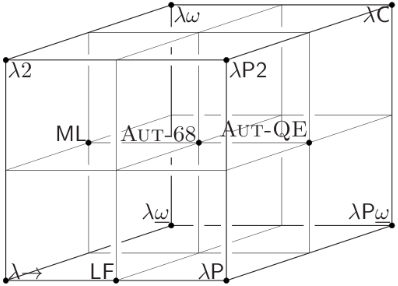
$\lambda\omega$
λC
$\lambda 2$
$\lambda P2$
$\lambda\underline{\omega}$
$\lambda P\underline{\omega}$
$\lambda\!\to$
λP
$(\square, *) \in \boldsymbol{R}$
$(\square, \square) \in \boldsymbol{R}$
$(*, \square) \in \boldsymbol{R}$

The refined Barendregt Cube



$(\Box, *) \in \boldsymbol{R}$
$(\Box, *) \in \boldsymbol{P}$
$(\Box, \Box) \in \boldsymbol{R}$
$(\Box, \Box) \in \boldsymbol{P}$
$(*, \Box) \in \boldsymbol{P}$
$(*, \Box) \in \boldsymbol{R}$
λω
λC
λ2
λP2
λω
λPω
λ→
λP

LF, ML, Aut-68, and Aut-QE in the refined Cube

- 

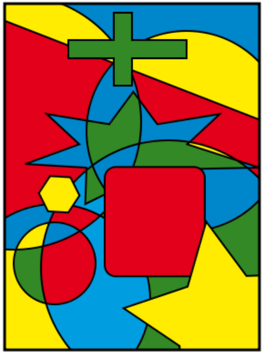
Fifty Years of a whole variety of proof assistants and programming languages

- The proof of the Kepler's conjecture was checked in a mixture of Isabelle and HOL Light.
- The proof of the four color theorem was checked in Coq.
- Landau book on foundations of Analysis was checked in Automath.
- 60% of the Compendium of Complete Lattices, was checked in Mizar.
- Numerous other provers/checkers. Why were they created? And what are they used for?
- Not to forget a huge number of programming lanaguages/paradigms each with a different purpose.

Kepler's conjecture

- Kepler 17th century: No packing of congruent balls in Eucledian space has density greater than the density of the face-centered cubic packing (74.04%).
- Sam Ferguson and Tom Hales proved it in 1998 but was not published until 2006
- The Flyspec project lasted over 10 years to give a computer proof of the Kepler conjecture.

Four colour theorem



- Given any separation of the plane into contiguous regions, producing a figure called a map, no more than four colours are required to color the regions of the map so that no two afjacent regions have the same color.

Four colour theorem

- 1976: This was the first major theorem proved using a computer: huge bits by hand, huge bits by computer.
- Proof not acceptable: computer assisted part infeasible for humans to check by hand.
- Revised version given in 1997.
- Fully computer checked in 2005.

Common Mathematical Language of mathematicians: CML

+ CML is *expressive*: it has linguistic categories like *proofs* and *theorems*.
+ CML has been refined by intensive use and is rooted in *long traditions*.
+ CML is *approved* by most mathematicians as a communication medium.
+ CML *accommodates many branches* of mathematics, and is adaptable to new ones.

- Since CML is based on natural language, it is *informal* and *ambiguous*.
- CML is *incomplete:* Much is left implicit, appealing to the reader's intuition.
- CML is *poorly organised:* In a CML text, many structural aspects are omitted.
- CML is *automation-unfriendly:* A CML text is a plain text and cannot be easily automated.

A CML-text

From chapter 1, § 2 of E. Landau's *Foundations of Analysis* (Landau 1930, 1951).

Theorem 6. [Commutative Law of Addition]

$$x + y = y + x.$$

Proof Fix y, and let $\mathfrak{M}$ be the set of all x for which the assertion holds.
I) We have

$$y + 1 = y',$$

and furthermore, by the construction in the proof of Theorem 4,

$$1 + y = y',$$

so that

$$1 + y = y + 1$$

and 1 belongs to $\mathfrak{M}$.
II) If x belongs to $\mathfrak{M}$, then

$$x + y = y + x,$$

Therefore

$$(x + y)' = (y + x)' = y + x'.$$

By the construction in the proof of Theorem 4, we have

$$x' + y = (x + y)',$$

hence

$$x' + y = y + x',$$

so that x' belongs to $\mathfrak{M}$. The assertion therefore holds for all x. □

The problem with formal logic

- No logical language is an alternative to CML
  - A logical language is *not universal* to all mathematicians, and is *not a good communication medium*.
  - Logical languages make fixed choices (*first versus higher order, predicative versus impredicative, constructive versus classical, types or sets*, etc.). But different parts of mathematics need different choices and there is no universal agreement as to which is the best formalism.
  - A logician reformulates in logic their *formalization* of a mathematical-text as a formal text which is structured considerably *unlike* the original, and is of little use to the *ordinary* mathematician.
  - Mathematicians do not want to use formal logic and have *for centuries* done mathematics without it.
- *So, mathematicians kept to CML.* But CML is difficult to computerise and formalise (either in a logical framework or computer system).
- We would like to find an alternative to CML which avoids some of the features of the logical languages which made them unattractive to mathematicians.

What are the options for computerization?

Computers can handle mathematical text at various levels:

- Images of pages may be stored. While useful, this is not a good representation of *language* or *knowledge*.
- Typesetting systems like LaTeX, TeXmacs, can be used.
- Document representations like OpenMath, OMDoc, MathML, can be used.
- Formal logics used by theorem provers (Coq, Isabelle, HOL, Mizar, Isar, etc.) can be used.

We are gradually developing a system named Mathlang which we hope will eventually allow building a bridge between the latter 3 levels.

This talk aims at discussing the motivations rather than the details.

The issues with typesetting systems

+ A system like LaTeX, TeXmacs, provides good defaults for visual appearance, while allowing fine control when needed.
+ LaTeX and TeXmacs support commonly needed document structures, while allowing custom structures to be created.
- Unless the mathematician is amazingly disciplined, the *logical structure of symbolic formulas is not represented* at all.
- The *logical structure of mathematics as embedded in natural language text is not represented.* Automated discovery of the semantics of natural language text is still too primitive and requires human oversight.

