Understanding Natural Language Mathematical Proofs

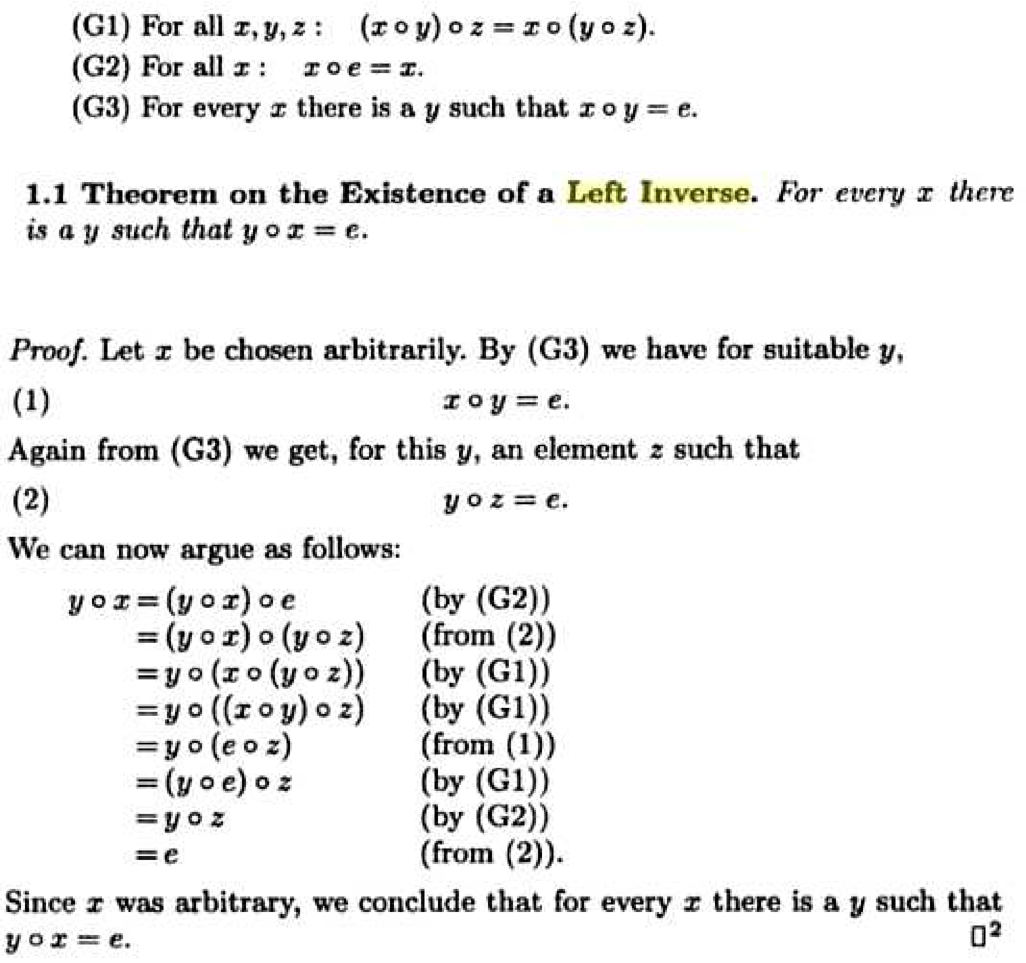
Peter Koepke, University of Bonn, Germany

CUNY Logic Workshop

New York City, September 4, 2009

Contents

Natural language proofs and formal proofs



(G1) For all x, y, z : $(x \circ y) \circ z = x \circ (y \circ z)$.
(G2) For all x : $x \circ e = x$.
(G3) For every x there is a y such that $x \circ y = e$.

