


the next generation of proof assistants

Freek Wiedijk

Radboud University Nijmegen
The Netherlands 

2010 08 31 , 16 : 30


LSFA 2010
Natal, Brazil 

the next generation of proof assistants:
ten questions

Freek Wiedijk

Radboud University Nijmegen
The Netherlands 

2010 08 31 , 16 : 30

LSFA 2010
Natal, Brazil 


the state of the art

some of the best current proof assistants

 ACL2

 B method

 PVS

 HOL

 Isabelle

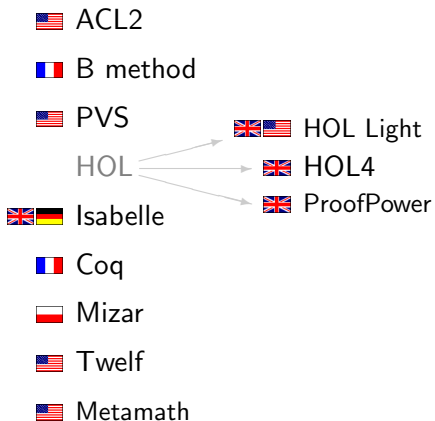
 Coq

 Mizar

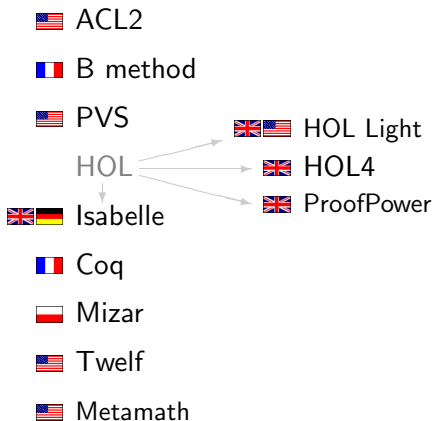
 Twelf

 Metamath

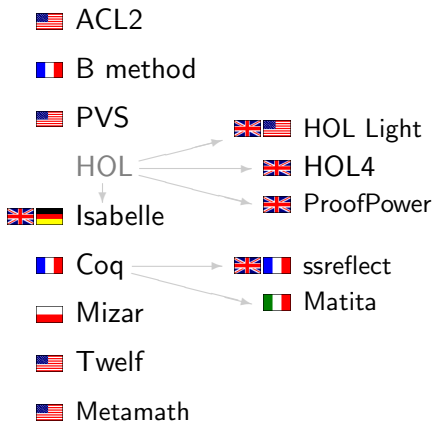
some of the best current proof assistants



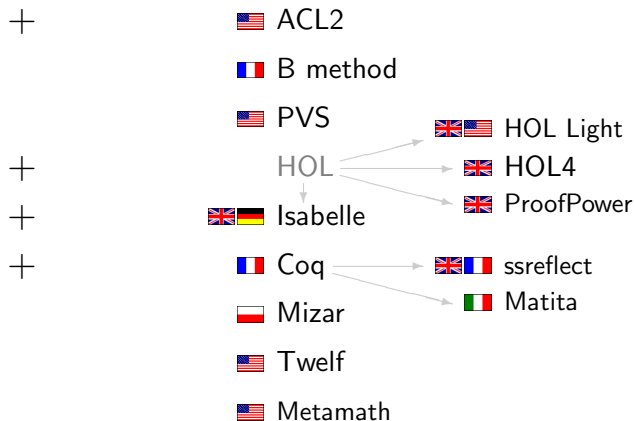
some of the best current proof assistants



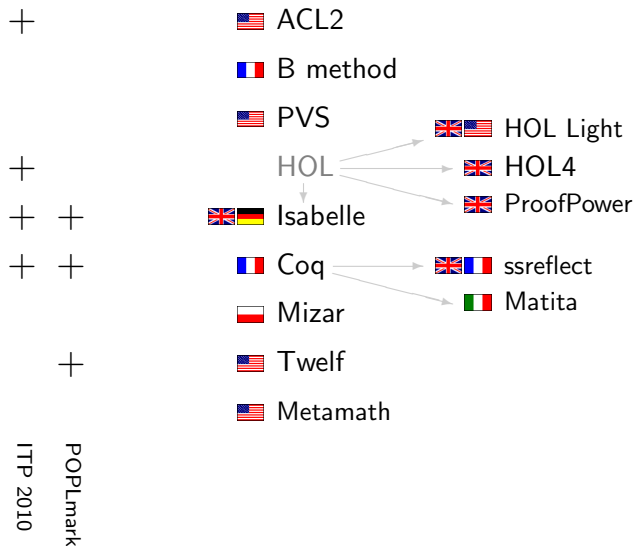
some of the best current proof assistants



