# Formal Mathematics and Controlled Natural Language

Peter Koepke, University of Bonn, Germany

Mathematical Institute

Oberseminar Diskrete Optimierung

Bonn, February 14, 2011

## The Gödel completeness theorem

*Über die Vollständigkeit des Logikkalküls (1929)*

1. Einleitung

Der Hauptgegenstand der folgenden Untersuchungen ist der Beweis der Vollständigkeit des in Russell, *Principia mathematica* [...] und ähnlich in Hilbert-Ackermann, *Grundzüge der theoretischen Logik* [...] angegebenen Axiomensystems des sogenannten engeren Funktionenkalküls. Dabei soll "Vollständigkeit" bedeuten, daß jede im engeren Funktionenkalkül ausdrückbare allgemein giltige Formel [...] sich durch eine endliche Reihe formaler Schlüsse aus den Axiomen deduzieren lässt.

(Kurt Gödel, Doctoral Dissertation, Vienna 1929)

## First order predicate logic

$$\mathrm{Var} ::= v_0|v_1|v_2|\ldots|x\,|\,y\,|\,z\,|\ldots$$

$$\mathrm{Func} ::= \ldots$$

$$\mathrm{Term} ::= \mathrm{Var}|\mathrm{Func}(\mathrm{Var},\ldots,\mathrm{Var})$$

$$\mathrm{Rel} ::= \ldots$$

$$\mathrm{AtomForm} ::= \mathrm{Rel}(\mathrm{Term},\ldots,\mathrm{Term})\,|\,\mathrm{Term} \equiv \mathrm{Term}$$

$$\mathrm{Form} ::= \mathrm{AtomForm}\,|\,\mathrm{Form} \rightarrow \mathrm{Form}\,|\,\bot\,|\,\forall\,\mathrm{Var}\,\mathrm{Form}$$

## A complete first order calculus

$$\frac{\Gamma \qquad \varphi}{\Gamma \quad \psi \quad \varphi}\ ;\ \frac{}{\Gamma \quad \varphi \quad \varphi}\ ;\ \frac{\Gamma \quad \varphi \quad \psi}{\Gamma \qquad \varphi \to \psi}\ ;\ \frac{\begin{array}{l}\Gamma \quad \varphi \\ \Gamma \quad \varphi \to \psi\end{array}}{\Gamma \quad \psi}\ ;\ \frac{\begin{array}{l}\Gamma \quad \varphi \\ \Gamma \quad \neg\varphi\end{array}}{\Gamma \quad \perp}\ ;$$

$$\frac{\Gamma \quad \neg\varphi \quad \perp}{\Gamma \qquad \varphi}\ ;\ \frac{\Gamma \quad \varphi\frac{y}{x}}{\Gamma \quad \forall x\varphi}\ ,\ \text{if } y \notin \text{free}(\Gamma \cup \{\forall x\varphi\});\ \frac{\Gamma \quad \forall x\varphi}{\Gamma \quad \varphi\frac{t}{x}}\ ;$$

$$\frac{}{\Gamma \quad t \equiv t}\ ;\ \frac{\begin{array}{l}\Gamma \quad \varphi\frac{t}{x} \\ \Gamma \quad t \equiv t'\end{array}}{\Gamma \quad \varphi\frac{t'}{x}}$$

## Formalizing mathematics in set theory

- $\mathbb{N}$: $0 = \emptyset, 1 = \{0\}, 2 = \{0, 1\}, \ldots, n + 1 = \{0, \ldots n\}, \ldots$
- $\mathbb{Q}$: $q = \frac{m}{n} = (m, n) = \{\{m\}, \{m, n\}\}$
- $\mathbb{R}$: $r \subset \mathbb{Q}$ (left half of a Dedekind cut)
- geometric space $\mathbb{R}^n$: $p = (p_0, \ldots, p_{n-1})$
- geometric objects: $M \subseteq \mathbb{R}^n$
- relations and functions as sets of tupels
- abstract topological spaces: $(X, T)$ where $T \subseteq \mathcal{P}(X)$
- ...

## Set theory has a first order formalization

- Gottlob Frege, *Begriffsschrift*, *Grundgesetze der Arithmetic*
- Ernst Zermelo 1908
- Abraham Fraenkel ~1920
- Thoralf Skolem 1929
- Zermelo-Fraenkel set theory (ZF or ZFC)

## Zermelo-Fraenkel axioms in first order logic

- Extensionality: $\forall x \forall y (\forall z (z \in x \leftrightarrow z \in y) \rightarrow x \equiv y)$

- Zermelo's Aussonderungs schema:

$$\forall x_1 \ldots \forall x_n \forall x \exists y \forall z (z \in y \leftrightarrow z \in x \wedge \varphi(z, x_1, \ldots, x_n))$$

- Infinity: $\exists x (\exists y (y \in x \wedge \forall z \neg z \in y) \wedge \forall y (y \in x \rightarrow \exists z (z \in x \wedge \forall w (w \in z \leftrightarrow w \in y \vee w \equiv y))))$

- ...

## What is mathematics?

- mathematics = set theory
- mathematics = first order logic + ZFC
- mathematical proofs = formal derivations from ZFC

## Formal mathematics

Every logically true mathematical statement has a formal derivation.

Every true mathematical statement has a formal derivation within some (foundational) axiom system.

Every mathematical proof can be replaced by a formal derivation.

Mathematics can be in principle be carried out completely formal (Formal mathematics).