LaTeX example

draft documents	✓
public documents	✓
computations and proofs	✗

```
\begin{theorem}[Commutative Law of Addition]\label{theorem:6}
$$x+y=y+x.$$
\end {theorem}
\begin{proof}
Fix $y$, and $\mathfrak{M}$ be the set of all $x$ for which
the assertion holds.
\begin{enumerate}
\item We have $$y+1=y',$$
and furthermore, by the construction in
the proof of Theorem \ref{theorem:4}, $$1+y=y',$$
so that $$1+y=y+1$$
and $1$ belongs to $\mathfrak{M}$.
```

```latex
\item If $x$ belongs to $\mathfrak{M}$, then $$x+y=y+x,$$
Therefore
$$(x+y)'=(y+x)'=y+x'.$$
By the construction in the proof of
Theorem \ref{theorem:4}, we have $$x'+y=(x+y)',$$
hence
$$x'+y=y+x',$$
so that $x'$ belongs to $\mathfrak{M}$.
\end{enumerate}
The assertion therefore holds for all $x$.
\end{proof}
```

Full formalization difficulties: choices

A CML-text is structured differently from a fully formalized text proving the same facts. *Making the latter involves extensive knowledge and many choices:*

- The choice of the *underlying logical system.*
- The choice of *how concepts are implemented* (equational reasoning, equivalences and classes, partial functions, induction, etc.).
- The choice of the *formal foundation*: a type theory (dependent?), a set theory (ZF? FM?), a category theory? etc.
- The choice of the *proof checker*: Automath, Isabelle, Coq, PVS, Mizar, HOL, ...

An issue is that one must in general commit to one set of choices.

Full formalization difficulties: informality

Any informal reasoning in a CML-text will cause various problems when fully formalizing it:

- A single (big) step may need to expand into a (series of) syntactic proof expressions. *Very long expressions can replace a clear CML-text.*
- The entire CML-text may need *reformulation* in a fully *complete* syntactic formalism where every detail is spelled out. New details may need to be woven throughout the entire text. The text may need to be *turned inside out.*
- Reasoning may be obscured by *proof tactics*, whose meaning is often *ad hoc* and implementation-dependent.

Regardless, ordinary mathematicians do not find the new text useful.

Coq example

draft documents	✗
public documents	✗
computations and proofs	✓

From Module `Arith.Plus` of Coq standard library (`http://coq.inria.fr/`).

```
Lemma plus_sym: (n,m:nat)(n+m)=(m+n).
Proof.
Intros n m ; Elim n ; Simpl_rew ; Auto with arith.
Intros y H ; Elim (plus_n_-Sm m y) ; Simpl_rew ; Auto with arith.
Qed.
```

Mathlang's Goal: Open borders between mathematics, logic and computation

- Ordinary mathematicians *avoid* formal mathematical logic.
- Ordinary mathematicians *avoid* proof checking (via a computer).
- Ordinary mathematicians *may use* a computer for computation: there are over 1 million people who use Mathematica (including linguists, engineers, etc.).
- Mathematicians may also use other computer forms like Maple, LaTeX, etc.
- But we are not interested in only *libraries* or *computation* or *text editing*.
- We want *freedeom of movement* between mathematics, logic and computation.
- At every stage, we must have *the choice* of the level of formalilty and the depth of computation.

Aim for Mathlang? (Kamareddine and Wells 2001, 2002)

Can we formalise a mathematical text, avoiding as much as possible the ambiguities of natural language, while still guaranteeing the following four goals?

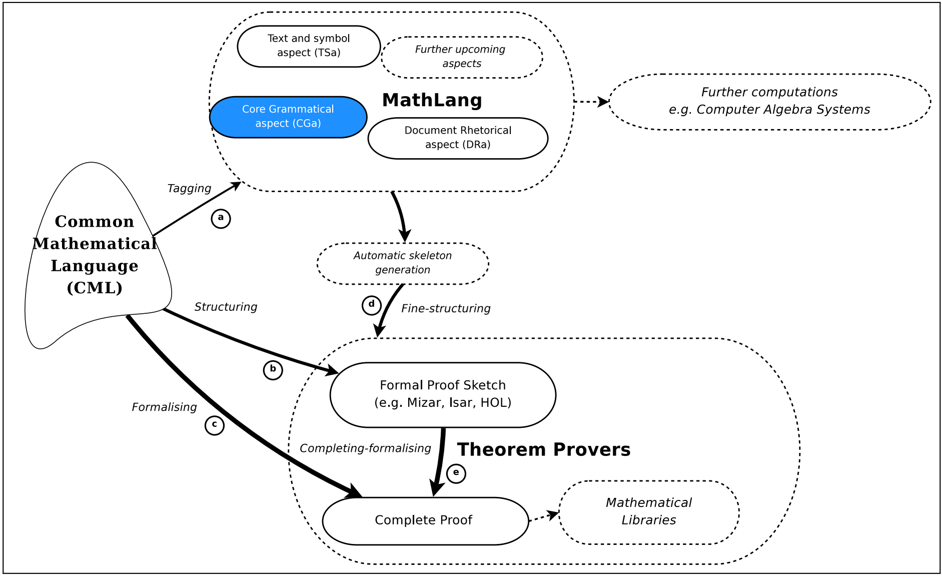
1. The formalised text looks very much like the original mathematical text (and hence the content of the original mathematical text is respected).
2. The formalised text can be fully manipulated and searched in ways that respect its mathematical structure and meaning.
3. Steps can be made to do computation (via computer algebra systems) and proof checking (via proof checkers) on the formalised text.
4. This formalisation of text is not much harder for the ordinary mathematician than LaTeX. *Full formalization down to a foundation of mathematics is not required*, although allowing and supporting this is one goal.

(No theorem prover's language satisfies these goals.)

	draft documents	✓
Mathlang	public documents	✓
	computations and proofs	✓

- A Mathlang text captures the grammatical and reasoning aspects of mathematical structure for further computer manipulation.
- A *weak type system* checks Mathlang documents at a grammatical level.
- A Mathlang text remains *close* to its CML original, allowing confidence that the CML has been captured correctly.
- We have been developing ways to weave natural language text into Mathlang.
- Mathlang aims to eventually support *all encoding uses*.
- The CML view of a Mathlang text should match the mathematician's intentions.
- The formal structure should be suitable for various automated uses.

Example of a MathLang Path


Text and symbol aspect (TSa)
Further upcoming aspects
MathLang
Core Grammatical aspect (CGa)
Document Rhetorical aspect (DRa)
Further computations e.g. Computer Algebra Systems
Common Mathematical Language (CML)
Tagging
a
Automatic skeleton generation
d
Fine-structuring
Structuring
b
Formal Proof Sketch (e.g. Mizar, Isar, HOL)
Formalising
c
Completing-formalising
Theorem Provers
e
Complete Proof
Mathematical Libraries

What is CGa? (Maarek's PhD thesis)

- CGa is a formal language derived from MV (N.G. de Bruijn 1987) and WTT (Kamareddine and Nederpelt 2004) which aims at expliciting the grammatical role played by the elements of a CML text.
- The structures and common concepts used in CML are captured by CGa with a finite set of grammatical/linguistic/syntactic categories: *Term* "$\sqrt{2}$", *set* "$\mathbb{Q}$", *noun* "number", *adjective* "even", *statement* "$a = b$", *declaration* "Let a be a number", *definition* "An even number is..", *step* "a is odd, hence $a \neq 0$", *context* "Assume a is even".
  term set noun adjective statement declaration definition step context .
- Generally, each syntactic category has a corresponding *weak type*.
- CGa's type system derives typing judgments to check whether the reasoning parts of a document are coherently built.