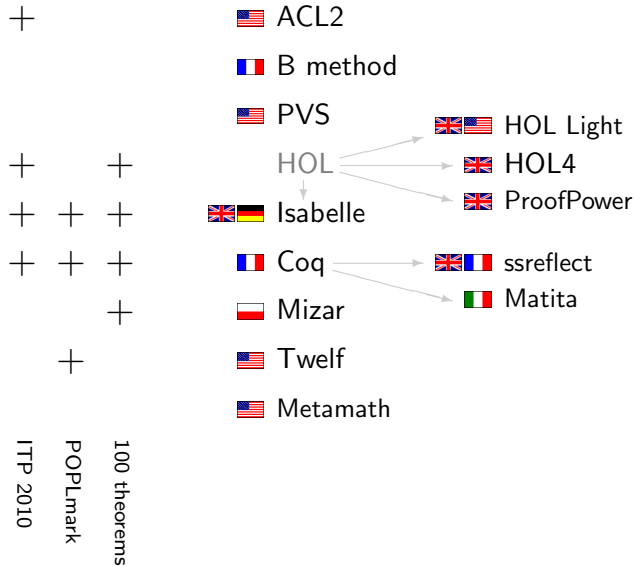
some of the best current proof assistants



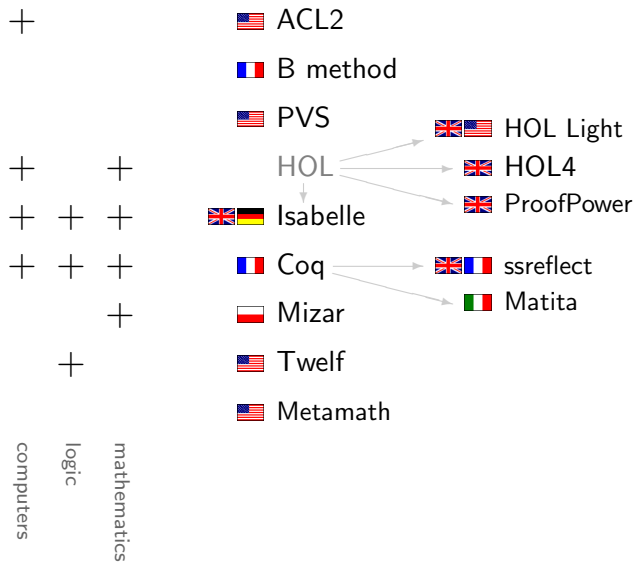
some of the best current proof assistants



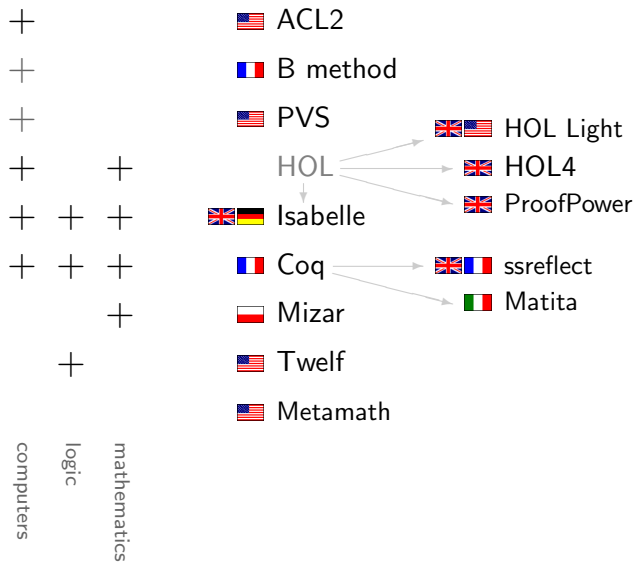
some of the best current proof assistants