| | | | | | |
|---|---|---|---|---|---|
| 1. | $\Phi_{\mathrm{Gr}}$ | $\neg \circ v_0\, e \equiv v_0$ | | $\neg\exists v_0 \neg \circ v_0\, e \equiv v_0$ | VR |
| 2. | $\Phi_{\mathrm{Gr}}$ | $\neg \circ v_0\, e \equiv v_0$ | | $\neg \circ v_0\, e \equiv v_0$ | VR |
| 3. | $\Phi_{\mathrm{Gr}}$ | $\neg \circ v_0\, e \equiv v_0$ | | $\exists v_0 \neg \circ v_0\, e \equiv v_0$ | $\exists$S auf 2 |
| 4. | $\Phi_{\mathrm{Gr}}$ | | | $\circ v_0\, e \equiv v_0$ | WR auf 1,3 |
| 5. | | | | $(v_2 \equiv \circ v_0\, e)\dfrac{\circ v_0\, e}{v_2}$ | $(\equiv)$ |
| 6. | | $\circ v_0\, e \equiv v_0$ | | $(v_2 \equiv \circ v_0\, e)\dfrac{v_0}{v_2}$ | Sub auf 5 |
| 7. | $\Phi_{\mathrm{Gr}}$ | $\circ v_0\, e \equiv v_0$ | | $v_0 \equiv \circ v_0\, e$ | AR auf 6 |
| 8. | $\Phi_{\mathrm{Gr}}$ | | | $v_0 \equiv \circ v_0\, e$ | KS auf 4,7 |
| 9. | $\Phi_{\mathrm{Gr}}$ | $v_0 \equiv e$ | | $v_0 \equiv e$ | VR |
| 10. | $\Phi_{\mathrm{Gr}}$ | $v_0 \equiv e$ | | $(\neg \circ v_0\, e \equiv e \vee v_0 \equiv e)$ | $\vee$S auf 9 |
| 11. | $\Phi_{\mathrm{Gr}}$ | $\neg v_0 \equiv e$ | | $(\neg v_2 \equiv e)\dfrac{v_0}{v_2}$ | VR |
| 12. | $\Phi_{\mathrm{Gr}}$ | $\neg v_0 \equiv e$ | $v_0 \equiv \circ v_0\, e$ | $(\neg v_2 \equiv e)\dfrac{\circ v_0\, e}{v_2}$ | Sub auf 11 |
| 13. | $\Phi_{\mathrm{Gr}}$ | $\neg v_0 \equiv e$ | $v_0 \equiv \circ v_0\, e$ | $\neg \circ v_0\, e \equiv e$ | 12 |
| 14. | $\Phi_{\mathrm{Gr}}$ | $\neg v_0 \equiv e$ | | $v_0 \equiv \circ v_0\, e$ | AR auf 8 |
| 15. | $\Phi_{\mathrm{Gr}}$ | $\neg v_0 \equiv e$ | | $\neg \circ v_0\, e \equiv e$ | KS auf 14 |
| 16. | $\Phi_{\mathrm{Gr}}$ | $\neg v_0 \equiv e$ | | $(\neg \circ v_0\, e \equiv e \vee v_0 \equiv e)$ | $\vee$S auf 15 |
| 17. | $\Phi_{\mathrm{Gr}}$ | | | $(\neg \circ v_0\, e \equiv e \vee v_0 \equiv e)$ | FU auf 10,16 |
| 18. | $\Phi_{\mathrm{Gr}}$ | $\neg(\neg \circ v_0\, e \equiv e \vee v_0 \equiv e)$ | $\neg\neg\exists v_0 \neg(\neg \circ v_0\, e \equiv e \vee v_0 \equiv e)$ | $(\neg \circ v_0\, e \equiv e \vee v_0 \equiv e)$ | AR auf 17 |
| 19. | $\Phi_{\mathrm{Gr}}$ | $\neg(\neg \circ v_0\, e \equiv e \vee v_0 \equiv e)$ | $\neg\neg\exists v_0 \neg(\neg \circ v_0\, e \equiv e \vee v_0 \equiv e)$ | $\neg(\neg \circ v_0\, e \equiv e \vee v_0 \equiv e)$ | VR |
| 20. | $\Phi_{\mathrm{Gr}}$ | $\neg(\neg \circ v_0\, e \equiv e \vee v_0 \equiv e)$ | | $\neg\exists v_0 \neg(\neg \circ v_0\, e \equiv e \vee v_0 \equiv e)$ | WR auf 18,19 |
| 21. | $\Phi_{\mathrm{Gr}}$ | $\exists v_0 \neg(\neg \circ v_0\, e \equiv e \vee v_0 \equiv e)$ | | $\neg\exists v_0 \neg(\neg \circ v_0\, e \equiv e \vee v_0 \equiv e)$ | $\exists$A auf 20 |
| 22. | $\Phi_{\mathrm{Gr}}$ | $\neg\exists v_0 \neg(\neg \circ v_0\, e \equiv e \vee v_0 \equiv e)$ | | $\neg\exists v_0 \neg(\neg \circ v_0\, e \equiv e \vee v_0 \equiv e)$ | VR |
| 23. | $\Phi_{\mathrm{Gr}}$ | | | $\neg\exists v_0 \neg(\neg \circ v_0\, e \equiv e \vee v_0 \equiv e)$ | FU auf 21,22 |

## Formal derivations can be checked and produced automatically

- derivations are formed by repeated applications of (simple) syntactic rules

- whether a formal text is a derivation can (easily) be checked algorithmically

## Formal proofs - derivations

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, [...]

## Computer-supported formal proofs

J. McCarthy:

Checking mathematical proofs is potentially one of the most interesting and useful applications of automatic computers. ... 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.

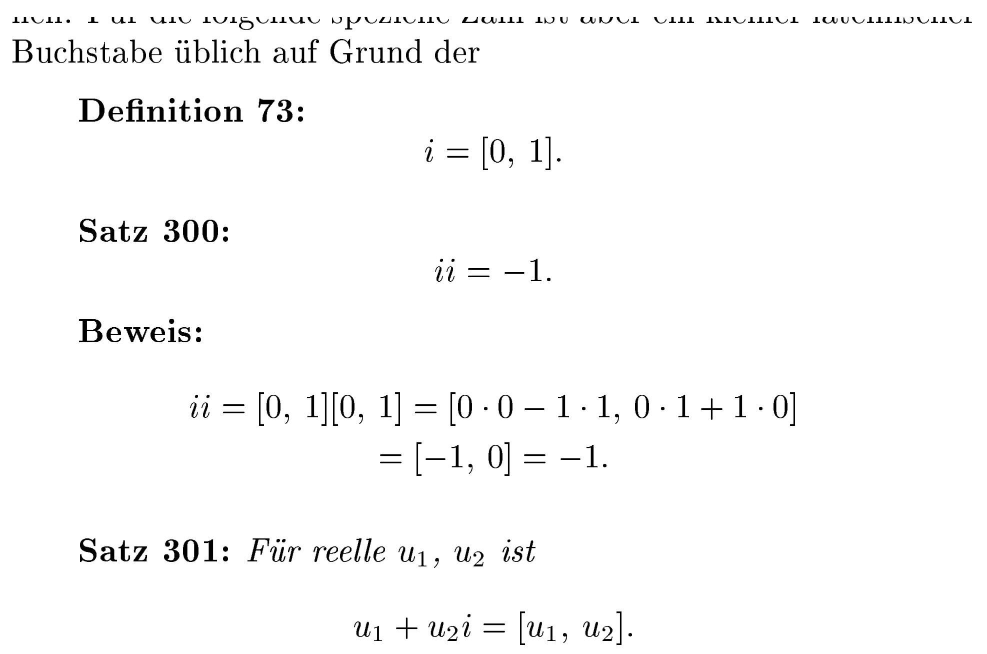
McCarthy, J. "Computer Programs for Checking Mathematical Proofs," Proceedings of the Symposium in Pure Math, Recursive Function Theory,Volume V, pages 219-228, AMS, Providence, RI, 1962.