Weak Type Theory

In Weak Type Theory (or $\textrm{W}\textsc{tt}$) we have the following linguistic categories:

- On the *atomic* level: *variables*, *constants* and *binders*,
- On the *phrase* level: *terms* $\mathcal{T}$, *sets* $\mathbb{S}$, *nouns* $\mathcal{N}$ and *adjectives* $\mathcal{A}$,
- On the *sentence* level: *statements* P and *definitions* $\mathcal{D}$,
- On the *discourse* level: *contexts* $\mathbf{\Gamma}$, *lines* $\mathbf{l}$ and *books* $\mathbf{B}$.

Categories of syntax of WTT

Other category	abstract syntax	symbol
expressions	$\mathcal{E} = T \vert \mathbb{S} \vert \mathcal{N} \vert P$	E
parameters	$\mathcal{P} = T \vert \mathbb{S} \vert P$ (note: $\overrightarrow{\mathcal{P}}$ is a list of $\mathcal{P}$s)	P
typings	$\mathbf{T} = \mathbb{S} : \text{ SET } \vert \mathcal{S} : \text{ STAT } \vert T : \mathbb{S} \vert T : \mathcal{N} \vert T : \mathcal{A}$	T
declarations	$\mathcal{Z} = \mathtt{V}^S : \text{ SET } \vert \mathtt{V}^P : \text{ STAT } \vert \mathtt{V}^T : \mathbb{S} \vert \mathtt{V}^T : \mathcal{N}$	Z

level	category	abstract syntax	symbol
atomic	*variables*	$\mathtt{V} = \mathtt{V}^T \mid \mathtt{V}^S \mid \mathtt{V}^P$	x
	constants	$\mathtt{C} = \mathtt{C}^T \mid \mathtt{C}^S \mid \mathtt{C}^N \mid \mathtt{C}^A \mid \mathtt{C}^P$	c
	binders	$\mathtt{B} = \mathtt{B}^T \mid \mathtt{B}^S \mid \mathtt{B}^N \mid \mathtt{B}^A \mid \mathtt{B}^P$	b
phrase	*terms*	$T = \mathtt{C}^T(\overrightarrow{\mathcal{P}}) \mid \mathtt{B}^T_{\mathcal{Z}}(\mathcal{E}) \mid \mathtt{V}^T$	t
	sets	$\mathbb{S} = \mathtt{C}^S(\overrightarrow{\mathcal{P}}) \mid \mathtt{B}^S_{\mathcal{Z}}(\mathcal{E}) \mid \mathtt{V}^S$	s
	nouns	$\mathcal{N} = \mathtt{C}^N(\overrightarrow{\mathcal{P}}) \mid \mathtt{B}^N_{\mathcal{Z}}(\mathcal{E}) \mid \mathcal{A}\mathcal{N}$	n
	adjectives	$\mathcal{A} = \mathtt{C}^A(\overrightarrow{\mathcal{P}}) \mid \mathtt{B}^A_{\mathcal{Z}}(\mathcal{E})$	a
sentence	*statements*	$P = \mathtt{C}^P(\overrightarrow{\mathcal{P}}) \mid \mathtt{B}^P_{\mathcal{Z}}(\mathcal{E}) \mid \mathtt{V}^P$	S
	definitions	$\mathcal{D} = \mathcal{D}^{\varphi} \mid \mathcal{D}^P$	D
		$\mathcal{D}^{\varphi} = \mathtt{C}^T(\overrightarrow{V}) := T \mid \mathtt{C}^S(\overrightarrow{V}) := \mathbb{S} \mid$	
		$\mathtt{C}^N(\overrightarrow{V}) := \mathcal{N} \mid \mathtt{C}^A(\overrightarrow{V}) := \mathcal{A}$	
		$\mathcal{D}^P = \mathtt{C}^P(\overrightarrow{V}) := P$	
discourse	*contexts*	$\mathbf{\Gamma} = \emptyset \mid \mathbf{\Gamma}, \mathcal{Z} \mid \mathbf{\Gamma}, P$	Γ
	lines	$\mathbf{l} = \mathbf{\Gamma} \triangleright P \mid \mathbf{\Gamma} \triangleright \mathcal{D}$	l
	books	$\mathbf{B} = \emptyset \mid \mathbf{B} \circ \mathbf{l}$	B

Derivation rules

(1) B is a weakly well-typed book: $\vdash \ B :: \texttt{book}.$

(2) Γ is a weakly well-typed context relative to book B: $B \ \vdash \ \Gamma :: \texttt{cont}.$

(3) t is a weakly well-typed term, etc., relative to book B and context Γ:

$$\begin{array}{lll} B;\Gamma \ \vdash \ t :: T, & B;\Gamma \ \vdash \ s :: S, & B;\Gamma \ \vdash \ n :: N, \\ B;\Gamma \ \vdash \ a :: A, & B;\Gamma \ \vdash \ p :: P, & B;\Gamma \ \vdash \ d :: D \end{array}$$

$OK(B;\Gamma).$ stands for: $\vdash \ B :: \texttt{book}$, *and* $B \ \vdash \ \Gamma :: \texttt{cont}$

Examples of derivation rules

- $\mathtt{dvar}(\emptyset) = \emptyset \qquad \mathtt{dvar}(\Gamma', x : W) = \mathtt{dvar}(\Gamma'), x$
  $\mathtt{dvar}(\Gamma', P) = \mathtt{dvar}(\Gamma')$

$$\frac{OK(B;\Gamma), \quad x \in \mathtt{V}^{\mathtt{T/S/P}}, \quad x \in \mathtt{dvar}(\Gamma)}{B;\Gamma \ \vdash \ x :: T/S/P} \quad (var)$$

$$\frac{B;\Gamma \ \vdash \ n :: N, \quad B;\Gamma \ \vdash \ a :: A}{B;\Gamma \ \vdash \ an :: N} \quad (adj{-}noun)$$

$$\frac{}{\vdash \ \emptyset :: \mathtt{book}} \quad (emp{-}book)$$

$$\frac{B;\Gamma \ \vdash \ p :: P}{\vdash \ B \circ \Gamma \rhd p :: \mathtt{book}} \qquad \frac{B;\Gamma \ \vdash \ d :: D}{\vdash \ B \circ \Gamma \rhd d :: \mathtt{book}}$$

$$(book{-}ext)$$

Properties of WTT

- *Every variable is declared* If $B;\Gamma \ \vdash \ \Phi :: \mathbf{W}$ then $FV(\Phi) \subseteq \mathtt{dvar}(\Gamma)$.
- *Correct subcontexts* If $B \ \vdash \ \Gamma :: \mathtt{cont}$ and $\Gamma' \subseteq \Gamma$ then $B \ \vdash \ \Gamma' :: \mathtt{cont}$.
- *Correct subbooks* If $\ \vdash \ B :: \mathtt{book}$ and $B' \subseteq B$ then $\ \vdash \ B' :: \mathtt{book}$.
- *Free constants are either declared in book or in contexts* If $B;\Gamma \ \vdash \ \Phi :: \mathbf{W}$, then $FC(\Phi) \subseteq \mathtt{prefcons}(B) \cup \mathtt{defcons}(B)$.
- *Types are unique* If $B;\Gamma \ \vdash \ A :: \mathbf{W_1}$ and $B;\Gamma \ \vdash \ A :: \mathbf{W_2}$, then $\mathbf{W_1} \equiv \mathbf{W_2}$.
- *Weak type checking is decidable* there is a decision procedure for the question $B;\Gamma \ \vdash \ \Phi :: \mathbf{W}$?.
- *Weak typability is computable* there is a procedure deciding whether an answer exists for $B;\Gamma \ \vdash \ \Phi :: ?$ and if so, delivering the answer.