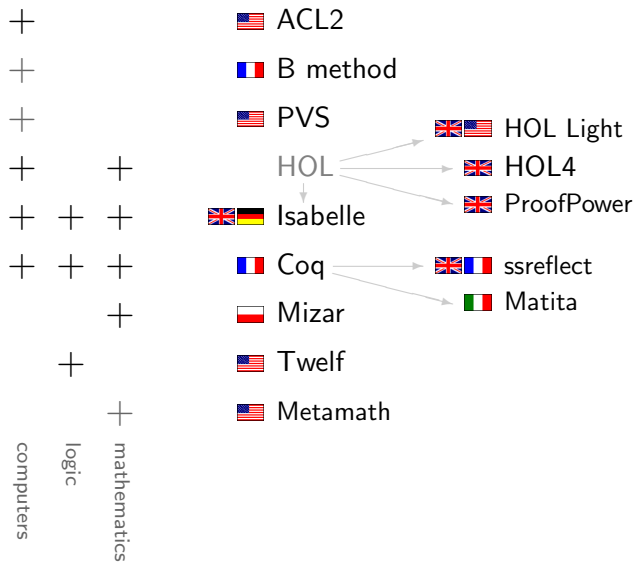
some of the best current proof assistants



some of the best current proof assistants



some of the best current proof assistants



the computer science spectrum of formal proof

circuits made of electronic components like transistors

the computer science spectrum of formal proof

circuits made of logical gates as described in a netlist



circuits made of electronic components like transistors

the computer science spectrum of formal proof

circuits specified on the register transfer level like in Verilog



circuits made of logical gates as described in a netlist



circuits made of electronic components like transistors

the computer science spectrum of formal proof

machine code



circuits specified on the register transfer level like in Verilog



circuits made of logical gates as described in a netlist



circuits made of electronic components like transistors

the computer science spectrum of formal proof

assembly programming languages



machine code



circuits specified on the register transfer level like in Verilog



circuits made of logical gates as described in a netlist