## Automatic proof checking

*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"(0,1rl),
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"(0,0),
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,0),
ts01(0,1rl,refis(real,0))),ts02(1rl,0,refis(real,0))):is"r"(pl"r"(ts"r"(0,1rl),ts"r"(1rl,0)),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

www.mizar.org

## **MIZAR example**: Proof of the Gödel completeness theorem by Patrick Braselmann and PK

```
theorem
  still_not-bound_in X is finite & X |= p implies X |- p
proof
  assume
A1: still_not-bound_in X is finite;
  assume
A2: X |= p;
  assume
A3: not X |- p;
  reconsider Y = X \/ {'not' p} as Subset of CQC-WFF;
A4: still_not-bound_in Y is finite by A1,Th36;
  Y is Consistent by A3,HENMODEL:9;
  then ex CZ,JH1 st ( JH1,valH |= Y) by A4,Th34;
  hence contradiction by A2,Th37;
end;
```

## Formal mathematics systems

- proof checking $\leftrightarrow$ automatic proving
- Classical logic $\leftrightarrow$ non-classical, constructive, intuitionistic logic
- general purpose $\leftrightarrow$ specialized
- natural deduction style $\leftrightarrow$ resolution/unification style
- readability $\leftrightarrow$ machine orientated

## Provers of the world

Of “The Hundred Greatest Theorems” list, there are formalizations in the following systems (see Freek Wiedijk):

– 76 in HOL Light (higher order logic, John Harrison)

– 51 in Mizar (classical)

– 49 in Coq (type theory, calculus of inductive definitions

– 46 in Isabelle (weak type theory, various logics)

– 42 in ProofPower

## Some “big” formalizations

– Four Colour Theorem (Georges Gonthier, Coq)

– Prime Number Theorem, “elementary” proof (Jeremy Avigad, Isabelle)

– Prime Number Theorem, analytic proof (John Harrison, HOL Light)

– work in progress: Flyspeck Project (Formal Proof of the Kepler Conjecture) (Tom Hales, various systems)

– Jordan Curve Theorem (Tom Hales, HOL Light)

## Some ‘‘big’’ formalizations

– Correctness of arithmetical algorithms like division, square root, transcendental functions (hardware and software)

– in particular floating point arithmetic

– Correctness of (RISC) microprocessors

– ...

– Software for driverless Paris Metro Line 14

## Natural proofs

- directed at human readers
- use human notions, intuitions, argumentations
- use natural language
- refer to other human proofs
- have to be compact, surveyable (Ludwig Wittgenstein)
- have a certain granularity, leaving out details and implicit knowledge

## Can formal proofs be made more natural?

## The Naproche project: Natural language proof checking

– studies the syntax and semantics of the language of proofs, emphasizing natural language and natural argumentation aspects, also in relation to formal mathematics

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

– joint work with Bernhard Schröder, linguistics; Bonn, Essen, Cologne; www.naproche.net

– development of a mathematical authoring system with a LATEX-quality graphical interface

## The Naproche system

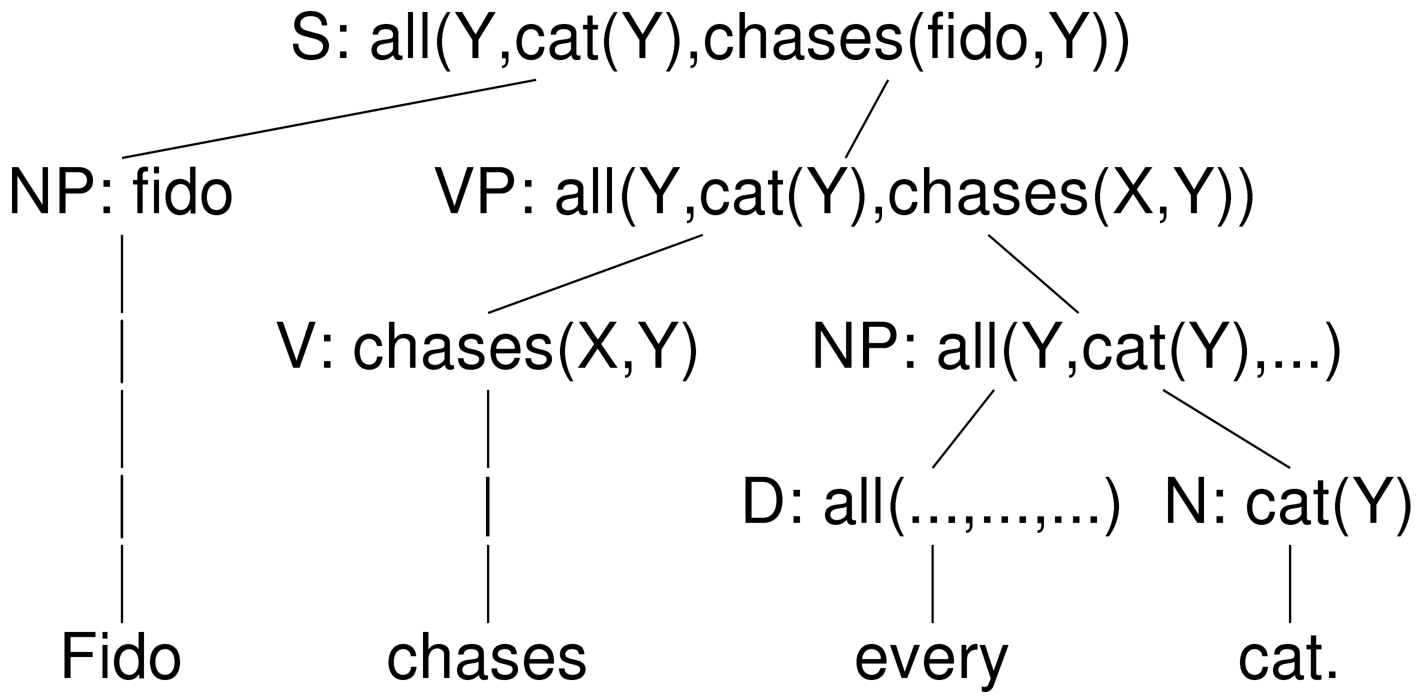
– To devise a strictly formal system for mathematics, implemented by computer, whose input language is an extensive part of the common mathematical language, and whose proof style is close to proof styles found in the mathematical literature.