1.1 Theorem on the Existence of a Left Inverse. *For every x there is a y such that $y \circ x = e$.*

Proof. Let x be chosen arbitrarily. By (G3) we have for suitable y,

(1) $$x \circ y = e.$$

Again from (G3) we get, for this y, an element z such that

(2) $$y \circ z = e.$$

We can now argue as follows:

$$\begin{aligned} y \circ x &= (y \circ x) \circ e && \text{(by (G2))} \\ &= (y \circ x) \circ (y \circ z) && \text{(from (2))} \\ &= y \circ (x \circ (y \circ z)) && \text{(by (G1))} \\ &= y \circ ((x \circ y) \circ z) && \text{(by (G1))} \\ &= y \circ (e \circ z) && \text{(from (1))} \\ &= (y \circ e) \circ z && \text{(by (G1))} \\ &= y \circ z && \text{(by (G2))} \\ &= e && \text{(from (2)).} \end{aligned}$$

Since x was arbitrary, we conclude that for every x there is a y such that $y \circ x = e$. □[2]

a formal proof is in general considerably longer than the corresponding mathematical proof. As an example we give here a formal proof of the theorem

$$\forall x \exists y\, y \circ x \equiv e$$

(existence of a left inverse) from the group axioms

$$\begin{aligned} \varphi_0 &:= \forall x \forall y \forall z (x \circ y) \circ z \equiv x \circ (y \circ z), \\ \varphi_1 &:= \forall x\, x \circ e \equiv x, \\ \varphi_2 &:= \forall x \exists y\, x \circ y \equiv e. \end{aligned}$$

The reader should compare the formal proof below with the mathematical proof of the same theorem in I.1.1. The "chain of equations" given there corresponds to the underlined formulas in the derivation up to line 23. For simplicity we shall write "xy" instead of "$x \circ y$" and we put $\Gamma := \varphi_0\, \varphi_1\, \varphi_2$.

1.	Γ	$\forall x\, xe \equiv x$	(Assm)
2.	Γ	$(yx)e \equiv yx$	5.5(a1) applied to 1. with $t = yx$
3.	Γ	$\underline{yx \equiv (yx)e}$	5.3(a) appl. to 2.
4.	$\Gamma\ e \equiv yz$	$yx \equiv (yx)(yz)$	(Sub) applied to 3.
5.	$\Gamma\ yz \equiv e$	$e \equiv yz$	5.3(a) and (Ant)
6.	$\Gamma\ yz \equiv e$	$\underline{yx \equiv (yx)(yz)}$	(Ant) and (Ch) appl. to 5. and 4.
7.	$\Gamma\ yz \equiv e$	$\forall x \forall y \forall z (xy)z \equiv x(yz)$	(Assm)
8.	$\Gamma\ yz \equiv e$	$\forall u \forall v (yu)v \equiv y(uv)$	5.5(a1) applied to 7. with $t = y$
9.	$\Gamma\ yz \equiv e$	$\forall w (yx)w \equiv y(xw)$	5.5(a1) applied to 8. with $t = x$
10.	$\Gamma\ yz \equiv e$	$(yx)(yz) \equiv y(x(yz))$	5.5(a1) applied to 9. with $t = yz$
11.	$\Gamma\ yz \equiv e$	$\underline{yx \equiv y(x(yz))}$	5.3(b) applied to 6. and 10.
12.	$\Gamma\ yz \equiv e\ x(yz) \equiv (xy)z$	$yx \equiv y((xy)z)$	(Sub) appl. to 11.
13.	$\Gamma\ yz \equiv e$	$(xy)z \equiv x(yz)$	5.5(a2) appl. three times to 7.
14.	$\Gamma\ yz \equiv e$	$x(yz) \equiv (xy)z$	5.3(a) appl. to 13.
15.	$\Gamma\ yz \equiv e$	$\underline{yx \equiv y((xy)z)}$	(Ch) applied to 14. and 12.
16.	$\Gamma\ yz \equiv e\ xy \equiv e$	$\underline{yx \equiv y(ez)}$	(Sub) appl. to 15.
17.	$\Gamma\ yz \equiv e\ xy \equiv e$	$(ye)z \equiv y(ez)$	with 5.5(a1) from φ_0 as for 10.
18.	$\Gamma\ yz \equiv e\ xy \equiv e$	$y(ez) \equiv (ye)z$	5.3(a) appl. to 17.
19.	$\Gamma\ yz \equiv e\ xy \equiv e$	$\underline{yx \equiv (ye)z}$	5.3(b) applied to 16. and 18.
20.	$\Gamma\ yz \equiv e\ xy \equiv e\ ye \equiv y$	$yx \equiv yz$	(Sub) appl. to 19.
21.	$\Gamma\ yz \equiv e\ xy \equiv e$	$ye \equiv y$	5.5(a1) applied to 1. with $t = y$ and (Ant)
22.	$\Gamma\ yz \equiv e\ xy \equiv e$	$\underline{yx \equiv yz}$	(Ch) applied to 21. and 20.
23.	$\Gamma\ xy \equiv e\ yz \equiv e$	$\underline{yx \equiv e}$	(Sub) and (Ant) applied to 22.
24.	$\Gamma\ xy \equiv e\ yz \equiv e$	$\exists y\, yx \equiv e$	($\exists$S) applied to 23.
25.	$\Gamma\ xy \equiv e\ \exists z\, yz \equiv e$	$\exists y\, yx \equiv e$	($\exists$A) applied to 24.
26.	$\Gamma\ xy \equiv e\ \forall y \exists z\, yz \equiv e$	$\exists y\, yx \equiv e$	5.5(b3) appl. to 25.
27.	$xy \equiv e$	$xy \equiv e$	(Assm)
28.	$xy \equiv e$	$\exists z\, xz \equiv e$	($\exists$S) applied to 27.
29.	$\exists y\, xy \equiv e$	$\exists z\, xz \equiv e$	($\exists$A) applied to 28.
30.	$\forall x \exists y\, xy \equiv e$	$\exists z\, xz \equiv e$	5.5(b3) appl. to 29.
31.	φ_2	$\forall y \exists z\, yz \equiv e$	5.5(b2) appl. to 30.
32.	$\Gamma\ xy \equiv e$	$\exists y\, yx \equiv e$	(Ant), (Ch) applied to 31. and 26.
33.	$\Gamma\ \forall x \exists y\, xy \equiv e$	$\exists y\, yx \equiv e$	($\exists$A) and 5.5(b3) applied to 32.
34.	Γ	$\forall x \exists y\, yx \equiv e$	(Ant) and 5.5(b4) applied to 33.