Definition unfolding

- Let $\vdash B ::$ book and $\Gamma \triangleright c(x_1, \ldots, x_n) := \Phi$ a line in B.
- We write $B \vdash c(P_1, \ldots, P_n) \overset{\delta}{\to} \Phi[x_i := P_i]$.
- *Church-Rosser* If $B \vdash \Phi \overset{\delta}{\twoheadrightarrow} \Phi_1$ and $B \vdash \Phi \overset{\delta}{\twoheadrightarrow} \Phi_2$ then there exists Φ_3 such that $B \vdash \Phi_1 \overset{\delta}{\twoheadrightarrow} \Phi_3$ andf $B \vdash \Phi_2 \overset{\delta}{\twoheadrightarrow} \Phi_3$.
- *Strong Normalisation* Let $\vdash B ::$ book. For all subformulas Ψ occurring in B, relation $\overset{\delta}{\to}$ is strongly normalizing (i.e., definition unfolding inside a well-typed book is a well-founded procedure).

CGa Weak Type Checking

Let $\mathfrak{M}$ be a set,

y and x are natural numbers,

if x belongs to $\mathfrak{M}$

then $x + y = y + x$

CGa Weak Type checking detects grammatical errors

Let $\mathfrak{M}$ be a set,

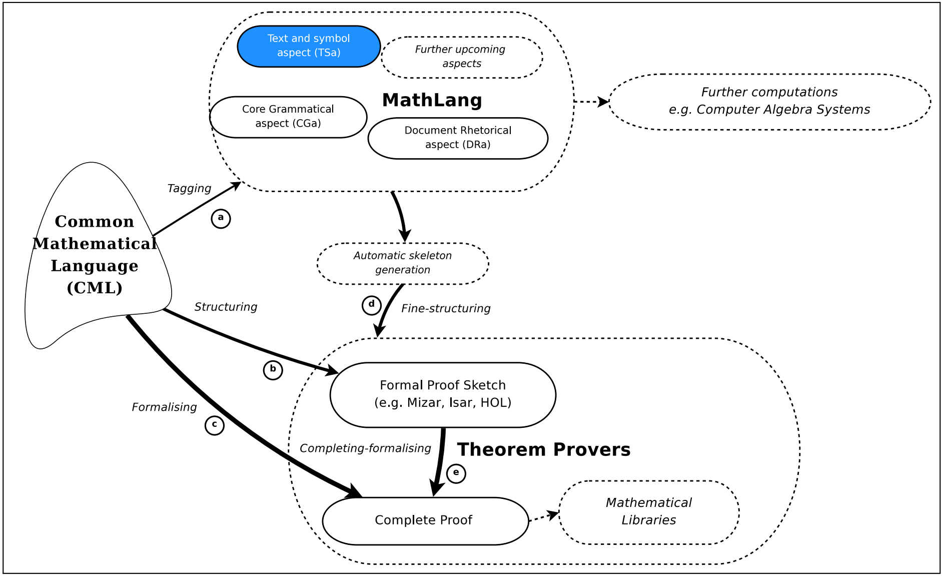
y and x are natural numbers,

if x belongs to $\mathfrak{M}$

then $x + y$ $\Leftarrow$ **error**

How complete is the CGa?

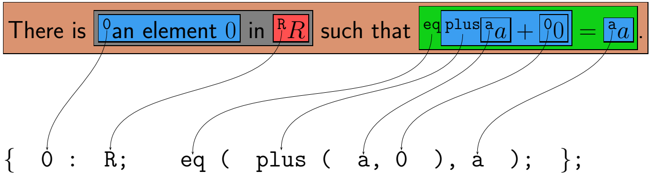
- CGa is quite advanced but remains under development according to new translations of mathematical texts. Are the current CGa categories sufficient?
- The metatheory of WTT has been established in (Kamareddine and Nederepelt 2004). That of CGa remains to be established. However, since CGa is quite similar to WTT, its metatheory might be similar to that of WTT.
- The type checker for CGa works well and gives some useful error messages. Error messages should be improved.


Text and symbol aspect (TSa)
Further upcoming aspects
MathLang
Core Grammatical aspect (CGa)
Document Rhetorical aspect (DRa)
Further computations e.g. Computer Algebra Systems
Common Mathematical Language (CML)
Tagging
a
Automatic skeleton generation
Structuring
d
Fine-structuring
b
Formal Proof Sketch (e.g. Mizar, Isar, HOL)
Formalising
c
Completing-formalising
Theorem Provers
e
Complete Proof
Mathematical Libraries

What is TSa? Lamar's PhD thesis

- TSa builds the bridge between a CML text and its grammatical interpretation and adjoins to each CGa expression a string of words and/or symbols which aims to act as its CML representation.
- *TSa plays the role of a user interface*
- *TSa can flexibly represent natural language mathematics.*
- The author wraps the natural language text with boxes representing the grammatical categories (as we saw before).
- *The author can also give interpretations to the parts of the text.*

Interpretations

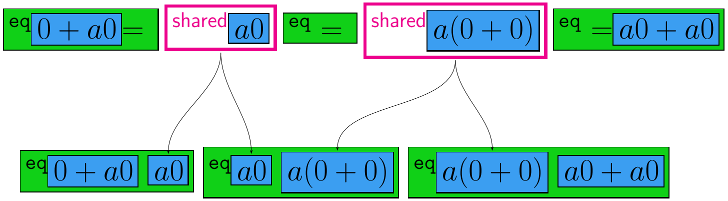


There is an element 0 in R such that $a + 0 = a$.

```
{  0 :  R;    eq (  plus (  a, 0  ), a  );  };
```