## Mathematical statements

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

## Linguistic analysis of sentences

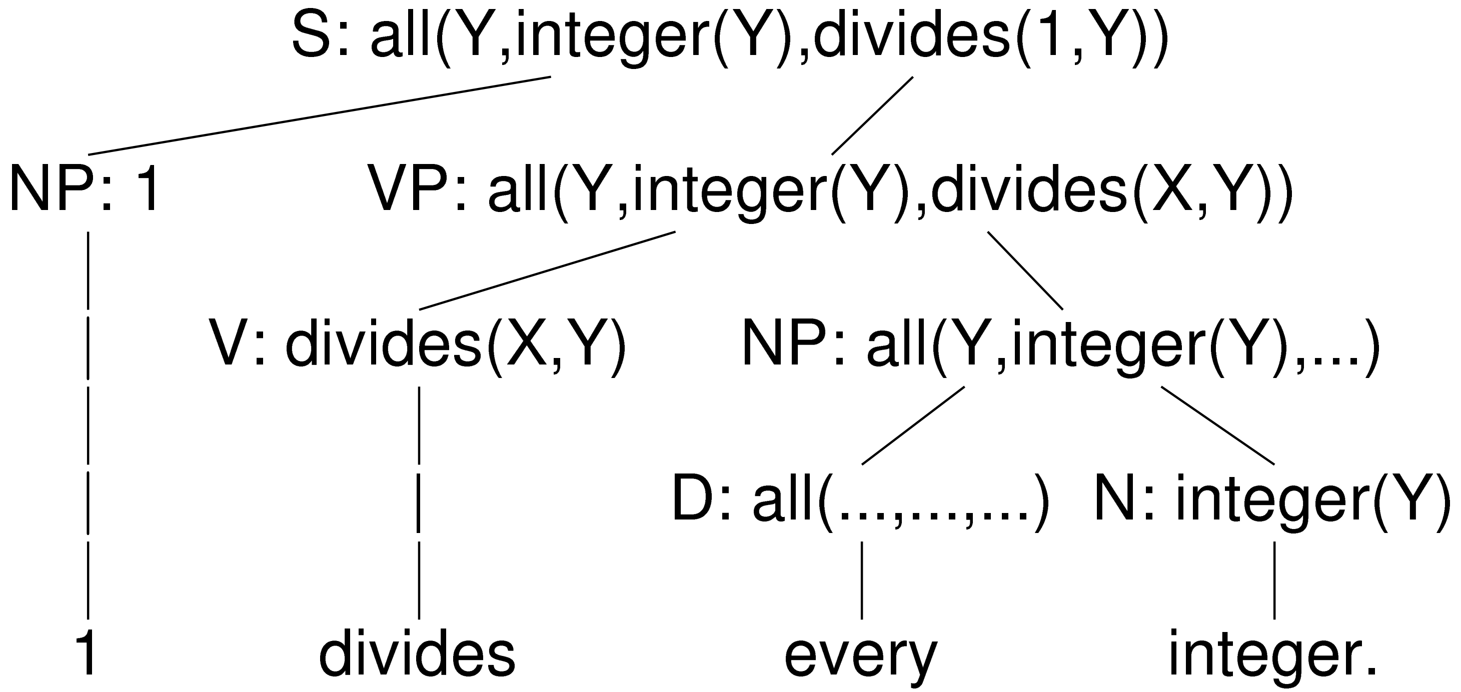
“Fido chases every cat.”



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

## Linguistic analysis of sentences

“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.

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

## Linguistic analysis of sentences

– Formal grammars, e.g., Phrase Structure Grammar

– Standard techniques of computational linguistics like tokenizing

– (Parsing mathematical notation like $\sum_{n=1}^{\infty}\frac{1}{n^2}$ and combining with the natural language parsing?)

– Less ambiguities in natural mathematical language than in general natural language: “a man loves a woman” versus “a negative number is smaller than a positive number”; in case of ambiguity a mathematician would explicitely write the quantifiers.

## Mathematical texts

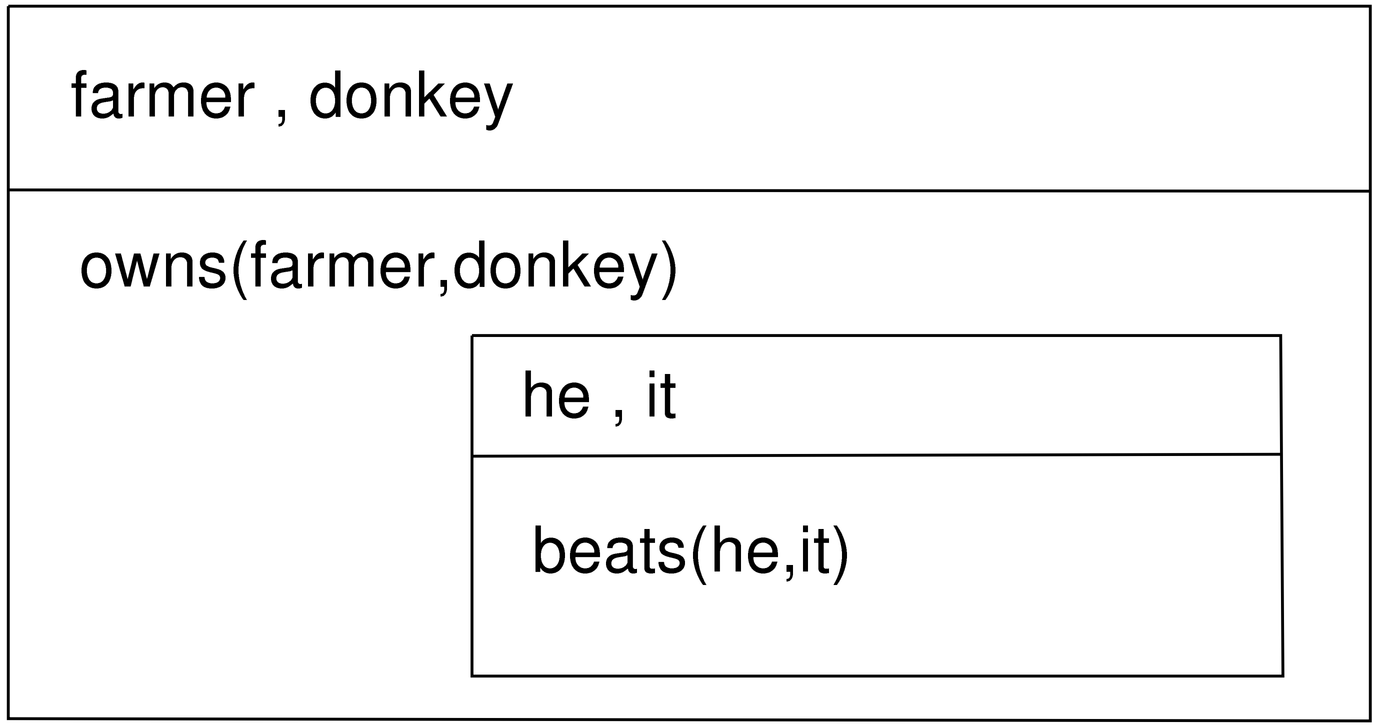
$$\begin{tikzcd}[column sep=small] \varphi. & \text{Then} & \psi. \arrow[ll, bend left] \end{tikzcd}$$

$$\begin{tikzcd}[column sep=small] \text{A} & \text{farmer} & \text{owns a} & \text{donkey.} & \text{He} \arrow[lll, bend left] & \text{beats} & \text{it.} \arrow[lll, bend left] \end{tikzcd}$$