Formal proofs

N. Bourbaki

If formalized mathematics were as simple as the game of chess, then once our chosen formalized language had been described there would remain only the task of writing out our proofs in this language, [...] But the matter is far from being as simple as that, and no great experience is necessary to perceive that such a project is absolutely unrealizable: the tiniest proof at the beginnings of the Theory of Sets would already require several hundreds of signs for its complete formalization. [...] formalized mathematics cannot in practice be written down in full, [...] We shall therefore very quickly abandon formalized mathematics, [...]

Saunders Mac Lane

As to precision, we have now stated an absolute standard of rigor: A mathematical proof is rigorous when it is (or could be) written out in the first-order predicate language $L(\in)$ as a sequence of inferences from the axioms ZFC, each inference made according to one of the stated rules. [...] When a proof is in doubt, its repair is usually a partial approximation to the fully formal version.

Computer-supported formal proofs

J. McCarthy:

Checking mathematical proofs is potentially one of the most interesting and useful applications of automatic computers. Computers can check not only the proofs of new mathematical theorems but also proofs that complex engineering systems and computer programs meet their specifications. Proofs to be checked by computer may be briefer and easier to write than the informal proofs acceptable to mathematicians. This is because the computer can be asked to do much more work to check each step than a human is willing to do, and this permits longer and fewer steps. . . . The combination of proof-checking techniques with proof-finding heuristics will permit mathematicians to try out ideas for proofs that are still quite vague and may speed up mathematical research.