circuits made of electronic components like transistors

the computer science spectrum of formal proof

low level programming languages like C
assembly programming languages



machine code



circuits specified on the register transfer level like in Verilog



circuits made of logical gates as described in a netlist



circuits made of electronic components like transistors

the computer science spectrum of formal proof

high level programming languages like Haskell

low level programming languages like C

assembly programming languages



machine code



circuits specified on the register transfer level like in Verilog

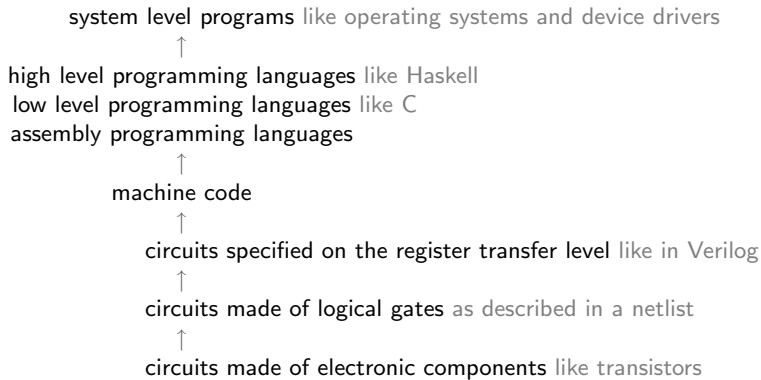


circuits made of logical gates as described in a netlist

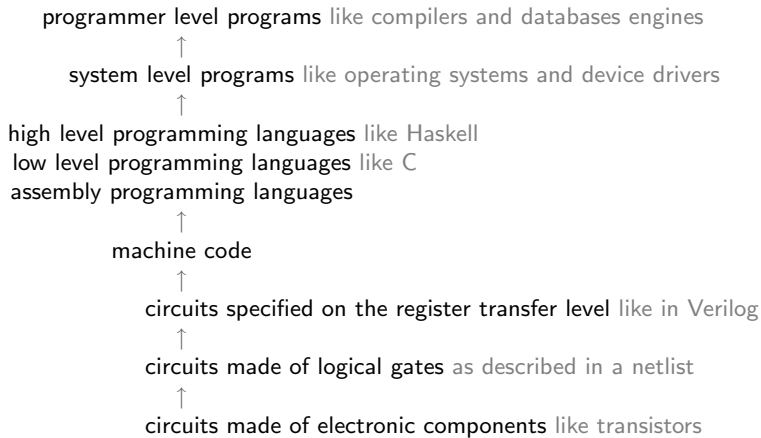


circuits made of electronic components like transistors

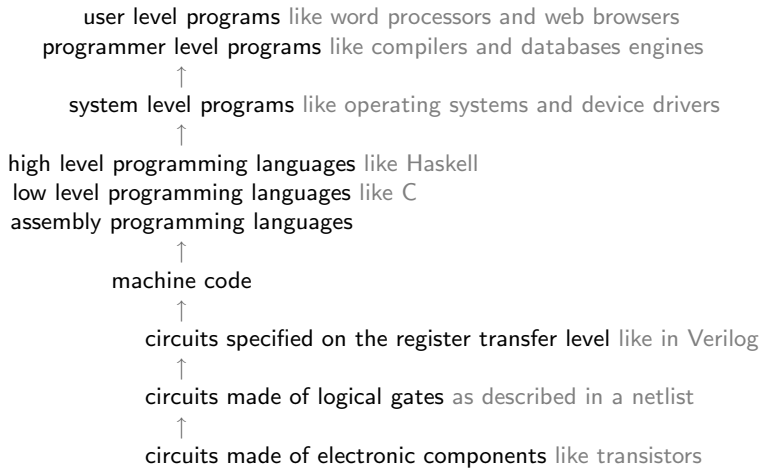
the computer science spectrum of formal proof



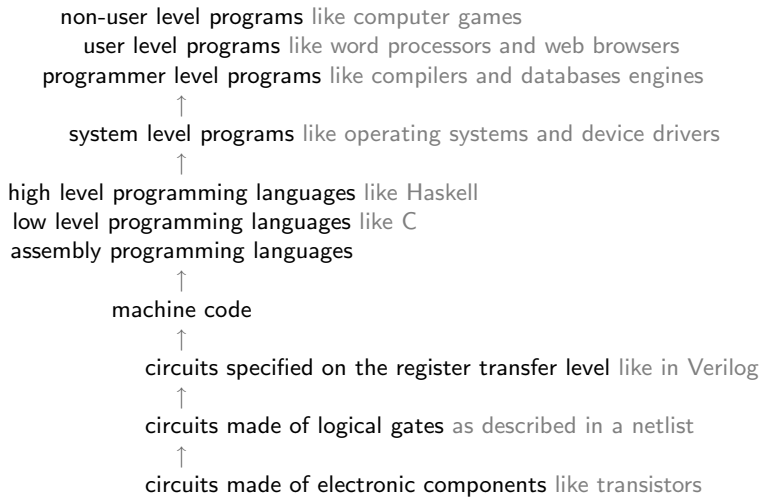
the computer science spectrum of formal proof



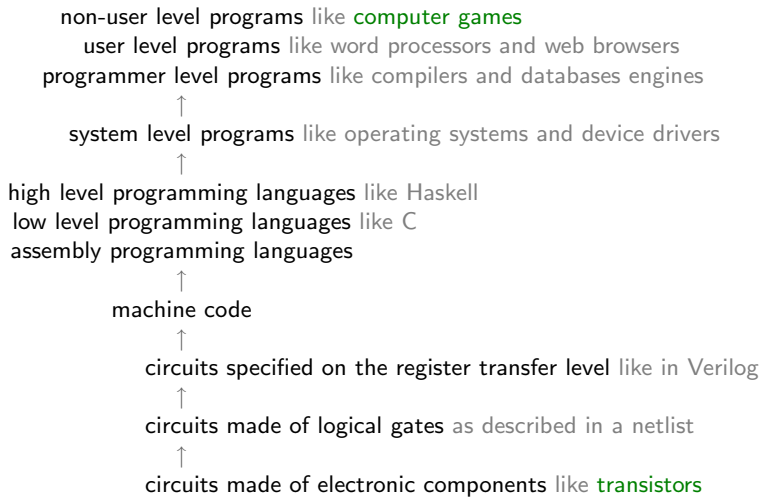
the computer science spectrum of formal proof



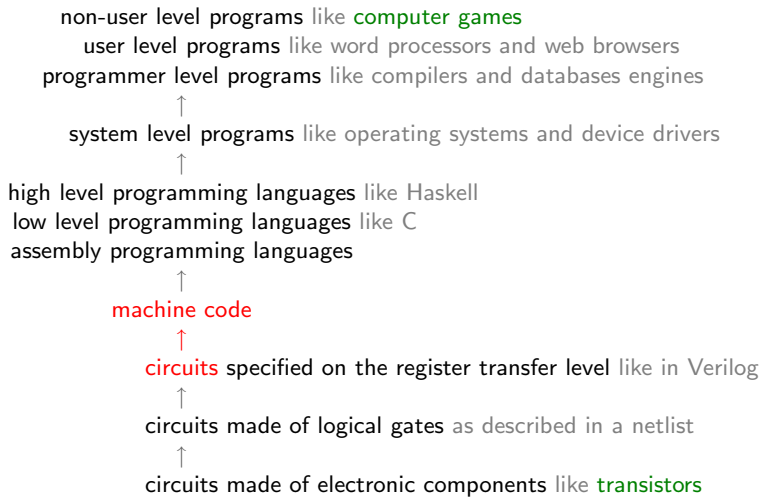
the computer science spectrum of formal proof



