EuroProofNet

FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

Natty: A Natural-Language Proof Assistant

Adam Dingle

Charles University, Prague

June 5, 2025

Outline

Outline

Vision

- Type mathematics in an editor, Natty will verify it
- Plain text with Unicode symbols
- Real-time feedback
- Controlled vocabulary/grammar, but as natural as possible
- Any mathematical domain

A proof in Natty

Right cancellation of multiplication

Theorem 5. Let a, b, c ∈ ℕ. If c ≠ 0 and ac = bc then a = b.

Proof. Let

$$G = \{ x \in \mathbb{N} \mid \text{for all } y, z \in \mathbb{N}, \text{ if } z \neq 0 \text{ and } xz = yz \text{ then } x = y \}.$$

Let b, c ∈ ℕ with c ≠ 0 and 0 · c = bc. Then bc = 0. Since c ≠ 0, we must have b = 0 by Theorem 4.1. So 0 = b, and hence 0 ∈ G.

Now let a ∈ ℕ, and suppose that a ∈ G. Let b, c ∈ ℕ, and suppose that c ≠ 0 and s(a) · c = bc. Then by Theorem 3.5 we deduce that ca + c = bc. If b = 0, then either s(a) = 0 or c = 0, which is a contradiction. Hence b ≠ 0. By Lemma 1 there is some p ∈ ℕ such that b = s(p). Therefore ca + c = s(p) · c, and we see that ca + c = cp + c. It follows by Theorem 2.1 that ca = cp, so ac = pc. By hypothesis it follows that a = p. Therefore s(a) = s(p) = b. Hence s(a) ∈ G, and we deduce that G = ℕ.

Interactive provers = proof assistants

- The user interacts with the system to build proofs
- In many systems, the user uses tactics to work toward a goal
- Some systems invoke automatic provers to verify steps
- Lots of math has been formalized in these systems
- Proofs are often not easy to read
- Isabelle, HOL Light, Rocq, Lean, ...

Wiedijk's 100 theorems

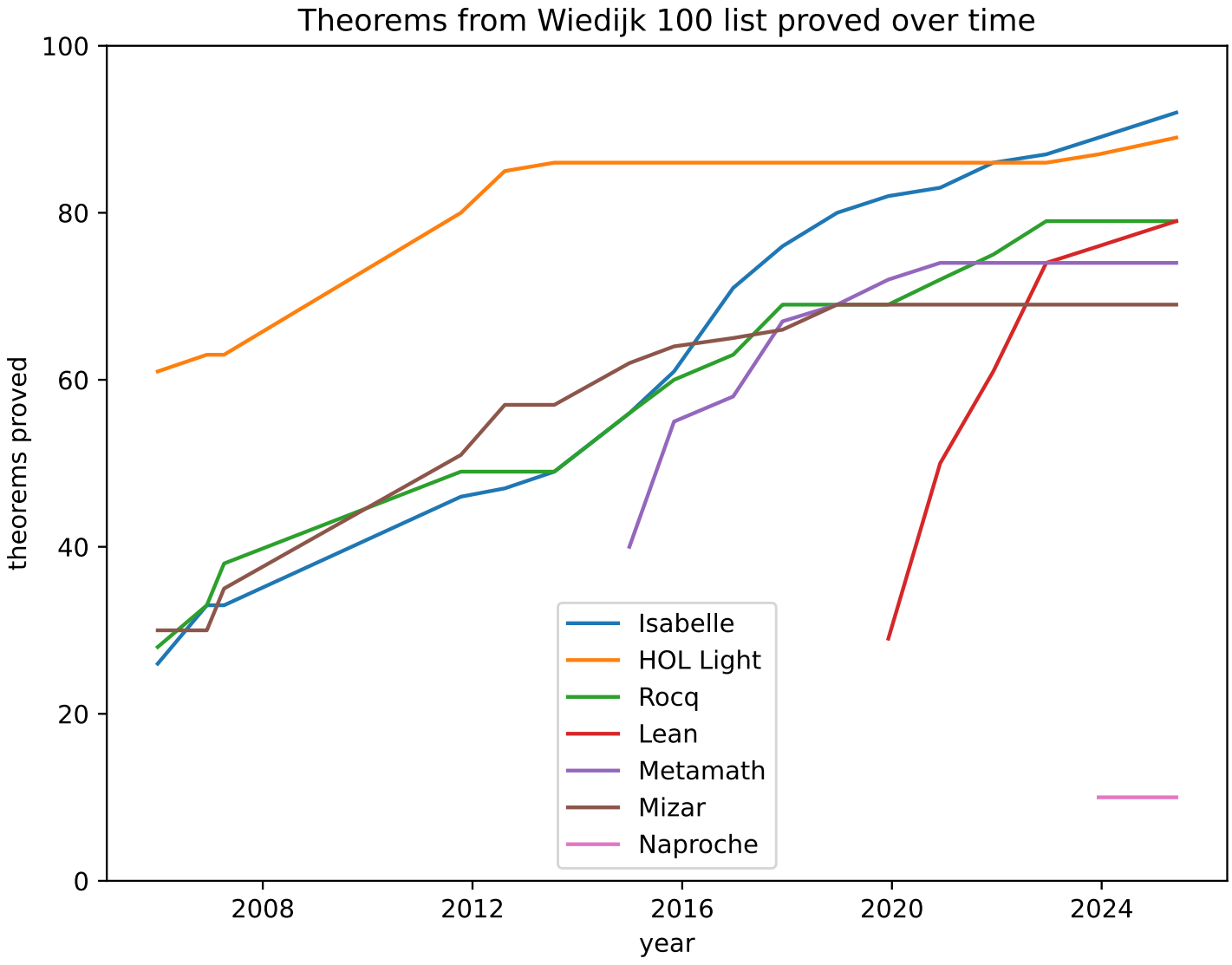
been formalized. Currently the fraction that already has been formalized seems to be