Logical references, premises $\leftrightarrow$ pronouns / nouns

## Linguistic analysis of texts

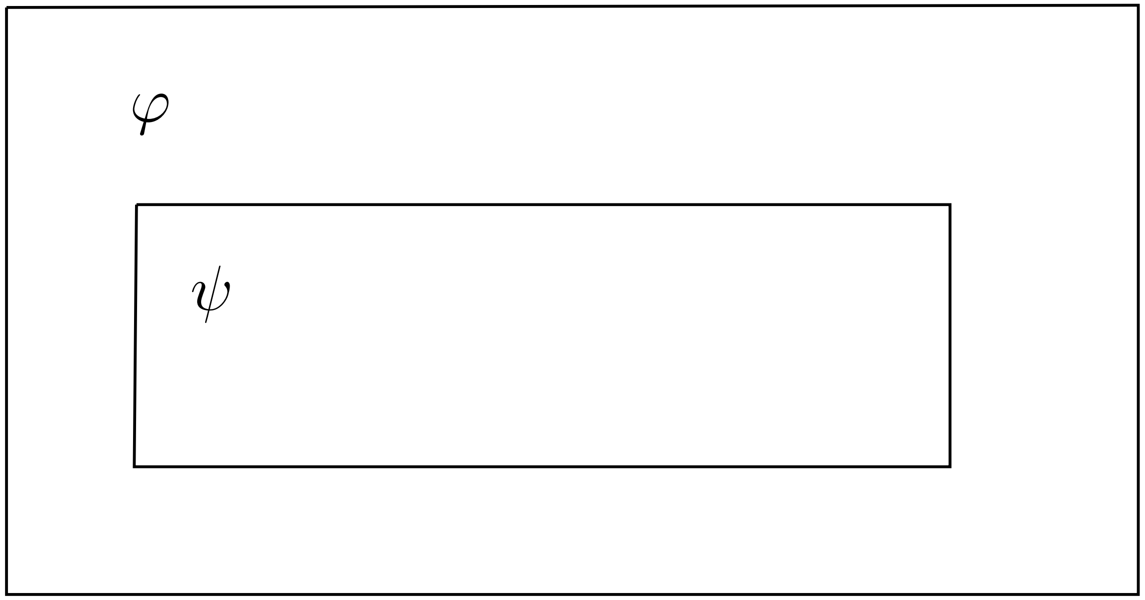
Discourse representation theory (Hans Kamp)



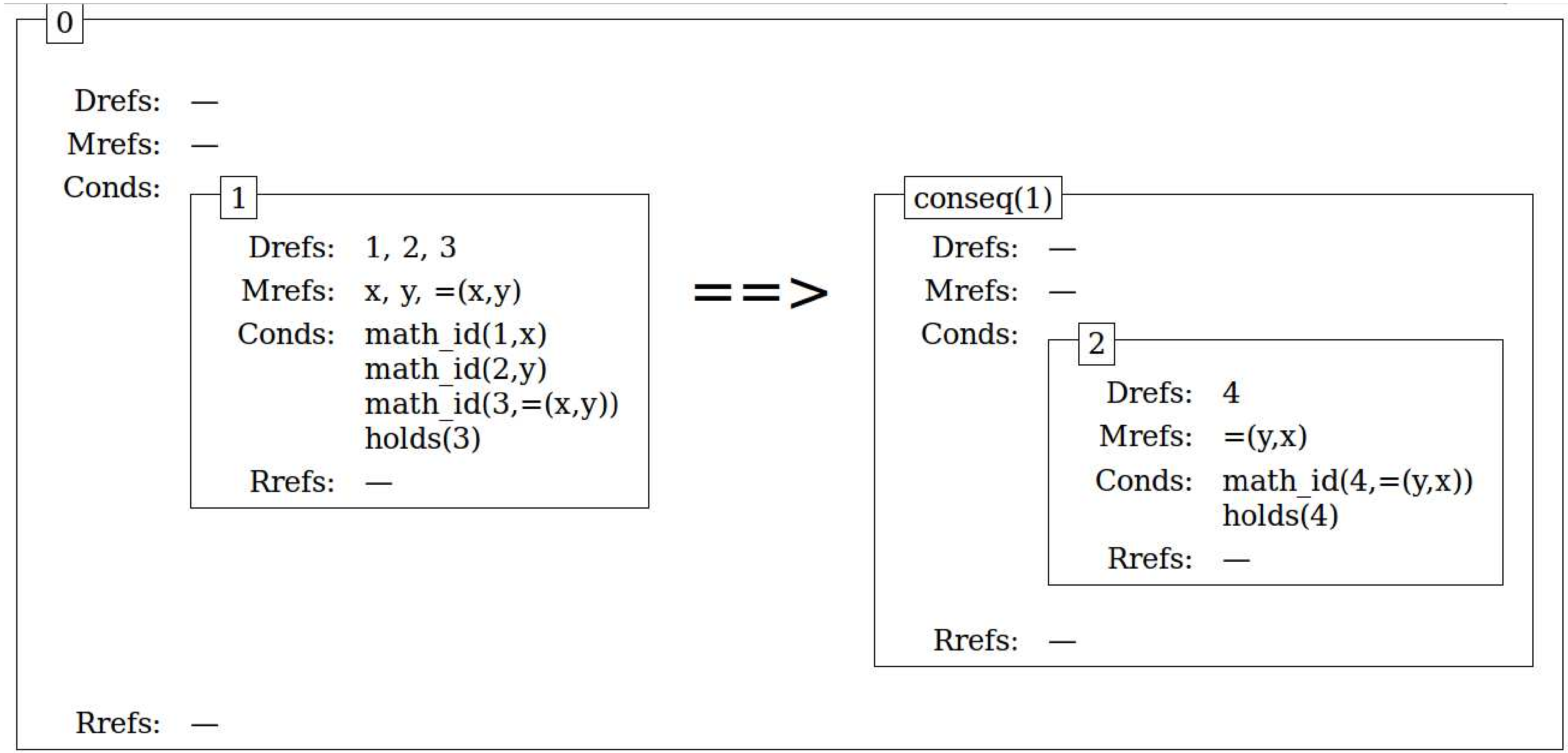
$$\exists f\,, d(\mathrm{owns}(f\,, d) \wedge \mathrm{beats}(f\,, d))$$

## Linguistic analysis of texts

Natural deduction (Lukasievicz, Gentzen) has a similar box structure



## Proof representation structures



## Natural proofs and natural argumentation

- (what is a proof?)
- natural proofs are mathematical argumentations
- techniques from the linguistics of argumentation may be used; argumentations are sometimes analyzed by formal logical tools
- Proof Representation Structures can be translated into input for formal mathematics

## Layers of the Naproche system

$\downarrow$ Standard editor or web editor

TeX-style input text

$\updownarrow$ Natural language processing (NLP)

Proof representation structure (PRS)