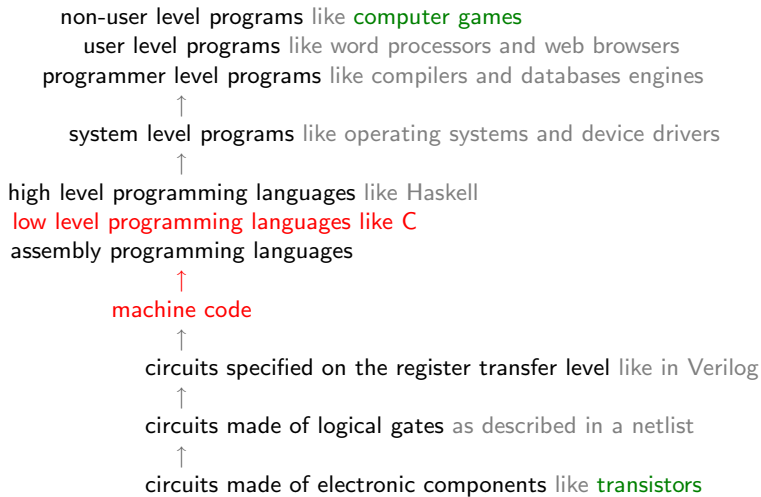
the computer science spectrum of formal proof



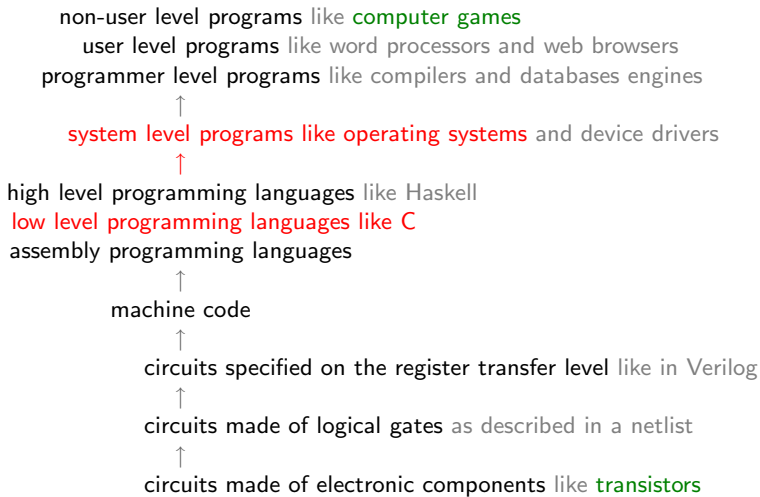
the computer science spectrum of formal proof



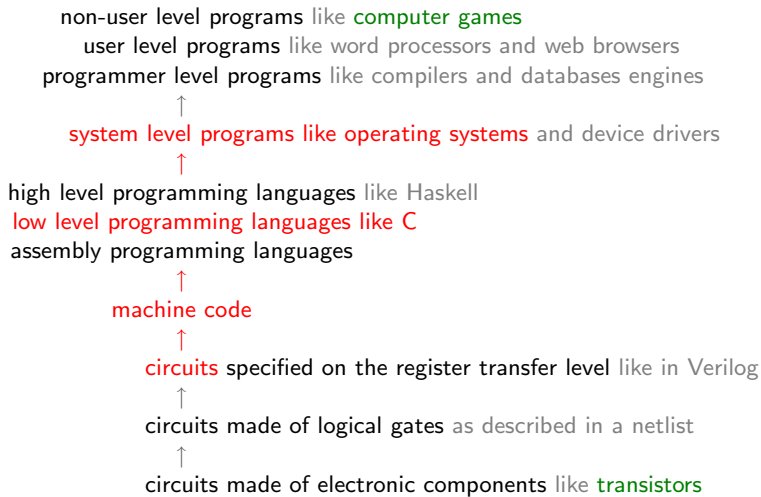
the computer science spectrum of formal proof