99%

The page does not keep track of *all* formalizations of these theorems. It just shows formalizations in systems that have formalized a significant number of theorems, or that have formalized a theorem that none of the others have done. The systems that this page refers to are (in order of the number of theorems that have been formalized, so the more interesting systems for mathematics are near the top):

Isabelle	92
HOL Light	89
Coq	79
Lean	79
Metamath	74
Mizar	69
nqthm/ACL2	47
ProofPower	43
PVS	26
Megalodon	12
Naproche	10
NuPRL/MetaPRL	8

Wiedijk's 100 theorems over time



Proof assistants: logical foundations

- First-order logic
  - Usual foundations: ZFC axioms for set theory
  - Everything (functions, integers, ...) built from sets
  - Mizar, Metamath, Naproche
- Classical higher-order logic
  - Evolved from Alonzo Church's work on simple type theory
  - Every variable has a type
  - Functions are primitive
  - Sets are usually functions of type $\tau \to \mathbb{B}$
  - HOL Light, Isabelle, Natty
- Dependent type theory
  - Martin-Lof type theory and descendents
  - Rocq, Lean
- Choice of foundation is visible to the user to some extent

Proof assistants: a spectrum of naturalness

Metamath | HOL Light | Isabelle | Mizar | Naproche Natty

⟵ Less natural — More natural ⟶

```
$( Prove a theorem $)
    th1 $p |- t = t $=
  $( Here is its proof: $)
       tt tze tpl tt weq tt tt weq tt a2 tt tze tpl
       tt weq tt tze tpl tt weq tt tt weq wim tt a2
       tt tze tpl tt tt a1 mp mp
     $.
```

Proof assistants: a spectrum of naturalness

Metamath | HOL Light | Isabelle | Mizar | Naproche Natty

⟵ Less natural — More natural ⟶

```
definition
  let n be Nat;
  func cseq n -> Real_Sequence means :Def3: :: IRRAT_1:def 3
  for k being Nat holds it . k = (n choose k) * (n ^ (- k));
  correctness
  proof end;
end;
```

Proof assistants: a spectrum of naturalness

Metamath | HOL Light | Isabelle | Mizar | Naproche Natty

$\longleftarrow$ Less natural — More natural $\longrightarrow$

Let us show that there exists a natural number a and nonzero natural number b such that $q = \frac{a}{b}$. Take an integer a and a nonzero integer b such that $q = \frac{a}{b}$.

Case $a = 0$ or a, b are natural numbers. Trivial.

Case ($a < 0$ and $b > 0$) or ($a > 0$ and $b < 0$). Then $\frac{a}{b} < 0$ and $q \geq 0$. Contradiction. End.

Case $a < 0$ and $b < 0$. Then $-a, -b \in \mathbb{N}$ and $q = \frac{-a}{-b}$. End. End. Take a natural number a and a nontrivial natural number b such that $q = \frac{a}{b}$.

Automatic provers

- E, Vampire, Zipperposition, ...
- Historically first-order logic, now also higher-order
- No major automatic provers for dependent type theory
- The most competitive provers use superposition
- Natty also contains an automatic prover

Why Natty has its own internal prover

- Natty needs to verify proof steps
  - in higher-order logic
  - quickly (< 5 sec)
  - reliably (100% success rate)
- Existing provers cannot do this!
- Natty's prover has a different objective
  - Goal: Prove easy theorems/steps quickly
  - Non-goal: Prove hard theorems (TPTP) in minutes

What Natty does

- Natural-language input: axioms, definitions, theorems
- A theorem may or may not include a proof
- Natty translates each theorem or proof step to a formula
- Can prove formulas directly, or export them to THF files
- Real-time feedback in Visual Studio Code

Natty: a young system

- 7 months of development
- 4000 lines of OCaml
- 66 theorems about $\mathbb{N}$ and $\mathbb{Z} \longrightarrow$ 335 proof steps
  - $\mathbb{N}$: proves 98% of steps, competitive performance
  - $\mathbb{Z}$: proves 82% of steps, slower
- Not quite ready for other mathematical domains
  - Type system, natural-language grammar need expansion
  - Automatic prover not scalable to larger premise sets

Performance comparison

Table: Theorems and proof steps ($\mathbb{N}$)

	Natty	E	Vampire	Zipperposition
proved (of 214)	209	179	173	185
proved (%)	98%	84%	81%	86%
average time	0.14	0.10	0.19	0.26

Table: Theorems and proof steps ($\mathbb{Z}$)

	Natty	E	Vampire	Zipperposition
proved (of 121)	99	107	90	75
proved (%)	82%	88%	74%	62%
average time	0.50	0.25	0.29	0.46

Comparison with Naproche

	Naproche	**Natty**
Logic	first-order	higher-order
Input	LaTeX	plain text with Unicode
Proof structure	explicit	implicit
Prover	usually E	internal
Written in	Haskell	OCaml
IDE	Isabelle	Visual Studio Code
Wiedjik thms proven	10	0

Outline

Getting Natty

- Currently you must build from source using OCaml 5.3.0
- Natty: `https://github.com/medovina/natty`
- VS Code extension:
  `https://github.com/medovina/natty-vscode`

A classic text

Explicationes.