Rewrite rules enable natural language representation

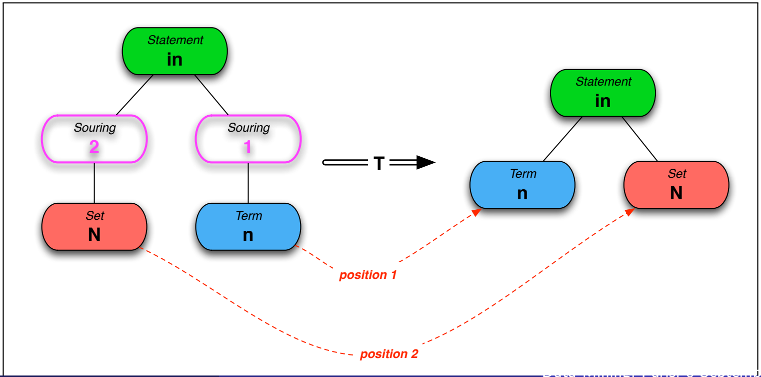
Take the example $0 + a0 = a0 = a(0 + 0) = a0 + a0$



reordering/position Souring

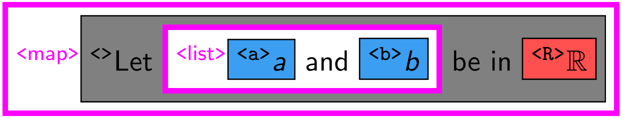
<in> <n> n $\in$ <N> $\mathbb{N}$

$ann =$ <in> <2> <N> $\mathbb{N}$ contains <1> <n> n



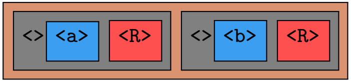
map souring

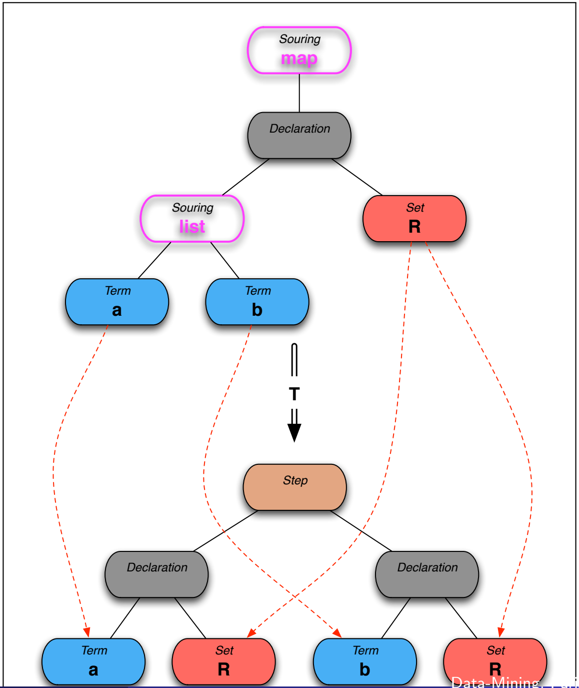
$ann =$



This is expanded to

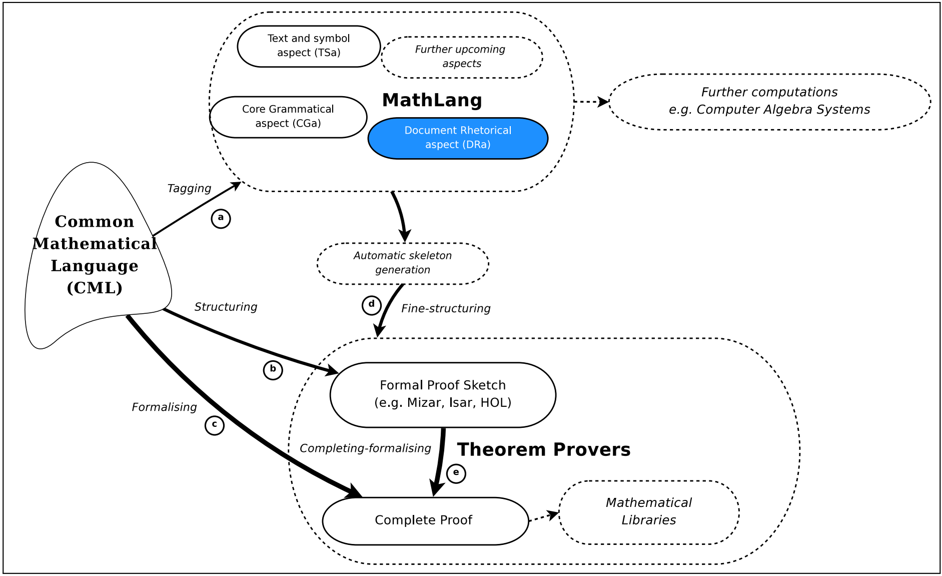
$\mathbf{T}\,(ann) =$




Souring
map
Declaration
Souring
list
Set
R
Term
a
Term
b
T
Step
Declaration
Declaration
Term
a
Set
R
Term
b
Set
R

How complete is TSa?

- TSa provides useful interface facilities but it is still under development.
- So far, only simple rewrite (souring) rules are used and they are not comprehensive. E.g., unable to cope with things like $\overbrace{x = \ldots = x}^{n \text{ times}}$.
- The TSa theory and metatheory need development.


Text and symbol aspect (TSa)
Further upcoming aspects
Core Grammatical aspect (CGa)
MathLang
Document Rhetorical aspect (DRa)
Further computations e.g. Computer Algebra Systems
Common Mathematical Language (CML)
Tagging
a
Automatic skeleton generation
d
Fine-structuring
Structuring
b
Formalising
c
Formal Proof Sketch (e.g. Mizar, Isar, HOL)
Completing-formalising
Theorem Provers
e
Complete Proof
Mathematical Libraries

What is DRa? Retel's PhD thesis

- DRa Document Rhetorical structure aspect.
- **Structural components of a document** like *chapter, section, subsection, etc.*
- **Mathematical components of a document** like *theorem, corollary, definition, proof, etc.*
- **Relations** between above components.
- These enhance readability, and ease the navigation of a document.
- Also, these help to go into more formal versions of the document.

Relations

Description
Instances of the StructuralRhetoricalRole class: preamble, part, chapter, section, paragraph, *etc.*
Instances of the MathematicalRhetoricalRole class: lemma, corollary, theorem, conjecture, definition, axiom, claim, proposition, assertion, proof, exercise, example, problem, solution, *etc.*

Relation
Types of relations: relatesTo, uses, justifies, subpartOf, inconsistentWith, exemplifies

What does the mathematician do?

- The mathematician wraps into boxes and uniquely names chunks of text
- The mathematician assigns to each box the structural and/or mathematical rhetorical roles
- The mathematician indicates the relations between wrapped chunks of texts

Lemma 1. For $m, n \in \mathbb{N}$ one has: $m^2 = 2n^2 \implies m = n = 0$.
Define on $\mathbb{N}$ the predicate:

$$P(m) \iff \exists n. m^2 = 2n^2 \ \& \ m > 0.$$

Claim. $P(m) \implies \exists m' < m. P(m')$. Indeed suppose $m^2 = 2n^2$ and $m > 0$. It follows that m^2 is even, but then m must be even, as odds square to odds. So $m = 2k$ and we have

$$2n^2 = m^2 = 4k^2 \implies n^2 = 2k^2$$

Since $m > 0$, if follows that $m^2 > 0, n^2 > 0$ and $n > 0$. Therefore $P(n)$. Moreover, $m^2 = n^2 + n^2 > n^2$, so $m^2 > n^2$ and hence $m > n$. So we can take $m' = n$.