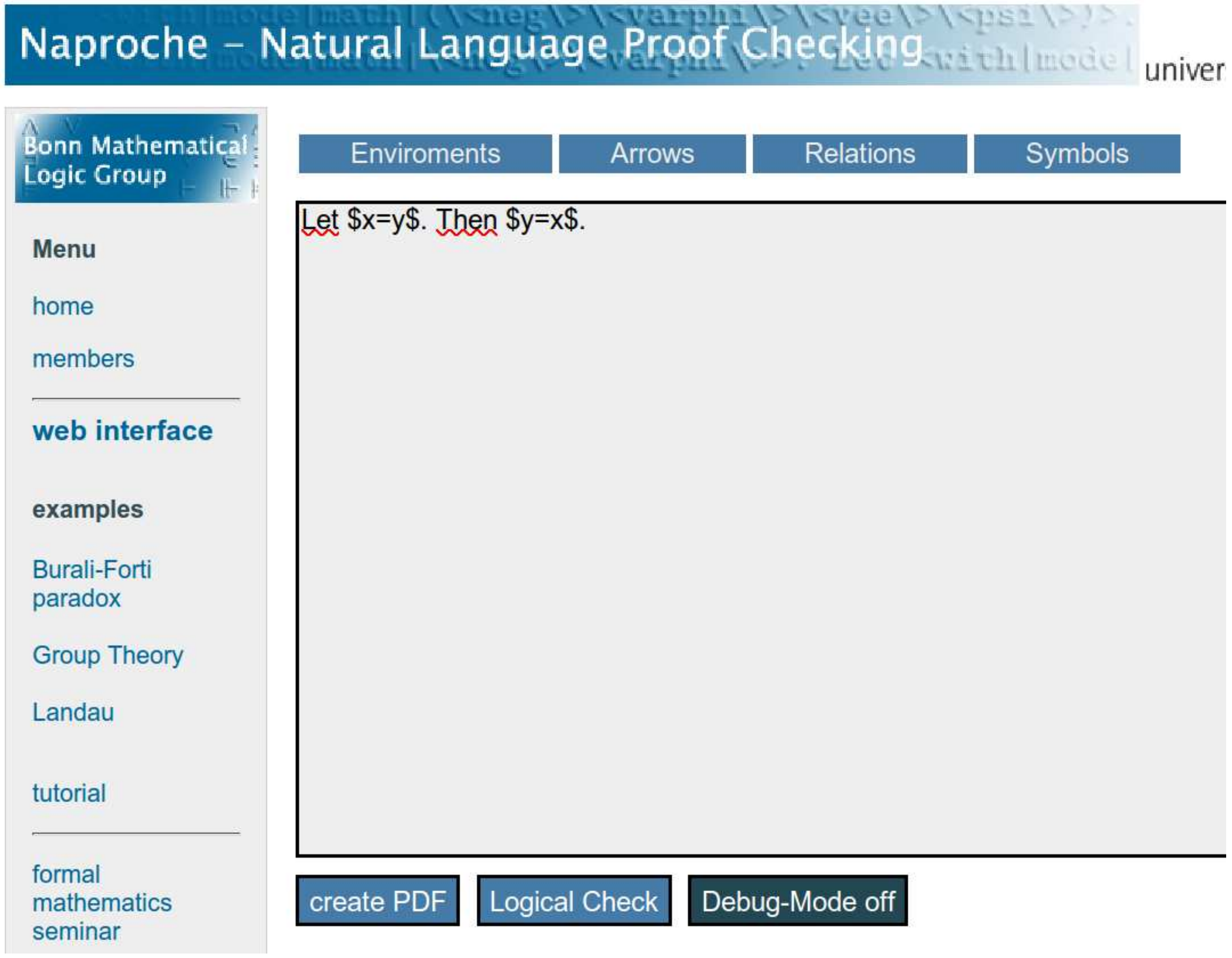
$\updownarrow$ First-order translation

First-order logic format (TPTP)

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

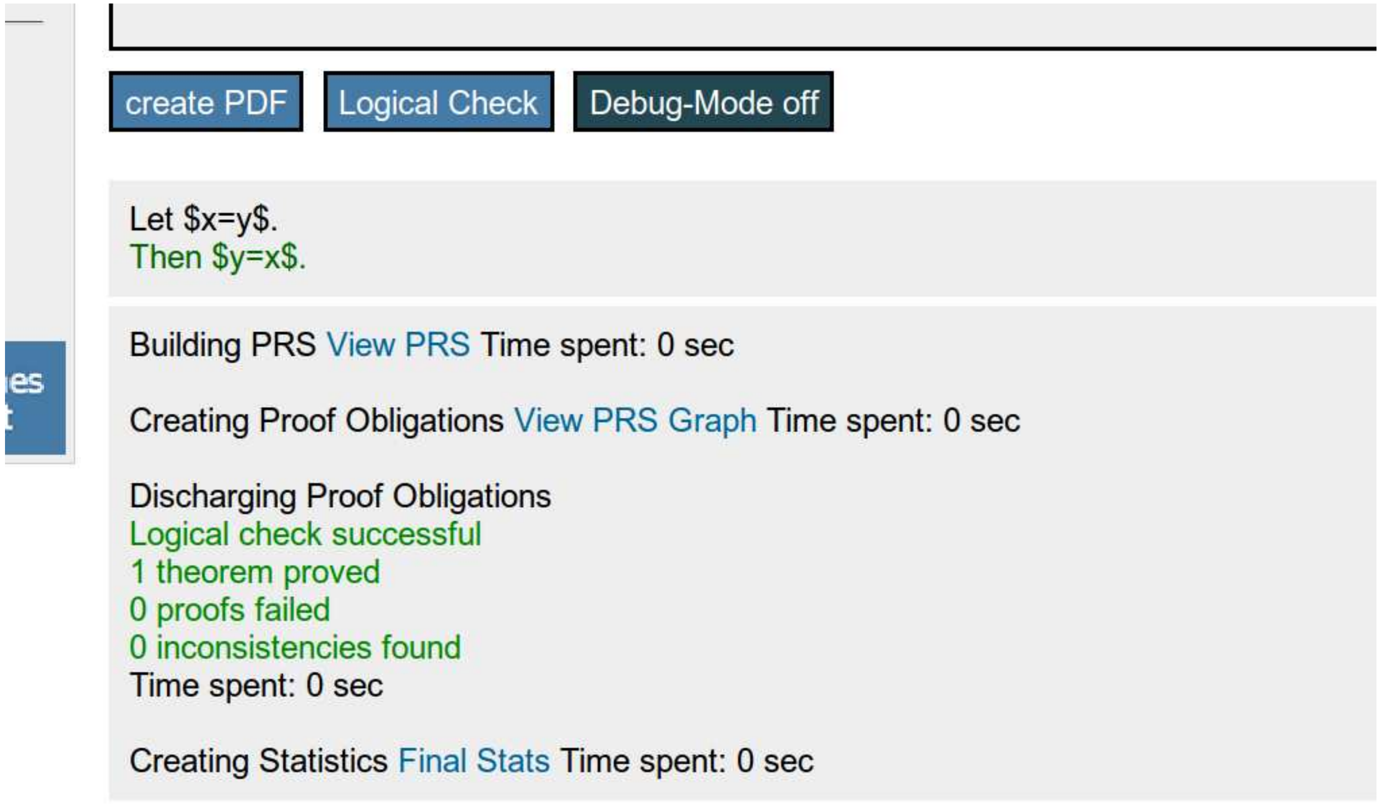
“Accepted”/“Not accepted”, with error messages

## The Naproche system:



Let $x=y$. Then $y=x$.