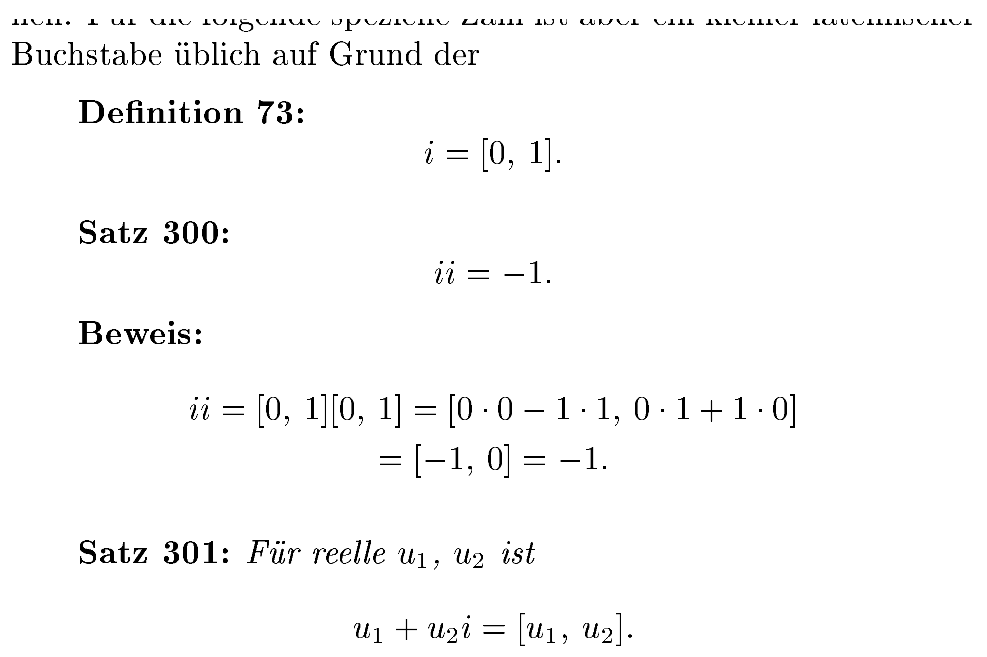
McCarthy, J. "Computer Programs for Checking Mathematical Proofs," Proceedings of the Symposium in Pure Math, Recursive Function Theory,Volume V, pages 219-228, AMS, Providence, RI, 1962.

Automatic proof checker

Automath (~1967)

N.G. de Bruijn

From the Automath formalization of E. Landau, *Grundlagen der Analysis*, 1930 by L. S. van Benthem Jutting, 1979:



hen. Für die folgende spezielle Zahl ist aber ein kleiner lateinischer
Buchstabe üblich auf Grund der

Definition 73:

$$i = [0,\ 1].$$

Satz 300:

$$ii = -1.$$

Beweis:

$$\begin{aligned} ii = [0,\ 1][0,\ 1] &= [0 \cdot 0 - 1 \cdot 1,\ 0 \cdot 1 + 1 \cdot 0] \\ &= [-1,\ 0] = -1. \end{aligned}$$

Satz 301: *Für reelle* $u_1,\ u_2$ *ist*

$$u_1 + u_2 i = [u_1,\ u_2].$$

```
ic:=pli(0,1rl):complex
+10300
t1:=tsis12a(0,1rl,0,1rl):is(ts(ic,ic),pli(mn"r"(ts"r"(0,0),ts"r"(1rl,1rl)),pl"r"(ts"r"(0,1
ts"r"(1rl,0))))
t2:=tris(real,mn"r"(ts"r"(0,0),ts"r"(1rl,1rl)),m0"r"(ts"r"(1rl,1rl)),m0"r"(1rl),pl01(ts"r"
m0"r"(ts"r"(1rl,1rl)),ts01(0,0,refis(real,0))),ism0"r"(ts"r"(1rl,1rl),1rl,satz195(1rl))):
is"r"(mn"r"(ts"r"(0,0),ts"r"(1rl,1rl)),m0"r"(1rl))
t3:=tris(real,pl"r"(ts"r"(0,1rl),ts"r"(1rl,0)),ts"r"(1rl,0),0,pl01(ts"r"(0,1rl),ts"r"(1rl,
ts01(0,1rl,refis(real,0))),ts02(1rl,0,refis(real,0))):is"r"(pl"r"(ts"r"(0,1rl),ts"r"(1rl,0
t4:=isrecx12(mn"r"(ts"r"(0,0),ts"r"(1rl,1rl)),m0"r"(1rl),pl"r"(ts"r"(0,1rl),
ts"r"(1rl,0)),0,t2,t3):is(pli(mn"r"(ts"r"(0,0),ts"r"(1rl,1rl)),
pl"r"(ts"r"(0,1rl),ts"r"(1rl,0))),cofrl(m0"r"(1rl)))
t5:=satz298j(1rl):is(cofrl(m0"r"(1rl)),m0(1c))
-10300
satz2300:=tr3is(cx,ts(ic,ic),pli(mn"r"(ts"r"(0,0),ts"r"(1rl,1rl)),
pl"r"(ts"r"(0,1rl),ts"r"(1rl,0))),cofrl(m0"r"(1rl)),m0(1c),t1".10300",t4".10300",t5".10300
is(ts(ic,ic),m0(1c))
```