the computer science spectrum of formal proof

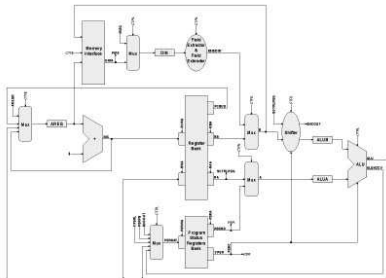


the computer science spectrum of formal proof

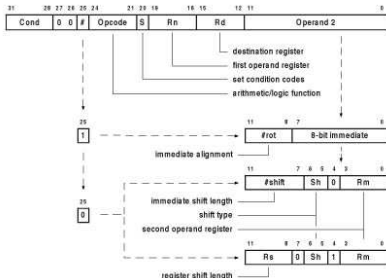


formally proving a processor correct

Anthony Fox, HOL4,  University of Cambridge, 2002




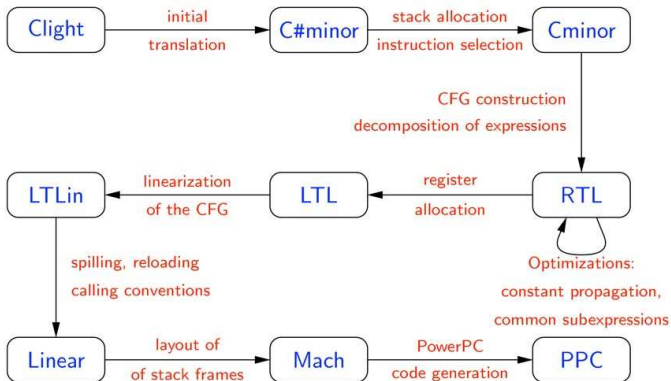
ARM6 architecture




ARMv3 instruction set

formally proving a compiler correct

Xavier Leroy, Coq,  INRIA, 2006



formally proving an operating system correct


Gerwin Klein, Isabelle,  NICTA, 2009

L4.verified project = verification of seL4 microkernel

1 microkernel
8,700 lines of C
0 bugs*
qed

* conditions apply

formally proving an operating system correct

Gerwin Klein, Isabelle,  NICTA, 2009

L4.verified project = verification of seL4 microkernel

1 microkernel

8,700 lines of C


0 bugs*

qed

* conditions apply

5,700 lines of Haskell

formally proving an operating system correct

Gerwin Klein, Isabelle,  NICTA, 2009

L4.verified project = verification of seL4 microkernel

1 microkernel

8,700 lines of C

0 bugs*

qed


* conditions apply

5,700 lines of Haskell

117,000 lines of Isabelle

20 person-years

formally proving an operating system correct

Gerwin Klein, Isabelle,  NICTA, 2009

L4.verified project = verification of seL4 microkernel

1 microkernel

8,700 lines of C

0 bugs*

qed

* conditions apply

5,700 lines of Haskell

117,000 lines of Isabelle

20 person-years

the mathematics spectrum of formal proof

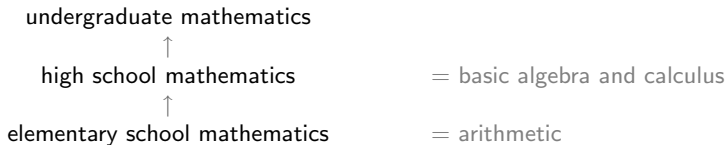
the mathematics spectrum of formal proof

elementary school mathematics = arithmetic

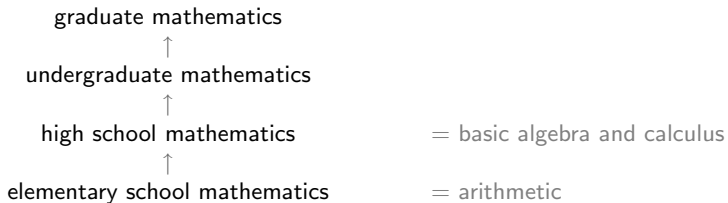
the mathematics spectrum of formal proof

high school mathematics	= basic algebra and calculus
↑	
elementary school mathematics	= arithmetic

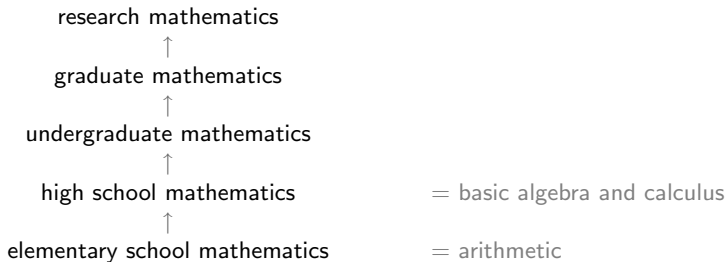
the mathematics spectrum of formal proof