# The Naproche system



```
Let $x=y$.
Then $y=x$.
```

```
Building PRS View PRS Time spent: 0 sec

Creating Proof Obligations View PRS Graph Time spent: 0 sec

Discharging Proof Obligations
Logical check successful
1 theorem proved
0 proofs failed
0 inconsistencies found
Time spent: 0 sec

Creating Statistics Final Stats Time spent: 0 sec
```

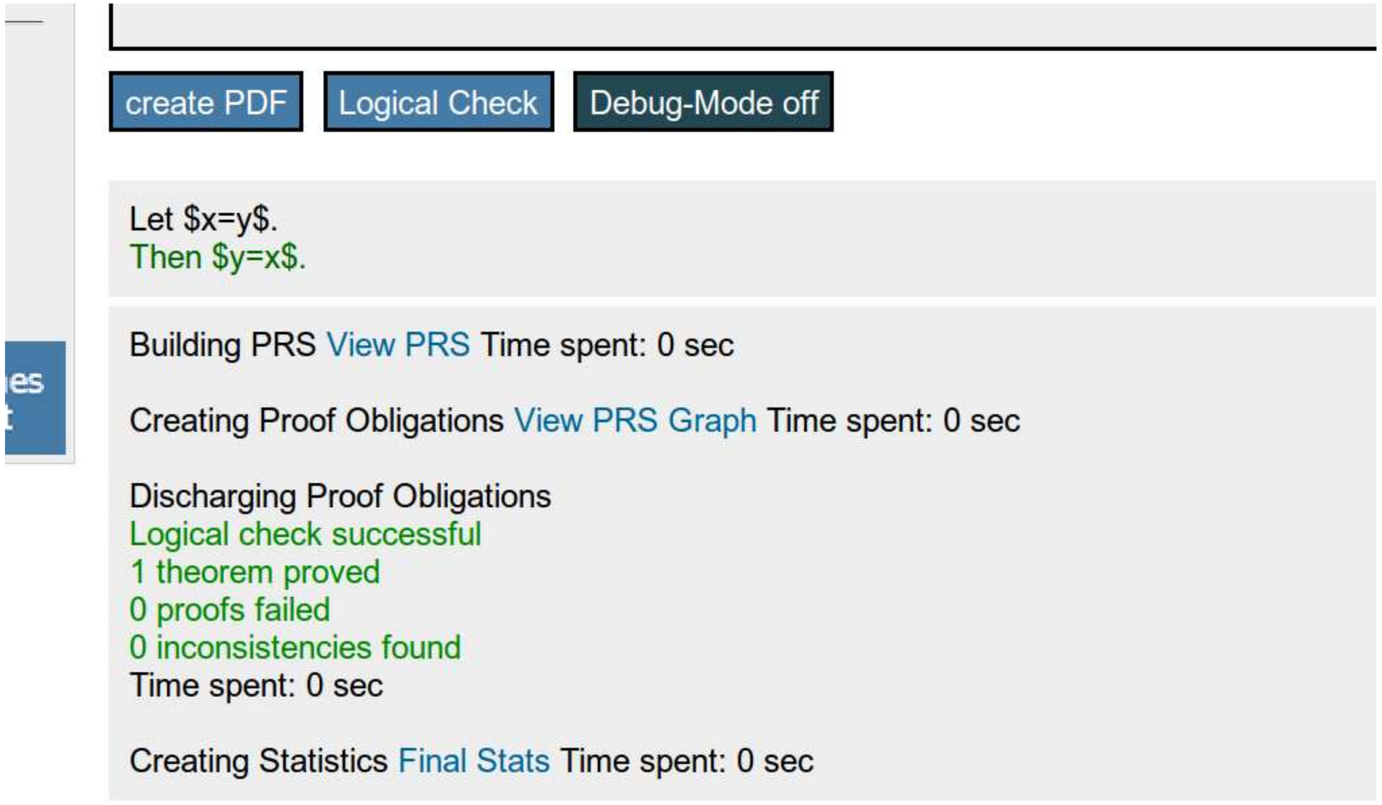
## The Naproche system

Proof obligation for $y = x$:

fof('holds(2, 4, 0)', conjecture, vd2 = vd1).

fof('holds(1, 3, 0)', axiom, vd1 = vd2).

# The Naproche system



Let \$x=y\$.
Then \$y=x\$.

Building PRS View PRS Time spent: 0 sec

Creating Proof Obligations View PRS Graph Time spent: 0 sec

Discharging Proof Obligations
Logical check successful
1 theorem proved
0 proofs failed
0 inconsistencies found
Time spent: 0 sec

Creating Statistics Final Stats Time spent: 0 sec

## The Naproche system

```
Axiom 1.
For all $x$, $y$, $z$, $(x*y)*z=x*(y*z)$.

Axiom 2.
For all $x$, $1*x=x$ and $x*1=x$.

Axiom 3.
For all $x$, $x*f(x)=1$ and $f(x)*x=1$.

Lemma 1.
If $u*x=x$ then $u=1$.

Proof.
Suppose that $u*x=x$.
Then $(u*x)*f(x)=x*f(x)$. By axiom 1, $u*(x*f(x))=x*f(x)$. So by axiom 3 $u*1=1$.
Then $u=1$ by axiom 2. Qed.
```

## The Naproche system

```
Lemma 2.
If $x*y=1$ then $y=f(x)$.

Proof.
Assume $x*y=1$.
Then $f(x)*(x*y)=f(x)*1$, i.e. $(f(x)*x)*y=f(x)$. Hence $1*y=f(x)$, i.e. $y=f(x)$.
Qed.

Theorem 1.
$f(x*y)=f(y)*f(x)$.

Proof.
Let $u=(x*y)*(f(y)*f(x))$.
Then $u=x*((y*f(y))*f(x))$ by axiom 1. So $u=x*(1*f(x))=x*f(x)=1$.
Thus $(x*y)*(f(y)*f(x))=1$. Hence $(f(y)*f(x))=f(x*y)$ by lemma 2.
Qed.
```

## The Naproche system

Axiom 1. For all $x$, $y$, $z$, $(x * y) * z = x * (y * z)$.

Axiom 2. For all $x$, $1 * x = x$ and $x * 1 = x$.

Axiom 3. For all $x$, $x * f(x) = 1$ and $f(x) * x = 1$.