The MIZAR system (1973 -) of Andrzej Trybulec

Language modeled after
''mathematical vernacular''

Natural deduction style

Automatic proof checker

Large mathematical library

Journal
Formalized Mathematics

www.mizar.org

MIZAR example in GOEDELCP.MIZ:

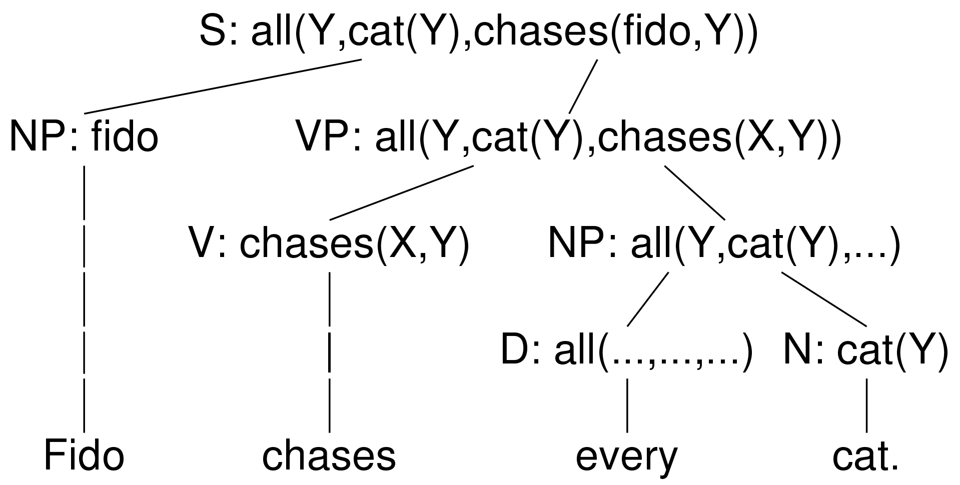
```
begin :: Goedel’s Completeness Theorem,
:: Ebb et al, Chapter V, Completeness The-
orem 4.1
theorem
  still_not-bound_in X is finite & X |= p
implies X |- p
  proof
    assume A1: still_not-bound_in X is
finite & X |= p;
    now assume not X |- p; then
      reconsider CX = X \/ {’not’ p} as Con-
sistent Subset of CQC-WFF
        by HENMODEL:9;
A2:  for A,J,v holds not J,v |= CX
      proof
        let A,J,v;
        now assume A3: J,v |= X \/ {’not’
p};
          now let q such that A4: q in X;
            X c= X \/ {’not’ p} by
XBOOLE_1:7;
            hence J,v |= q by
A3,A4,CALCUL_1:def 11;
          end; then
A5:       J,v |= X by CALCUL_1:def 11;
          now let q such that A6: q in
{’not’ p};
            {’not’ p} c= X \/ {’not’ p} by
XBOOLE_1:7;
            hence J,v |= q by
A3,A6,CALCUL_1:def 11;
          end; then
A7:       J,v |= {’not’ p} by CALCUL_1:def
11;
          ’not’ p in {’not’ p} by TARSKI:def
1; then
          J,v |= ’not’ p by A7,CALCUL_1:def
11; then
          J,v |= X & not J,v |= p by
A5,VALUAT_1:28;
          hence contradiction by
A1,CALCUL_1:def 12;
        end;
        hence not J,v |= CX;
      end;
      still_not-bound_in 'not' p is finite
by CQC_SIM1:20; then
      still_not-bound_in {'not' p} is finite
by Th26; then
      still_not-bound_in X \/
      still_not-bound_in {'not' p} is finite
by A1,FINSET_1:14; then
      still_not-bound_in CX is finite by
Th27; then
      consider CZ,JH1 such that A8: JH1,valH
|= CX by Th34;
      thus contradiction by A2,A8;
    end;
    hence thesis;
  end;
```

Mathematical statements

“1 divides every integer.” $\longleftrightarrow$ “Fido chases every cat.”