By the claim $\forall m \in \mathbb{N}. \neg P(m)$, since there are no infinite descending sequences of natural numbers.

Now suppose $m^2 = 2n^2$ with $m \neq 0$. Then $m > 0$ and hence $P(m)$. Contradiction. Therefore $m = 0$. But then also $n = 0$.

Corollary 1. $\sqrt{2} \notin \mathbb{Q}$.
Suppose $\sqrt{2} \in \mathbb{Q}$, i.e. $\sqrt{2} = p/q$ with $p \in \mathbb{Z}, q \in \mathbb{Z} - \{0\}$. Then $\sqrt{2} = m/n$ with $m = |p|, n = |q| \neq 0$. It follows that $m^2 = 2n^2$. But then $n = 0$ by the lemma. Contradiction shows that $\sqrt{2} \notin \mathbb{Q}$.

Barendregt

Lemma 1.

For $m, n \in \mathbb{N}$ one has: $m^2 = 2n^2 \implies m = n = 0$ $\boxed{\text{A}}$

Proof.

Define on $\mathbb{N}$ the predicate: $\boxed{\text{E}}$

$$P(m) \iff \exists n. m^2 = 2n^2 \ \& \ m > 0.$$

Claim. $P(m) \implies \exists m' < m. P(m')$. $\boxed{\text{F}}$

Indeed suppose $m^2 = 2n^2$ and $m > 0$. It follows that m^2 is even, but then m must be even, as odds square to odds. $\boxed{\text{G}}$ So $m = 2k$ and we have $2n^2 = m^2 = 4k^2 \implies n^2 = 2k^2$ Since $m > 0$, if follows that $m^2 > 0$, $n^2 > 0$ and $n > 0$. Therefore $P(n)$. Moreover, $m^2 = n^2 + n^2 > n^2$, so $m^2 > n^2$ and hence $m > n$. So we can take $m' = n$. $\boxed{\text{B}}$

By the claim $\forall m \in \mathbb{N}. \neg P(m)$, since there are no infinite descending sequences of natural numbers.

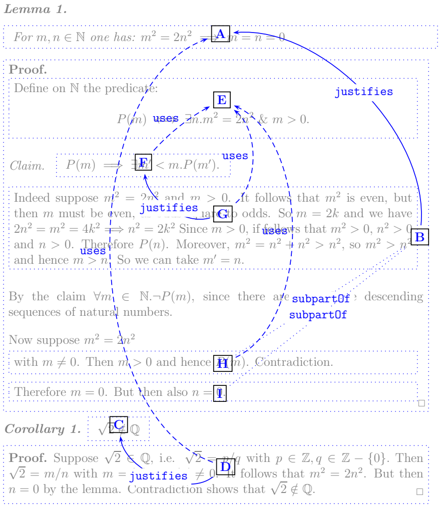
Now suppose $m^2 = 2n^2$

with $m \neq 0$. Then $m > 0$ and hence $P(m)$. Contradiction. $\boxed{\text{H}}$

Therefore $m = 0$. But then also $n = 0$. $\boxed{\text{I}}$ □

Corollary 1. $\sqrt{2} \notin \mathbb{Q}$ $\boxed{\text{C}}$

Proof. Suppose $\sqrt{2} \in \mathbb{Q}$, i.e. $\sqrt{2} = p/q$ with $p \in \mathbb{Z}, q \in \mathbb{Z} - \{0\}$. Then $\sqrt{2} = m/n$ with $m = |p|, n = |q| \neq 0$. It follows that $m^2 = 2n^2$. But then $n = 0$ by the lemma. Contradiction shows that $\sqrt{2} \notin \mathbb{Q}$. $\boxed{\text{D}}$ □



Lemma 1.

For $m, n \in \mathbb{N}$ one has: $m^2 = 2n^2 \implies m = n = 0$

Proof. Define on $\mathbb{N}$ the predicate:

$$P(m) \iff \exists n. m^2 = 2n^2 \;\&\; m > 0.$$

Claim. $P(m) \implies \exists m' < m. P(m')$.

Indeed suppose $m^2 = 2n^2$ and $m > 0$. It follows that m^2 is even, but then m must be even, [illegible] to odds. So $m = 2k$ and we have $2n^2 = m^2 = 4k^2 \implies n^2 = 2k^2$ Since $m > 0$, it follows that $m^2 > 0$, $n^2 > 0$ and $n > 0$. Therefore $P(n)$. Moreover, $m^2 = n^2 + n^2 > n^2$, so $m^2 > n^2$ and hence $m > n$. So we can take $m' = n$.

By the claim $\forall m \in \mathbb{N}.\neg P(m)$, since there are [illegible] descending sequences of natural numbers.

Now suppose $m^2 = 2n^2$ with $m \neq 0$. Then $m > 0$ and hence $P(m)$. Contradiction.

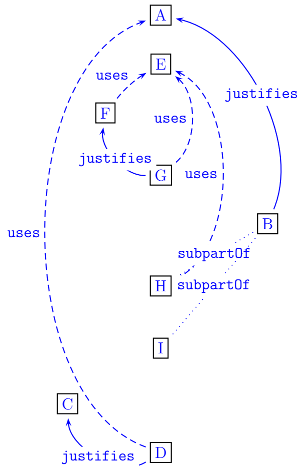
Therefore $m = 0$. But then also $n = 0$. □

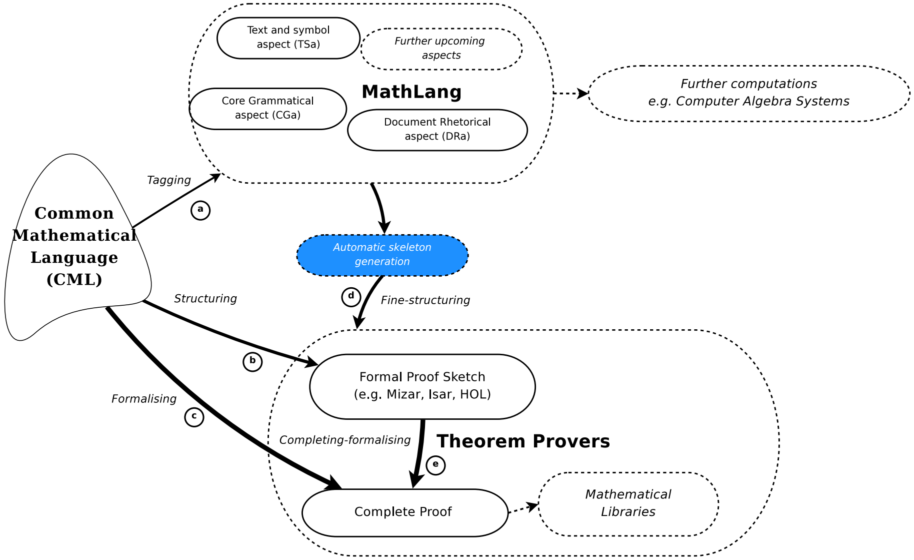
Corollary 1. $\sqrt{2} \notin \mathbb{Q}$