Signo N significatur *numerus (integer positivus).*
» 1 » *unitas.*
» $a+1$ » *sequens a,* sive *a plus* 1.
» $=$ » *est aequalis.* Hoc ut novum signum considerandum est, etsi logicae signi figuram habeat.

Axiomata.

1. $1 \in \mathrm{N}.$
2. $a \in \mathrm{N} . \supset . a = a.$
3. $a, b, c \in \mathrm{N} . \supset : a = b . = . b = a.$
4. $a, b \in \mathrm{N} . \supset \therefore a = b . b = c : \supset . a = c.$
5. $a = b . b \in \mathrm{N} : \supset . a \in \mathrm{N}.$
6. $a \in \mathrm{N} . \supset . a + 1 \in \mathrm{N}.$
7. $a, b \in \mathrm{N} . \supset : a = b . = . a + 1 = b + 1.$
8. $a \in \mathrm{N} . \supset . a + 1 - = 1.$
9. $k \in \mathrm{K} \therefore 1 \in k \therefore x \in \mathrm{N} . x \in k : \supset_x . x + 1 \in k :: \supset . \mathrm{N} \supset k.$

Definitiones.

10. $2 = 1 + 1;\ 3 = 2 + 1;\ 4 = 3 + 1;$ etc.

- Who wrote this?
- When?

A classic text

Explicationes.

Signo N significatur *numerus (integer positivus).*
» 1 » *unitas.*
» $a+1$ » *sequens a,* sive *a plus* 1.
» $=$ » *est aequalis.* Hoc ut novum signum considerandum est, etsi logicae signi figuram habeat.

Axiomata.

1. $1 \in \mathrm{N}.$
2. $a \in \mathrm{N} . \supset . a = a.$
3. $a, b, c \in \mathrm{N} . \supset : a = b . = . b = a.$
4. $a, b \in \mathrm{N} . \supset \therefore a = b . b = c : \supset . a = c.$
5. $a = b . b \in \mathrm{N} : \supset . a \in \mathrm{N}.$
6. $a \in \mathrm{N} . \supset . a + 1 \in \mathrm{N}.$
7. $a, b \in \mathrm{N} . \supset : a = b . = . a + 1 = b + 1.$
8. $a \in \mathrm{N} . \supset . a + 1 - = 1.$
9. $k \in \mathrm{K} \therefore 1 \in k \therefore x \in \mathrm{N} . x \in k : \supset_x . x + 1 \in k :: \supset . \mathrm{N} \supset k.$

Definitiones.

10. $2 = 1 + 1;\ 3 = 2 + 1;\ 4 = 3 + 1;$ etc.

- Who wrote this?
- When?

Giuseppe Peano

Arithmetices Principia, 1889

- Peano axioms for $\mathbb{N}$
- Origin of $\in$, $\cap$, $\cup$, $\subset$

Arithmetices Principia: Logical propositions

Sint $a, b, c, \ldots$ propositiones. Tunc erit:

1. $a \supset a.$
2. $a \supset b \,.\, b \supset c : \supset : a \supset c.$
3. $a = b \,.=: a \supset b \,.\, b \supset a.$
4. $a = a.$
5. $a = b \,.=.\, b = a.$
6. $a = b \,.\, b \supset c : \supset . a \supset c.$
7. $a \supset b \,.\, b = c : \supset . a \supset c.$
8. $a = b \,.\, b = c : \supset . a = c.$
9. $a = b \,.\, \supset . a \supset b.$
10. $a = b \,.\, \supset . b \supset a.$

---

11. $ab \supset a.$
12. $ab = ba.$
13. $a\,(bc) = (ab)\,c = abc.$

Arithmetices Principia: Logical propositions

Sint $a, b, c, \ldots$ propositiones. Tunc erit:

1. $a \supset a.$
2. $a \supset b . b \supset c : \supset : a \supset c.$
3. $a = b . = : a \supset b . b \supset a.$
4. $a = a.$
5. $a = b . = . b = a.$
6. $a = b . b \supset c : \supset . a \supset c.$
7. $a \supset b . b = c : \supset . a \supset c.$
8. $a = b . b = c : \supset . a = c.$
9. $a = b . \supset . a \supset b.$
10. $a = b . \supset . b \supset a.$

---

11. $ab \supset a.$
12. $ab = ba.$
13. $a(bc) = (ab)c = abc.$

Theorem 1. Let $a, b, c \in \mathbb{B}$.