Linguistic analysis

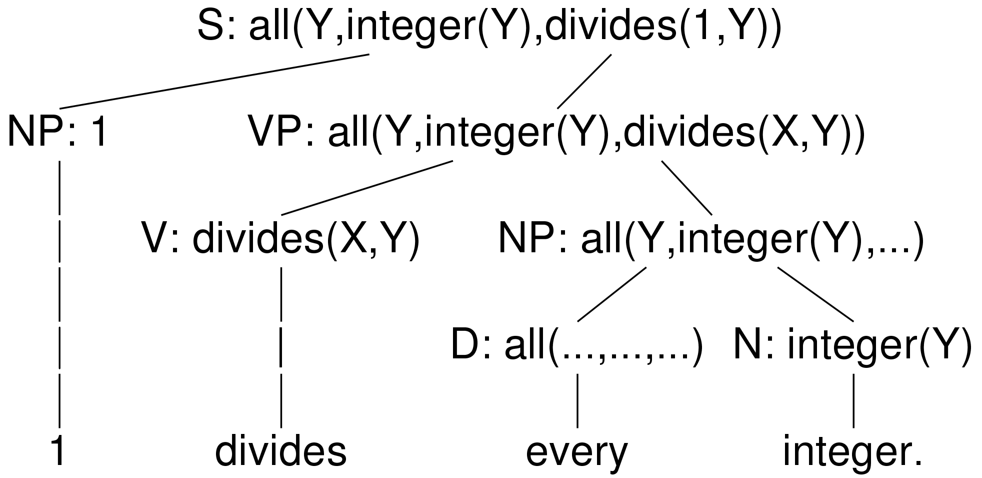
“Fido chases every cat.”



$\forall Y\,(\mathrm{cat}(Y) \rightarrow \mathrm{chases}(\mathrm{fido}, Y))$.

Linguistic analysis

“1 divides every integer.”


S: all(Y,integer(Y),divides(1,Y))
NP: 1
VP: all(Y,integer(Y),divides(X,Y))
V: divides(X,Y)
NP: all(Y,integer(Y),...)
D: all(...,...,...)
N: integer(Y)
1
divides
every
integer.

$\forall Y\,(\mathrm{integer}(Y) \rightarrow 1|Y)$.

Mathematical statements

– are (extended) natural language statements

– may contain formal terms and formulas (“semiformal language”)

– have a particular typography

– ambiguity is avoided, e.g. by using variables or specifically defined notions

– have a first-order meaning (formal semantics)

– can be parsed by standard NLP techniques (Natural Language Processing)

Mathematical texts

- consist of mathematical statements
- possess large-scale structures: definition / theorem / proof
- new notions can be defined
- assumptions can be introduced and retracted: ‘‘assume ...’’, ‘‘assume instead ...’’
- proof steps have to be justified from earlier statements (temporal character)
- justifications may contain explicit long-range references to earlier statements

What is a mathematical proof?

- description of the/some mathematical ‘‘reality’’?
- argumentative text about the/some mathematical ‘‘reality’’?
- argumentative text within some system of initial assumptions (axioms)?
- in part a formal derivation within some calculus?
- abbreviation for some (long) formal derivation?
- recipe for building a formal derivation if required?
- a formal derivation in some very rich formal system (Montague: English as a formal language)?

Ramifications of the proof-question

- Mathematical logic (claims to) model(s) the axiomatic method of modern mathematics. Mathematical logic is mathematically successful but it does not really reflect the actual languages and arguments of mathematicians.
- The gap between natural and formal mathematical proofs is a topic in the philosophy of mathematics.
- The language of mathematics as an expert language stands out by the fact that its intended semantics is in principle fully captured by a translation into first-order formulas. This makes the language of mathematics a paradigm for studies in theoretical and computational linguistics.
- Answering the proof-question may have practical applications for mathematical authoring and tutoring tools.