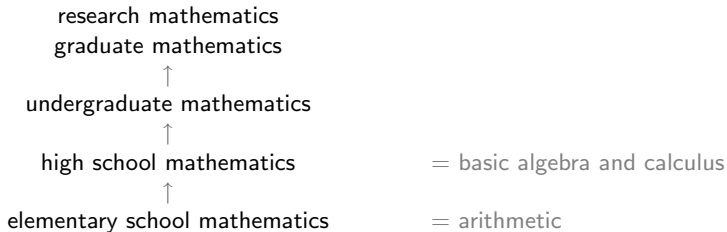
the mathematics spectrum of formal proof



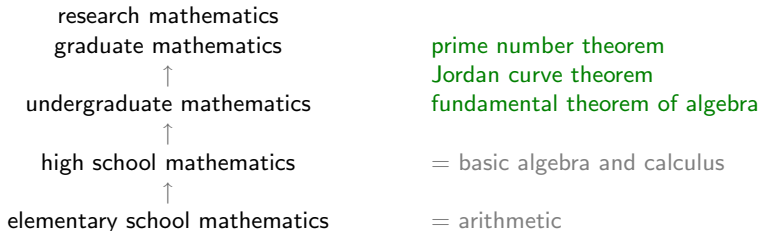
the mathematics spectrum of formal proof