1. $a \to a$.
2. $(a \to b) \wedge (b \to c) \to (a \to c)$.
3. $(a = b) \leftrightarrow ((a \to b) \wedge (b \to a))$.
4. $a = a$.
5. $(a = b) \leftrightarrow (b = a)$.

Arithmetices Principia: Axioms for $\mathbb{N}$

Axiomata.

1. $1 \in \mathrm{N}$.
2. $a \in \mathrm{N} . \supset . a = a$.
3. $a, b, c \in \mathrm{N} . \supset : a = b . = . b = a$.
4. $a, b \in \mathrm{N} . \supset \therefore a = b . b = c : \supset . a = c$.
5. $a = b . b \in \mathrm{N} : \supset . a \in \mathrm{N}$.
6. $a \in \mathrm{N} . \supset . a + 1 \in \mathrm{N}$.
7. $a, b \in \mathrm{N} . \supset : a = b . = . a + 1 = b + 1$.
8. $a \in \mathrm{N} . \supset . a + 1 - = 1$.
9. $k \in \mathrm{K} \therefore 1 \in k \therefore x \in \mathrm{N} . x \in k : \supset_x . x + 1 \in k : : \supset . \mathrm{N} \supset k$.

Definitiones.

10. $2 = 1 + 1;\ 3 = 2 + 1;\ 4 = 3 + 1;$ etc.

Arithmetices Principia: First theorems

11. $2 \in \mathrm{N}$.

Demonstratio:

$\mathrm{P}\,1 . \supset :$	$1 \in \mathrm{N}$	(1)
$1\,[a]\,(\mathrm{P}\,6) . \supset :$	$1 \in \mathrm{N} . \supset . 1 + 1 \in \mathrm{N}$	(2)
$(1)(2) . \supset :$	$1 + 1 \in \mathrm{N}$	(3)
$\mathrm{P}\,10 . \supset :$	$2 = 1 + 1$	(4)
$(4) . (3) . (2, 1 + 1)\,[a, b]\,(\mathrm{P}\,5) : \supset :$	$2 \in \mathrm{N}$	(Theorema).

Nota. — Huius facillimae demonstrationis gradus omnes explicite scripsimus. Brevitatis causa ipsam ita scribemus:

$\mathrm{P}\,1 . 1\,[a]\,(\mathrm{P}\,6) : \supset : 1 + 1 \in \mathrm{N} . \mathrm{P}\,10 . (2, 1 + 1)\,[a, b]\,(\mathrm{P}\,5) : \supset : \mathrm{Th}.$

vel

$\mathrm{P}\,1 . \mathrm{P}\,6 : \supset : 1 + 1 \in \mathrm{N} . \mathrm{P}\,10 . \mathrm{P}\,5 : \supset : \mathrm{Th}.$

12. $3, 4, \ldots \in \mathrm{N}$.

13. $a, b, c, d \in \mathrm{N} . a = b . b = c . c = d : \supset : a = d$.

Dem. Hyp. $\mathrm{P}\,4 : \supset : a, c, d \in \mathrm{N} . a = c . c = d . \mathrm{P}4 : \supset : \mathrm{Thes}.$

14. $a, b, c \in \mathrm{N} . a = b . b = c . a - = c : = \Lambda$.

Dem. $\mathrm{P}\,4 . \mathrm{L}\,39 : \supset . \mathrm{Theor}.$

15. $a, b, c \in \mathrm{N} . a = b . b - = c : \supset . a - = c$.

16. $a, b \in \mathrm{N} . a = b : \supset . a + 1 = b + 1$.

16′. $a, b \in \mathrm{N} . a + 1 = b + 1 : \supset . a = b$.

Dem. $\mathrm{P}\,7 = (\mathrm{P}\,16)\,(\mathrm{P}\,16')$.

17. $a, b \in \mathrm{N} . \supset : a - = b . = . a + 1 - = b + 1$.

Dem. $\mathrm{P}\,7 . \mathrm{L}\,21 : \supset . \mathrm{Theor}.$

Arithmetices Principia: Definition of addition

Definitio.

18. $a, b \in \mathrm{N} . \supset . a + (b + 1) = (a + b) + 1.$

Nota. — Hanc definitionem ita legere oportet: si a et b sunt numeri, et $(a + b) + 1$ sensum habet (scilicet si $a + b$ est numerus), sed $a + (b + 1)$ nondum definitus est, tunc $a + (b + 1)$ significat numerum qui $a + b$ sequitur.