The Naproche project: Natural language proof checking

– studies the syntax and semantics of the language of proofs, emphasizing natural language and natural argumentation aspects

– models natural language proofs using computer-supported methods of formal linguistics and formal logic

– develops a mathematical authoring system with a LaTeX-quality graphical interface

– joint work with Bernhard Schröder, linguistics, and two graduate students; Bonn, Essen, Cologne

– www.naproche.net

The Naproche project: Natural language proof checking

– To devise a strictly formal system for mathematics, implemented by computer, whose input language is an extensive part of the common mathematical language, and whose proof style is close to proof styles found in the mathematical literature.

– The ... project Naproche aims at constructing a system which accepts a controlled but rich subset of ordinary mathematical language including TeX-style typeset formulas and transforms them into formal statements. We adapt linguistic techniques to allow for common grammatical constructs and to extract mathematically relevant implicit information about hypotheses and conclusions. Combined with proof checking software we obtain Natural language proof checkers ...

Layers of the Naproche system:

$\downarrow$ Standard or web editor; or WYSIWYG mathematics editor (TEX$_{\text{MACS}}$)

TeX-style input text

$\updownarrow$ Natural language processing (NLP)

Proof representation structure (PRS)

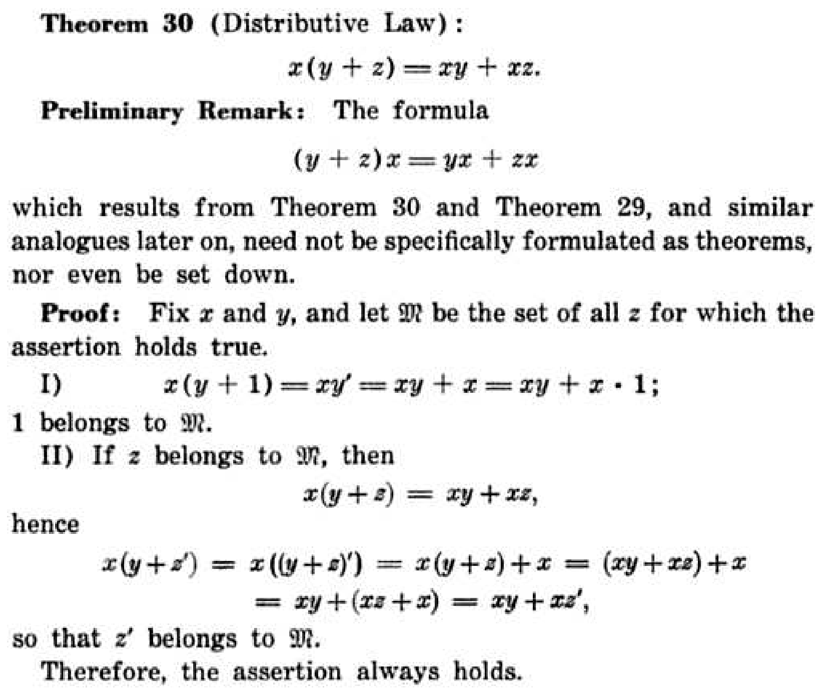
$\updownarrow$ First-order translation

First-order logic format (TPTP)

$\updownarrow$ Proof checker or automatic theorem prover (ATP)

“Accepted”/“Not accepted”, with error messages

E. Landau, *Grundlagen der Analysis*, 1930: Theorem 30



Theorem 30 (Distributive Law):

$$x(y+z) = xy + xz.$$

Preliminary Remark: The formula

$$(y+z)x = yx + zx$$

which results from Theorem 30 and Theorem 29, and similar analogues later on, need not be specifically formulated as theorems, nor even be set down.

Proof: Fix x and y, and let $\mathfrak{M}$ be the set of all z for which the assertion holds true.

I) $$x(y+1) = xy' = xy + x = xy + x \cdot 1;$$

1 belongs to $\mathfrak{M}$.