Lemma 1. If $u * x = x$ then $u = 1$.

Proof. Suppose that $u * x = x$. Then $(u * x) * f(x) = x * f(x)$. By axiom 1, $u * (x * f(x)) = x * f(x)$. So by axiom 3 $u * 1 = 1$. Then $u = 1$ by axiom 2. Qed.

Lemma 2. If $x * y = 1$ then $y = f(x)$.

Proof. Assume $x * y = 1$. Then $f(x) * (x * y) = f(x) * 1$, i.e. $(f(x) * x) * y = f(x)$. Hence $1 * y = f(x)$, i.e. $y = f(x)$. Qed.

Theorem 1. $f(x * y) = f(y) * f(x)$.

Proof. Let $u = (x * y) * (f(y) * f(x))$. Then $u = x * ((y * f(y)) * f(x))$ by axiom 1. So $u = x * (1 * f(x)) = x * f(x) = 1$. Thus $(x * y) * (f(y) * f(x)) = 1$. Hence $(f(y) * f(x)) = f(x * y)$ by lemma 2. Qed.

## The Naproche system

Building PRS View PRS Time spent: 4 sec

Creating Proof Obligations View PRS Graph Time spent: 0 sec

Discharging Proof Obligations Logical check successful

17 theorems proved

0 proofs failed

0 inconsistencies found

Time spent: 3 sec

Creating Statistics Final Stats Time spent: 0 sec

## Components of the Naproche system: 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)

## Components of the Naproche system: linguistic analysis

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

– DRS: tool for anaphora 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

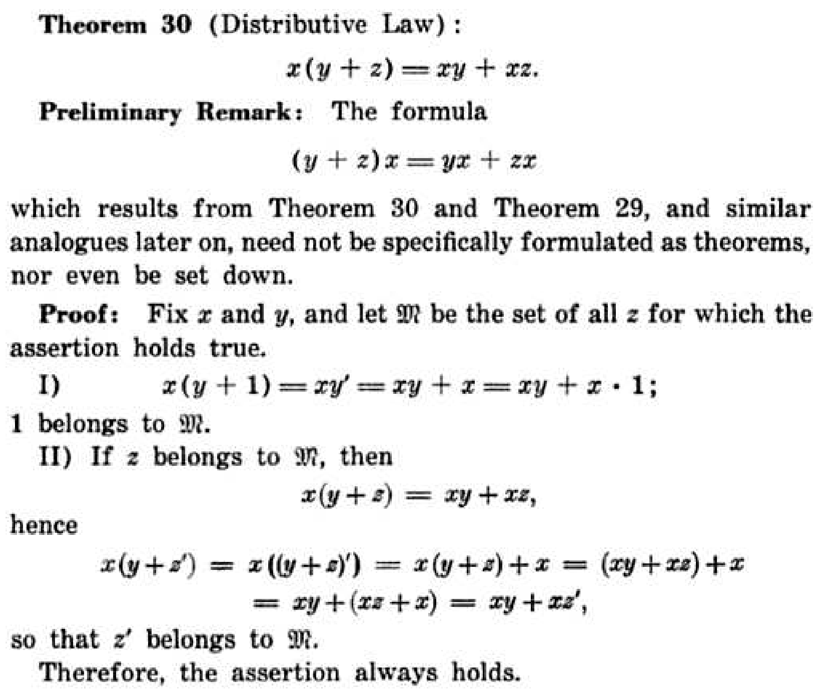
## Components of the Naproche system: Checking logical correctness

- translating the PRS conditions into the first-order format TPTP (Thousands of Problems for Theorem Provers)
- generate relevant premises for every condition
- automatic theorem prover (ATP) used to prove every condition from its relevant premises; strength of the ATP may allow to bridge “gaps” in the proof
- proof is accepted if ATP can prove every condition

## Automatic theorem provers (ATPs)

- First order theorem provers, usully based on resolution, superposition, normal forms and code-optimization
- Examples: Otter, SPASS, Vampire, ...
- Development driven by yearly competitions (CASC = CADE ATP System Competition)

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

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.

# Chapter 1 from Landau in Naproche

by Merlin Carl, Marcos Cramer, Daniel Khlwein

*February 14, 2011*

**Abstract**

This is a reformulation of the first chapter of Landau's *Grundlagen der Analysis* in the Controlled Natural Language of Naproche. Talk about sets is still avoided. One consequence of this is that Axiom 5 (the induction axiom) cannot be formulated; instead we use an induction proof method.

Axiom 3: For every $x$, $x' \neq 1$.

Axiom 4: If $x' = y'$, then $x = y$.

Theorem 1: If $x \neq y$ then $x' \neq y'$.
Proof:
Assume that $x \neq y$ and $x' = y'$. Then by axiom 4, $x = y$. Qed.

Theorem 2: For all $x$ $x'\neq x$.
Proof:
By axiom 3, $1'\neq 1$. Suppose $x'\neq x$. Then by theorem 1, $(x')'\neq x'$. Thus by induction, for all $x$ $x'\neq x$. Qed.

Theorem 3: If $x\neq 1$ then there is a $u$ such that $x=u'$.
Proof:
If $1\neq 1$ then there is a $u$ such that $1=u'$.
Assume $x'\neq 1$. If $u=x$ then $x'=u'$. So there is a $u$ such that $x'=u'$.
Thus by induction, if $x\neq 1$ then there is a $u$ such that $x=u'$. Qed.

Definition 1:
Define $+$ recursively:
$x+1=x'$.
$x+y'=(x+y)'$.

Theorem 5: For all $x$, $y$, $z$, $(x+y)+z=x+(y+z)$.
Proof:
Fix $x$, $y$.
$(x+y)+1=(x+y)'=x+y'=x+(y+1)$.
Assume that $(x+y)+z=x+(y+z)$. Then $(x+y)+z'=((x+y)+z)'=(x+(y+z))'=x+(y+z)'=x+(y+z')$. So $(x+y)+z'=x+(y+z')$.
Thus by induction, for all $z$, $(x+y)+z=x+(y+z)$. Qed.

Lemma 4a: For all $y$, $1+y=y'$.
Proof:
By definition 1, $1+1=1'$.
Suppose $1+y=y'$. Then by definition 1, $1+y'=(1+y)'$. So $1+y'=(y')'$.
Thus by induction, for all $y$ $1+y=y'$. Qed.

## Current projects

- Formalizing Landau
- Rewriting and updating the Naproche software for greater modularity and more linguistic variants
- Formalizing Euclid’s Elements, book 1?
- Putting a Naproche layer on the formal proof of the Gödel completeness theorem

## 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

## General issues

- Linguistics: construction and analysis of a mathematical language with a definite first order semantics
- Can the gap between natural proofs and formal derivations be narrowed
- Philosophy of mathematics: what is a mathematical proof?
- there are some natural(ly looking) proofs that are fully formal with respect to the Naproche system

**Thank You!**