Ab hac definitione, et a praecedentibus deducitur:

$$a \in \mathrm{N} . \supset \therefore a + 2 = a + (1 + 1) = (a + 1) + 1.$$

$$a \in \mathrm{N} . \supset \therefore a + 3 = a + (2 + 1) = (a + 2) + 1, \text{ etc.}$$

Arithmetices Principia: Theorems about addition

20. *Def.* $a+b+c=(a+b)+c.$

21. $a,b,c\,\epsilon\,\mathrm{N}\,.\,\supset\,.\,a+b+c\,\epsilon\,\mathrm{N}.$

22. $a,b,c\,\epsilon\,\mathrm{N}\,.\,\supset:a=b\,.=.\,a+c=b+c.$

Dem. $a,b\,\epsilon\,\mathrm{N}\,.\,\mathrm{P}\,7:\supset\,.\,1\,\epsilon\,[c\,\epsilon]\,\mathrm{Ts}.$ (1)

$a,b\,\epsilon\,\mathrm{N}\,.\,\supset::c\,\epsilon\,\mathrm{N}\,.\,c\,\epsilon\,[c\,\epsilon]\,\mathrm{Ts}\,\therefore\,\supset\,\therefore\,a=b\,.=.\,a+c=b+c:$
$a+c,b+c\,\epsilon\,\mathrm{N}:a+c=b+c\,.=.\,a+c+1=b+c+$
$1\,\therefore\,\supset\,\therefore\,a=b\,.=.\,a+(c+1)=b+(c+1)\,\therefore\,\supset\,\therefore(c+1)$
$\epsilon\,[c\,\epsilon]\,\mathrm{Ts}.$ (2)

$a,b\,\epsilon\,\mathrm{N}\,.\,(1).(2):\supset::1\,\epsilon\,[c\,\epsilon]\,\mathrm{Ts}.\therefore c\,\epsilon\,[c\,\epsilon]\,\mathrm{Ts}\,.\supset.\,(c+1)\,\epsilon\,[c\,\epsilon]$
$\mathrm{Ts}::\supset::c\,\epsilon\,\mathrm{N}\,.\,\supset\,.\,\mathrm{Ts}.$ (3)

(3) $\supset$ Theor.

23. $a,b,c\,\epsilon\,\mathrm{N}\,.\,\supset\,.\,a+(b+c)=a+b+c.$

Dem. $a,b\,\epsilon\,\mathrm{N}.\,\mathrm{P}\,18\,.\,\mathrm{P}\,20:\supset\,.\,1\,\epsilon\,[c\,\epsilon]\,\mathrm{Ts}.$ (1)

$a,b\,\epsilon\,\mathrm{N}\,.\,\supset\,\therefore\,c\,\epsilon\,\mathrm{N}\,.\,c\,\epsilon\,[c\,\epsilon]\,\mathrm{Ts}:\supset:a+(b+c)=a+b+c.$
$\mathrm{P}\,7:\supset:a+(b+c)+1=a+b+c+1\,.\,\mathrm{P}\,18:\supset:a+$
$(b+(c+1))=a+b+(c+1):\supset.\,c+1\,\epsilon\,[c\,\epsilon]\,\mathrm{Ts}.$ (2)

(1) (2) (P 9) . $\supset$. Theor.

24. $a\,\epsilon\,\mathrm{N}\,.\,\supset\,.\,1+a=a+1.$

Dem. $\mathrm{P}\,2\,.\,\supset\,.\,1\,\epsilon\,[a\,\epsilon]\,\mathrm{Ts}.$ (1)

$a\,\epsilon\,\mathrm{N}\,.\,a\,\epsilon\,[a\,\epsilon]\,\mathrm{Ts}:\supset:1+a=a+1:\supset:1+(a+1)=(a+$
$1)+1:\supset:(a+1)\,\epsilon\,[a\,\epsilon]\,\mathrm{Ts}.$ (2)

(1) (2) . $\supset$. Theor.

Arithmetices Principia: More theorems about addition

25. $a, b \in \mathrm{N} . \supset . a + b = b + a.$

Dem. $a \in \mathrm{N} . \mathrm{P}\,24 : \supset : 1 \in [b \in] \mathrm{Ts}.$ (1)

$a \in \mathrm{N} . \supset \therefore b \in \mathrm{N} . b \in [b \in] \mathrm{Ts} : \supset : a + b = b + a . \mathrm{P}\,7 : \supset : (a + b) + 1 = (b + a) + 1 . (a + b) + 1 = a + (b + 1) . (b + a) + 1 = 1 + (b + a) . 1 + (b + a) = (1 + b) + a . (1 + b) + a = (b + 1) + a : \supset : a + (b + 1) = (b + 1) + a : \supset : (b + 1) \in [b \in] \mathrm{Ts}.$ (2)

(1) (2) . ⊃ . Theor.

26. $a, b, c \in \mathrm{N} . \supset : a = b . = . c + a = c + b.$

27. $a, b, c \in \mathrm{N} . \supset : a + b + c = a + c + b.$

28. $a, b, c, d \in \mathrm{N} . a = b . c = d : \supset . a + c = b + d.$

Arithmetics Principia: theorems in Natty

§1. Numbers and addition

Axiom 1. There exists a type $\mathbb{N}$ with a constant $1 \in \mathbb{N}$ and a binary operation $+ : \mathbb{N} \to \mathbb{N} \to \mathbb{N}$ such that

7. For all $a, b \in \mathbb{N}$, $a = b$ iff $a + 1 = b + 1$.
8. For all $a \in \mathbb{N}$, $a + 1 \neq 1$.
9. Let $A : \mathbb{N} \to \mathbb{B}$. If $1 \in A$, and $x \in A$ implies $x + 1 \in A$ for all $x \in \mathbb{N}$, then $A = \mathbb{N}$.

Theorem 1. Let $a, b \in \mathbb{N}$.