II) If z belongs to $\mathfrak{M}$, then

$$x(y+z) = xy + xz,$$

hence

$$\begin{aligned} x(y+z') &= x((y+z)') = x(y+z)+x = (xy+xz)+x \\ &= xy+(xz+x) = xy+xz', \end{aligned}$$

so that z' belongs to $\mathfrak{M}$.

Therefore, the assertion always holds.

Theorem 30: For all x, y, z, $x * (y + z) = (x * y) + (x * z)$.
Proof:
Fix x, y. $x * (y + 1) = x * y' = x * y + x = (x * y) + (x * 1)$.
Now suppose $x * (y + z) = (x * y) + (x * z)$. Then $x * (y + z') = x * ((y + z)') = (x * (y + z)) + x = ((x * y) + (x * z)) + x = (x * y) + ((x * z) + x) = (x * y) + (x * z')$.
Thus by induction, for all z $x * (y + z) = (x * y) + (x * z)$. Qed.

Components: input

- LaTeX-like input language: use the mathematical typesetting facilities of TeX / LaTeX; or a mathematical XML-format; or the mathematical WYSIWYG editor TeXmacs

- input corresponds to typeset mathematical texts

- web interface under `www.naproche.net`

Components: linguistic analysis

- standard analysis by a Prolog Definite Clause Grammar (DCG), the grammar defines a controlled natural language for mathematics (CNL), i.e. a formal subset of the common mathematical language

- translation into a formal semantics (without ambiguity)

- formal semantics: proof representation structures (PRS), extending discourse representation structures (DRS)

- DRS: tool for anaphor resolution (Let $\underline{x}$ be a set. It is ...) and for interpretation of natural language quantification (Every prime number is positive; a prime number is positive)

- PRS, moreover, represent global text structurings: Theorem / Proof, introductions and retractions of assumptions

Proof representation structures

```
> Every man dances.

 ___________________________________
|                                   |
|___________________________________|
|  ________         ______________  |
| | x1     |       | x2           | |
| |________|       |______________| |
| | man(x1)|  ==>  | dance(x2)    | |
| |________|       | agent(x2,x1) | |
|                  | event(x2)    | |
|                  |______________| |
|___________________________________|
```

Proof representation structures

```
> A square number is positive.

  ________________________________________
 |                                        |
 |________________________________________|
 |   ____________        ______________   |
 |  | x1         |      |              |  |
 |  |____________|      |______________|  |
 |  | square(x1) | ==>  | positive(x1) |  |
 |  |____________|      |______________|  |
 |                                        |
 |                                        |
 |________________________________________|
```

Components: Checking for logical correctness

- translating the PRS conditions into some first-order format
- use TPTP-format (Thousands of Problems for Theorem Provers)
- generate relevant premises for every condition
- automatic theorem prover used to prove every condition from its relevant premises
- proof is accepted if ATP can prove every condition
- feedback of success/error messages

Results

- Naproche input language allows natural reformulation of (simple) mathematical texts
- some example texts and parts of Landau, Foundations of Analysis have been reformulated and checked

Possible applications

- Natural language interfaces to formal mathematics
- Mathematical authoring and checking tools
- writing texts that are simultaneously acceptable by human readers and formal mathematics systems (‘‘Logic for men and machines’’)
- Tutorial applications: teaching how to prove

Technical issues

– inclusion of more mathematical argumentation patterns into Naproche while preserving the unambiguity of the language

– resolving frequent ambiguities: assumptions are explicitely introduced but often implicitely retracted

– including implicit background knowledge on numbers and sets into the system

– improving the ‘‘naturality’’ of the Naproche system (background axioms, interface, linguistic richness, logical richness, strength of automatic theorem provers)

General issues

– Naproche interprets a mathematical proof as a collection of indicators for the construction of a fully formal proof

– natural language components are not just syntactic sugar but serve as indicators

– can/should one model the cognitive processes that take part in the understanding of natural proofs?

– there are natural(ly looking) proofs that are fully formal with respect to the Naproche system

– this defines a ‘‘fortified formalism’’, using linguistic methods and computer implementations, which allows to view some natural proofs as fully formal

– can a ‘‘fortified formalism’’ help to mediate between the ‘‘two streams’’ in the philosophy of mathematics (formalistic / naturalistic)

Thank You!