Proof. Suppose $\sqrt{2} \in \mathbb{Q}$, i.e. $\sqrt{2} = p/q$ with $p \in \mathbb{Z}, q \in \mathbb{Z} - \{0\}$. Then $\sqrt{2} = m/n$ with $m =$ [illegible] $\neq 0$. It follows that $m^2 = 2n^2$. But then $n = 0$ by the lemma. Contradiction shows that $\sqrt{2} \notin \mathbb{Q}$. □

The automatically generated dependency Graph

Dependency Graph (DG)




Text and symbol aspect (TSa)
Further upcoming aspects
MathLang
Core Grammatical aspect (CGa)
Document Rhetorical aspect (DRa)
Further computations e.g. Computer Algebra Systems
Common Mathematical Language (CML)
Tagging
a
Automatic skeleton generation
d
Fine-structuring
Structuring
b
Formal Proof Sketch (e.g. Mizar, Isar, HOL)
Formalising
c
Completing-formalising
Theorem Provers
e
Complete Proof
Mathematical Libraries

An example of a full formalisation in Coq via MathLang

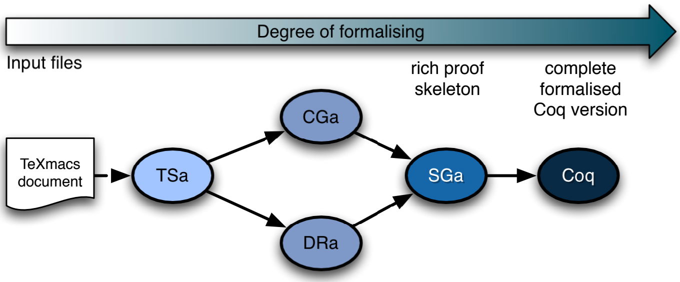
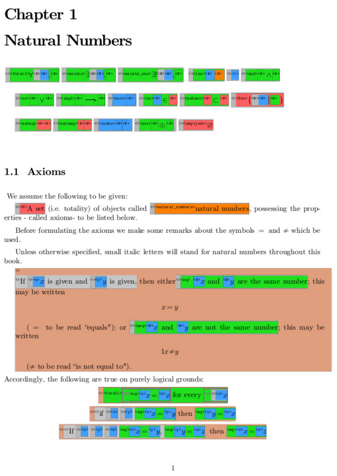


Figure 4: The path for processing the Landau chapter



Chapter 1
Natural Numbers

<> <forall> ∀ <#> <#> . <#>   <> <exists> ∃ <#> <#> . <#>   <> <exists_one> ∃! <#> <#> . <#>   <> <isa> <#> <#>   <> <1>   <> <and> <#> ∧ <#>

<> <or> <#> ∨ <#>   <> <impl> <#> ⟶ <#>   <> <succ> <#>   <> <in> <#> ∈ <#>   <> <subset> <#> ⊂ <#>   <> <Set> { <#> <#> | <#> }

<> <seteq> <#> <#>   <> <setneq> <#> <#>   <> <index> <#> <#>   <> <xor> <#> ⊕ <#>   <> <emptyset> ∅

1.1 Axioms

We assume the following to be given:

<> <N> A set (i.e. totality) of objects called <> <natural_numbers> natural numbers, possessing the properties - called axioms- to be listed below.

Before formulating the axioms we make some remarks about the symbols = and ≠ which be used.

Unless otherwise specified, small italic letters will stand for natural numbers throughout this book.

<>

<> If <> <x> x is given and <> <y> y is given, then either <> <eq> <#> x and <#> y are the same number; this may be written

$$x = y$$

(= to be read "equals"); or <> <neq> <#> x and <#> y are not the same number; this may be written

$$1x \neq y$$

(≠ to be read "is not equal to").

Accordingly, the following are true on purely logical grounds:

<> <forall> <2> <eq> <x> x = <x> x for every <1> <> <x> x

<> <> if <> <x> <> <y> <eq> <x> x = <y> y then <eq> <y> y = <x> x

<> <> If <> <x> <> <y> <> <z> <eq> <x> x = <y> y, <eq> <y> y = <z> z then <eq> <x> x = <z> z

1

Chapter 1 of Landau

- 5 axioms which we annotate with the mathematical role “axiom”, and give them the names“ax11” - “ax15”.
- 6 definitions which we annotate with the mathematical role “definition”, and give them names “def11” - “def16”.
- 36 nodes with the mathematical role “theorem”, named “th11” - “th136” and with proofs “pr11” - “pr136”.
- Some proofs are partitioned into an existential part and a uniqueness part.
- Other proofs consist of different cases which we annotate as unproved nodes with the mathematical role “case”.

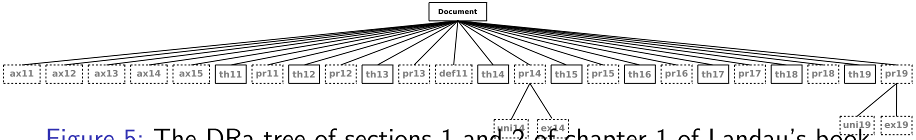


Figure 5: The DRa tree of sections 1 and 2 of chapter 1 of Landau's book

- The relations are annotated in a straightforward manner.
- Each proof *justifies* its corresponding theorem.
- Axiom 5 ("ax15") is the axiom of induction. So every proof which uses induction, *uses* also this axiom.
- Definition 1 ("def11") is the definition of addition. Hence every node which uses addition also *uses* this definition.
- Some theorems *use* other theorems via texts like: "By Theorem ...".
- In total we have 36 *justifies* relations, 154 uses relations, 6 caseOf, 3 existencePartOf and 3 uniquenessPartOf relations.
- The DG and GoTO are automatically generated.
- The GoTO is automatically checked and no errors result. So, we proceed to the next stage: automatically generating the SGa.

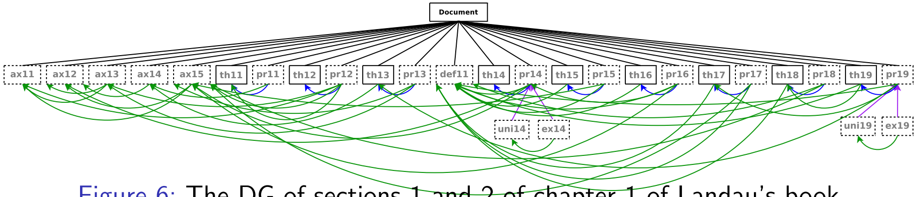
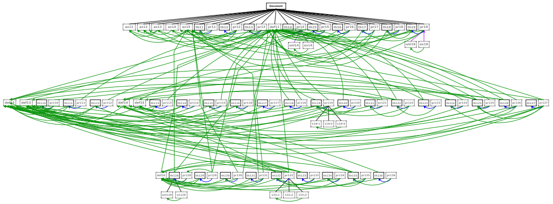


Figure 6: The DG of sections 1 and 2 of chapter 1 of Landau's book

DG of sections 1..4



With the help of the CGa annotations and the automatically generated rich proof skeleton, Zengler (who was not familiar with Coq) completed the Coq proofs of the whole of chapter one in a couple of hours.