17. If $a \neq b$, then $a + 1 \neq b + 1$.

Axiom 2. For all $a, b \in \mathbb{N}$,

18. $a + (b + 1) = (a + b) + 1$.

Theorem 2. Let $a, b, c \in \mathbb{N}$.

22. $a = b$ iff $a + c = b + c$.
23. $a + (b + c) = (a + b) + c$.
24. $1 + a = a + 1$.
25. $a + b = b + a$.
26. $a = b$ iff $c + a = c + b$.

Input file `nat.n`

- The naturals $\mathbb{N}$
  - Defined using Peano axioms
  - $+$, $\cdot$, $>$, $<$ on $\mathbb{N}$ defined axiomatically
  - 40 theorems
- The integers $\mathbb{Z}$
  - Defined axiomatically using $\mathbb{N}$
  - $+$, $\cdot$, $>$, $<$ on $\mathbb{Z}$ defined axiomatically
  - 26 theorems
- A foundation for more number theory

Natural input: universal sets

Axiom 1. There exists a type $\mathbb{N}$ with an element $0 \in \mathbb{N}$ and a function $s : \mathbb{N} \to \mathbb{N}$ such that

a. There is no $n \in \mathbb{N}$ such that $s(n) = 0$.
b. For all $n, m \in \mathbb{N}$, if $s(n) = s(m)$ then $n = m$.
c. Let $P : \mathbb{N} \to \mathbb{B}$. If $P(0)$ is true, and $P(k)$ implies $P(s(k))$ for all $k \in \mathbb{N}$, then $P = \mathbb{N}$.

Definition. Let $1 : \mathbb{N} = s(0)$.

- $\mathbb{N}$ is a type
- It is also the universal set $\lambda x : \mathbb{N} \,.\, \top$

Natural input: implicit multiplication

Theorem 3. Let $a, b, c \in \mathbb{N}$.

1. $a \cdot 0 = 0 = 0 \cdot a$.
2. $a \cdot 1 = a = 1 \cdot a$.
3. $c(a + b) = ca + cb$.
4. $(ab)c = a(bc)$.
5. $s(a) \cdot b = ab + b$.
6. $ab = ba$.
7. $(a + b)c = ac + bc$.

- $c(a + b)$ means $c \cdot (a + b)$
- $s(a)$ is a function application

Types distinguish these cases!

Outline

Parsing

- Context-free grammar (200 lines of EBNF)
- Parsed using parser combinators

```
so = 'also' | 'hence' | 'so' | 'then' | 'therefore' |
    'which implies' ['that'] | 'which means that';

have = 'clearly' |
    'it' ['then'] 'follows' ['from' reference | reason] 'that' |
    'it is clear that' | 'it must be that' | 'similarly' [','] |
    'the only alternative is' |
    'we conclude that' | 'we deduce that' | 'we have' | 'we know that' |
    'we must have' | 'we see that';

contra = ',' ['which is'] ['again' | 'also' | 'similarly']
         'a contradiction' ['to' theorem_ref];
```

Parser: output

- Parser outputs a series of statements
  - TypeDecl, ConstDecl, Axiom, Definition, Theorem
- Each proof is parsed into a series of steps
  - Assert, Let, LetVal, Assume, IsSome, Escape

Proof steps for Peano theorem 22

22. Let a, b ∈ ℕ. Let

$$G = \{ c \in \mathbb{N} \mid \text{if } a + c = b + c, \text{ then } a = b \}.$$

1 ∈ G by Axiom 1.7. Now let c : ℕ, and suppose that c ∈ G. We will show that c + 1 ∈ G. Suppose that a + (c + 1) = b + (c + 1). Then (a + c) + 1 = (b + c) + 1 by Axiom 2.18. Then a + c = b + c by Axiom 1.7, and so a = b by the inductive hypothesis. It follows that c + 1 ∈ G. So G = ℕ.

```
let a, b : ℕ
let_val G : ? = λc:ℕ.(a + c = b + c → a = b)
assert 1 ∈ G
escape
let c : ℕ
assume c ∈ G
assume a + (c + 1) = b + (c + 1)
assert (a + c) + 1 = (b + c) + 1
assert a + c = b + c
assert a = b
assert c + 1 ∈ G
assert G = ℕ
```

Proof structure inference

- Parser outputs a linear series of proof steps
- Natty will infer a block structure for the proof
- Important question: when are assumptions discharged?
- If the block structure is wrong, proof will fail to verify!

Proof tree for Peano theorem 22

22. Let a, b ∈ ℕ. Let

$$G = \{ c \in \mathbb{N} \mid \text{if } a + c = b + c, \text{ then } a = b \}.$$

1 ∈ G by Axiom 1.7. Now let c : ℕ, and suppose that c ∈ G. We will show that c + 1 ∈ G. Suppose that a + (c + 1) = b + (c + 1). Then (a + c) + 1 = (b + c) + 1 by Axiom 2.18. Then a + c = b + c by Axiom 1.7, and so a = b by the inductive hypothesis. It follows that c + 1 ∈ G. So G = ℕ.

```
let a, b : ℕ
  let_val G : ? = λc:ℕ.(a + c = b + c → a = b)
    assert 1 ∈ G
    let c : ℕ
      assume c ∈ G
        assume a + (c + 1) = b + (c + 1)
          assert (a + c) + 1 = (b + c) + 1
          assert a + c = b + c
          assert a = b
          assert c + 1 ∈ G
    assert G = ℕ
assert ∀a:ℕ.∀b:ℕ.∀c:ℕ.(a = b ↔ a + c = b + c)
```

Proof structure inference: heuristics

- Definition: a block is a step plus all its descendents in the tree
- A Let or LetVal block encloses a scope that is as small as possible, however
  - must enclose all steps that refer to variable(s) introduced by S
- An Assume or IsSome block extends to the end of its nearest enclosing Let or LetVal block
- An Escape step or assertion of $\perp$ will force an Assume block to end

Proof structure inference: Escape steps

- Words such as “now”, “next” emit an Escape step
- This forces an assumption to be discharged

Theorem 7. Let $a, b \in \mathbb{N}$.

1. Precisely one of $a < b$ or $a = b$ or $a > b$ holds.

...

Proof.

1. Let $a, b \in \mathbb{N}$. Suppose that $a < b$ and $a = b$. It then follows that $a < a$, which is a contradiction to Theorem 6.1. Now suppose that $a = b$ and $a > b$. It follows that $a > a$, which is similarly a contradiction. Now suppose that $a < b$ and $b < a$. By Theorem 6.6 we deduce that $a < a$, again a contradiction.

Formula generation

- After parsing and type checking, Natty generates formulas
- Each formula has a set of premises available for proving it
- A theorem with no proof generates a single formula
  - Its premises are all previous axioms/theorems in the input
- If a theorem has a proof:
  - Each step Assert(ϕ) produces the formula ϕ
  - Each step IsSome(x, τ, ψ) produces the formula $\exists x : \tau \,.\, \psi$
  - A formula's premises are the conclusions of earlier steps in the proof, plus axioms/theorems that appeared earlier in the input

Natty's automatic prover

- Based on higher-order superposition calculus
  - ... but only an incomplete fragment
- Negates the goal, searches for $\bot$
- Uses DISCOUNT loop, also found in E
- Unification is mostly first-order
- Term ordering uses lexicographic path order