Some points to consider

- MathLang aims to support automated processing of knowledge.
- MathLang aims to be independent of any foundation of mathematics.
- MathLang allows anyone to be involved, whether a mathematician, a computer engineer, a computer scientist, a linguist, a logician, etc.
- MathLang aims to allow any level of computerisation, from simple text processing, to calculations with a software tool (e.g., MatLab, Mathematica, SPSS, etc) to full formalisations and correctness (in Isabelle, Coq, Mizar, etc).
- MathLang allows processing of incomplete information (draft specifications rather than fully worked out details).
- MathLang can be used to check safety and correctness of software. E.g., brewery making, medical equipment, court cases, etc.

[1] Zena M. Ariola, Matthias Felleisen, John Maraist, Martin Odersky, and Philip Wadler. The call-by-need lambda calculus. pages 233–246.

[2] H[endrik] P[ieter] Barendregt. *The Lambda Calculus: Its Syntax and Semantics*. North-Holland, revised edition, 1984.

[3] R. Bloo, F. Kamareddine, and R. Nederpelt. The Barendregt Cube with Definitions and Generalised Reduction. *Information and Computation*, 126(2):123–143, 1996.

[4] Alonzo Church. A formulation of the simple theory of types. *J. Symbolic Logic*, 5:56–68, 1940.

[5] Thierry Coquand and Gérard Huet. The Calculus of Constructions. *Inform. & Comput.*, 76:95–120, 1988.

[6] N.G. de Bruijn. The mathematical language Automath – its usage and some of its extensions. In L. Laudet, D. Lacombe, and M. Schuetzenberger, editors, *Symposium on Automatic Demonstration*, volume 125 of *Lecture Notes in Mathematics*, pages 29–61, Heidelberg, 1970. Springer-Verlag. Reprinted in [25, A.2].

[7] Philippe de Groote. The conservation theorem revisited. pages 163–178.

[8] Gottlob Frege. *Begriffsschrift: eine der arithmetischen nachgebildete Formelsprache des reinen Denkens*. Nebert, Halle, 1879. Can be found on pp. 1–82 in [32].

[9] Gerhard Gentzen. Untersuchungen über das logische Schließen. *Mathematische Zeitschrift*, 39:176–210, 405–431, 1934.

[10] J[ean]-Y[ves] Girard. *Interprétation Fonctionnelle et Elimination des Coupures de l'Arithmétique d'Ordre Supérieur*. Thèse d'Etat, Université de Paris VII, 1972.

[11] R. Harper, F. Honsell, and G. Plotkin. A framework for defining logics. In *Proceedings Second Symposium on Logic in Computer Science*, pages 194–204, Washington D.C., 1987. IEEE.

[12] D. Hilbert and W. Ackermann. *Grundzüge der Theoretischen Logik*. Die Grundlehren der Mathematischen Wissenschaften in Einzeldarstellungen, Band XXVII. Springer Verlag, Berlin, first edition, 1928.

[13] J. Roger Hindley and Jonathan P. Seldin. *Introduction to Combinators and λ-calculus*, volume 1 of *London Mathematical Society Student Texts*. Cambridge University Press, 1986.

[14] F. Kamareddine, R. Bloo, and R. Nederpelt. On Π-conversion in the λ-cube and the combination with abbreviations. *Ann. Pure Appl. Logic*, 97(1–3):27–45, 1999.

[15] F. Kamareddine, T. Laan, and R. P. Nederpelt. Revisiting the λ-calculus notion of function. *J. Algebraic & Logic Programming*, 54:65–107, 2003.

[16] Fairouz Kamareddine. Postponement, conservation and preservation of strong normalisation for generalised reduction. *J. Logic Comput.*, 10(5):721–738, 2000.

[17] Fairouz Kamareddine and Rob Nederpelt. Refining reduction in the λ-calculus. *J. Funct. Programming*, 5(4):637–651, October 1995.

[18] Fairouz Kamareddine and Rob Nederpelt. A useful λ-notation. *Theoret. Comput. Sci.*, 155(1):85–109, 1996.

[19] Fairouz Kamareddine, Alejandro Ríos, and J. B. Wells. Calculi of generalised β-reduction and explicit substitutions: The type free and simply typed versions. *J. Funct. Logic Programming*, 1998(5), June 1998.

[20] A. J. Kfoury and J. B. Wells. A direct algorithm for type inference in the rank-2 fragment of the second-order λ-calculus. pages 196–207.

[21] A. J. Kfoury and J. B. Wells. New notions of reduction and non-semantic proofs of β-strong normalization in typed λ-calculi. pages 311–321.

[22] Assaf J. Kfoury, Jerzy Tiuryn, and Paweł Urzyczyn. An analysis of ML typability. *J. ACM*, 41(2):368–398, March 1994.

[23] G. Longo and E. Moggi. Constructive natural deduction and its modest interpretation. Technical Report CMU-CS-88-131, Carnegie Mellon University, Pittsburgh, USA, 1988.

[24] Rob Nederpelt. *Strong Normalization in a Typed Lambda Calculus With Lambda Structured Types*. PhD thesis, Technical University of Eindhoven, 1973.

[25] Rob Nederpelt, J. H. Geuvers, and Roel C. de Vrijer. *Selected Papers on Automath*, volume 133 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, Amsterdam, 1994.

[26] F.P. Ramsey. The foundations of mathematics. *Proceedings of the London Mathematical Society,* 2nd series, 25:338–384, 1926.

[27] Laurent Regnier. *Lambda calcul et réseaux*. PhD thesis, University Paris 7, 1992.

[28] G.R. Renardel de Lavalette. Strictness analysis via abstract interpretation for recursively defined types. *Information and Computation*, 99:154–177, 1991.

[29] J. C. Reynolds. Towards a theory of type structure. In *Colloque sur la Programmation*, volume 19 of *LNCS*, pages 408–425. Springer-Verlag, 1974.

[30] B. Russell. Letter to Frege. English translation in [32], pages 124–125, 1902.

[31] B. Russell. *The Principles of Mathematics*. Allen & Unwin, London, 1903.

[32] J. van Heijenoort. *From Frege to Gödel: A Source Book in Mathematical Logic, 1879–1931*. Harvard University Press, 1967.

[33] Alfred North Whitehead and Bertrand Russel. *Principia Mathematica*. Cambridge University Press, 1910–1913. In three volumes published from 1910 through 1913. Second edition published from 1925 through 1927. Abridged edition published in 1962.