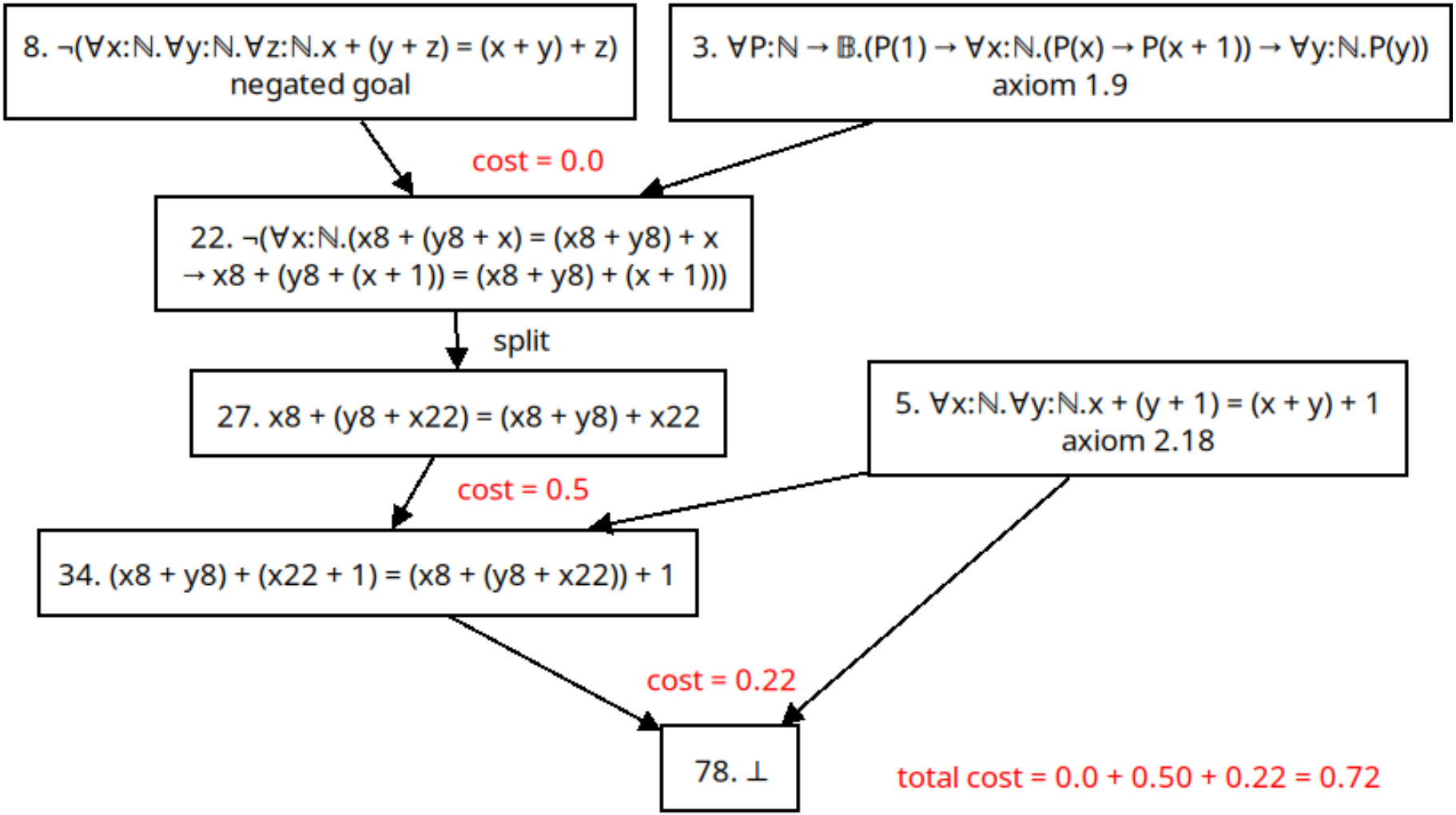
Proof procedure

// U = unprocessed (passive) set
// P = processed (active) set
while U $\neq$ 0 **do**
    Choose a given formula f from U
    Remove f from U, add f to P
    Simplify formulas in P by rewriting
    Generate new formulas G from P via superposition
    if any formula in G is $\perp$ then **exit** (proof found)
    Add formulas G to U

Given formula selection

- A single priority queue holds all unprocessed formulas
- The queue orders formulas by cost
- The cost of a formula is the sum of the costs of all superposition steps in its derivation
- The cost of each superposition step is calculated heuristically using a function that was derived via simple machine learning

The cost of a formula



Heuristic cost function

- Natty can output a CSV file with all formulas generated during a proof
- Each formula has features (weight, # of literals, ...)
- Each formula was either used or not used in the proof
- Using supervised learning, we can build a classifier that predicts the probability that a formula will be used
- Natty uses logistic regression for classification
- The lasso $= \ell_1$ regularization selects a small number of features
- Goal: a simple, explainable model

The learned cost function

// The cost of a superposition step producing ϕ from ψ_1, ψ_2.
LEARNED-COST(ϕ) =
  0.668
  + 0.041 if ϕ was generated by paramodulation
  − 0.030 if any ancestor of ϕ is a hypothesis
  − 0.251 if any ancestor of ϕ is the goal formula
  − 0.007 if ψ_1 or ψ_2 is a definition
  − 0.399 if ψ_1 or ψ_2 is an inductive formula
  − 0.009 if lits(ϕ) $<$ min(lits(ψ_1), lits(ψ_2))
  + 0.007 if lits(ϕ) > 1
  + 0.082 $\cdot$ (lits(ϕ) − lits(ψ_2))
  + 0.008 $\cdot$ (weight(ϕ) − max(weight(ψ_1), weight(ψ_2)))
  + 0.002 $\cdot$ (weight(ϕ) − weight(ψ_2))
  − 0.182 if ϕ was generated by resolution and
        weight(ϕ) $<$ min(weight(ψ_1), weight(ψ_2))

Why do provers fail to prove trivial proof steps?

- Some easy proof steps require only one or two superpositions
- But provers such as E and Vampire may fail to prove them!

Proof. Let

$$G = \{ x \in \mathbb{N} \mid \text{for all } y, z \in \mathbb{N}, \text{ if } z \neq 0 \text{ and } xz = yz \text{ then } x = y \}.$$

Let $b, c \in \mathbb{N}$ with $c \neq 0$ and $0 \cdot c = bc$. Then $bc = 0$. Since $c \neq 0$, we must have $b = 0$ by Theorem 4.1. So $0 = b$, and hence $0 \in G$.

Now let $a \in \mathbb{N}$, and suppose that $a \in G$. Let $b, c \in \mathbb{N}$, and suppose that $c \neq 0$ and $s(a) \cdot c = bc$. Then by Theorem 3.5 we deduce that $ca + c = bc$. If $b = 0$, then either $s(a) = 0$ or $c = 0$, which is a contradiction. Hence $b \neq 0$. By Lemma 1 there is some $p \in \mathbb{N}$ such that $b = s(p)$. Therefore $ca + c = s(p) \cdot c$, and we see that $ca + c = cp + c$. It follows by Theorem 2.1 that $ca = cp$, so $ac = pc$. By hypothesis it follows that $a = p$. Therefore $s(a) = s(p) = b$. Hence $s(a) \in G$, and we deduce that $G = \mathbb{N}$.

Rewriting can make easy problems hard

- Destructive rewriting is critical for prover efficiency
- But it can turn easy problems into hard ones!
- Natty combats this in two ways
  - Quick refute searches for single-step proofs before main refutation loop
  - Main loop introduces goal, then hypotheses, then goal again

Outline

Future work

- Natural language
  - Allow types named by words, e.g. “a and b are integers”
  - Allow predicates named by words, e.g. “n is prime”
  - Allow relations named by words, e.g. “a divides b”
- Type system
  - Polymorphic types, so we can talk about sets of any type
  - Will probably require type inference
  - May make performance comparisons with E more difficult
- Automatic prover
  - Include heuristic cost for known formulas
  - Term index
- Infrastructure
  - Allow a theory to span multiple source files
  - Cache prover results