the mathematics spectrum of formal proof



the mathematics spectrum of formal proof



the mathematics spectrum of formal proof

proofs with large computer calculations



research mathematics
graduate mathematics



undergraduate mathematics



high school mathematics



elementary school mathematics

four color theorem

prime number theorem
Jordan curve theorem
fundamental theorem of algebra

= basic algebra and calculus

= arithmetic

the mathematics spectrum of formal proof

recent solutions to famous open problems



proofs with large computer calculations



research mathematics
graduate mathematics



undergraduate mathematics



high school mathematics



elementary school mathematics

four color theorem

prime number theorem

Jordan curve theorem

fundamental theorem of algebra

= basic algebra and calculus

= arithmetic

the mathematics spectrum of formal proof

recent solutions to famous open problems

$P \neq NP?$



proofs with large computer calculations

four color theorem



research mathematics
graduate mathematics

prime number theorem

Jordan curve theorem



undergraduate mathematics

fundamental theorem of algebra



high school mathematics

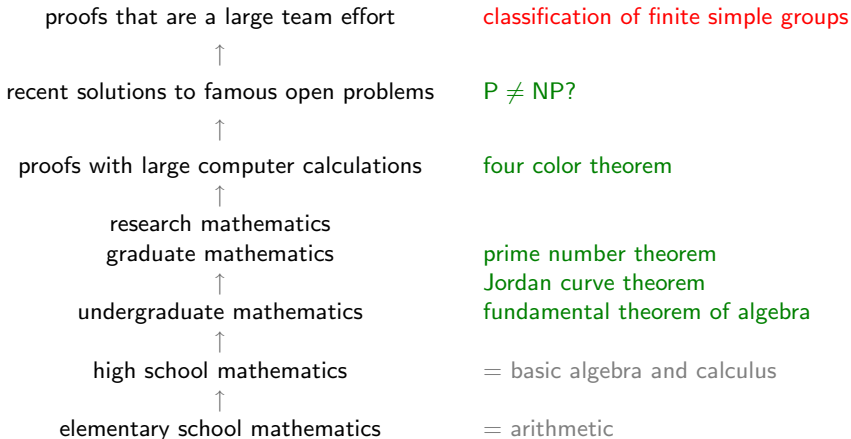
= basic algebra and calculus



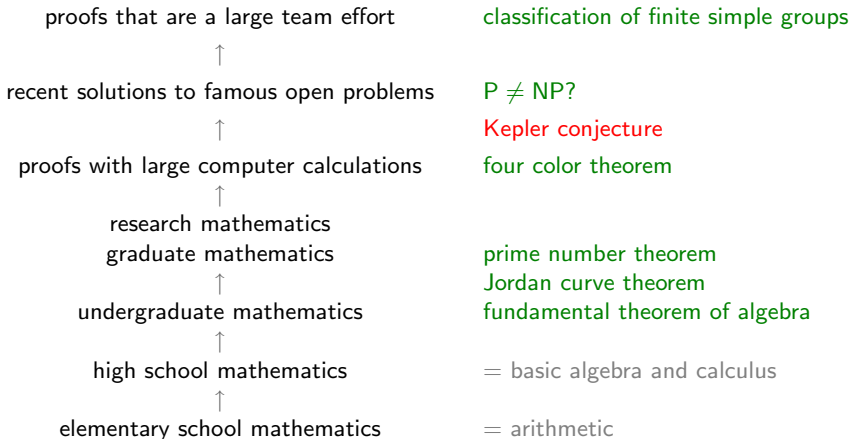
elementary school mathematics

= arithmetic

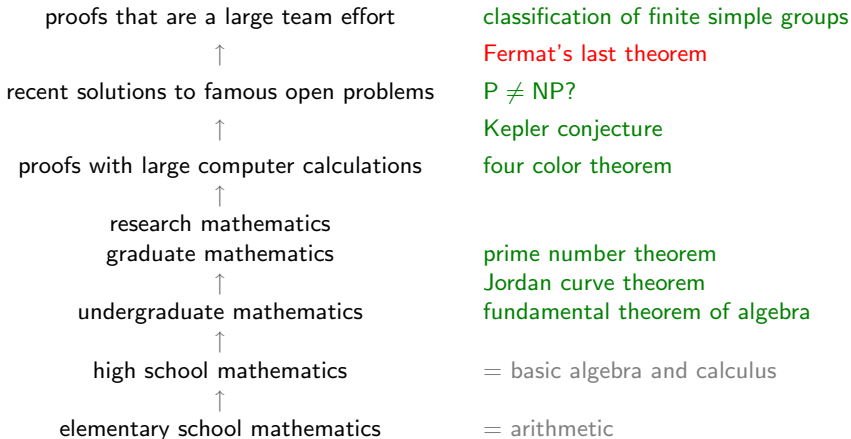
the mathematics spectrum of formal proof



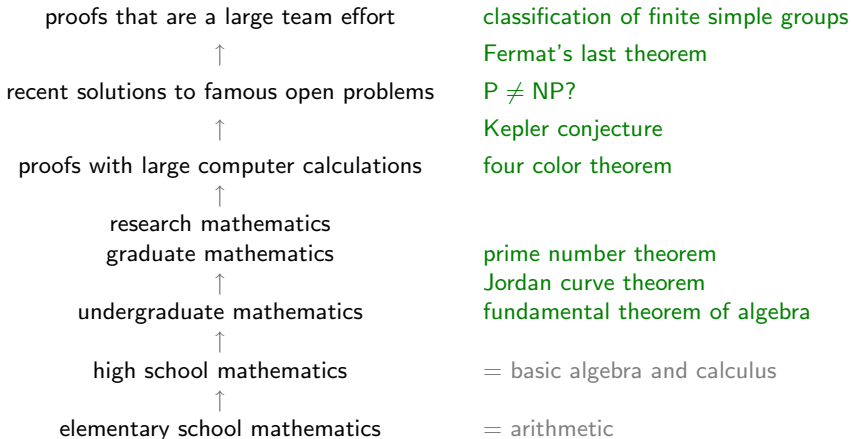
the mathematics spectrum of formal proof



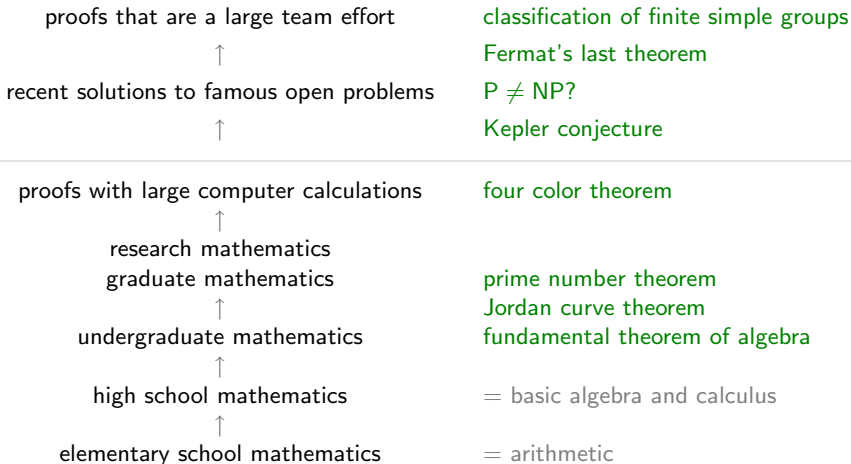
the mathematics spectrum of formal proof



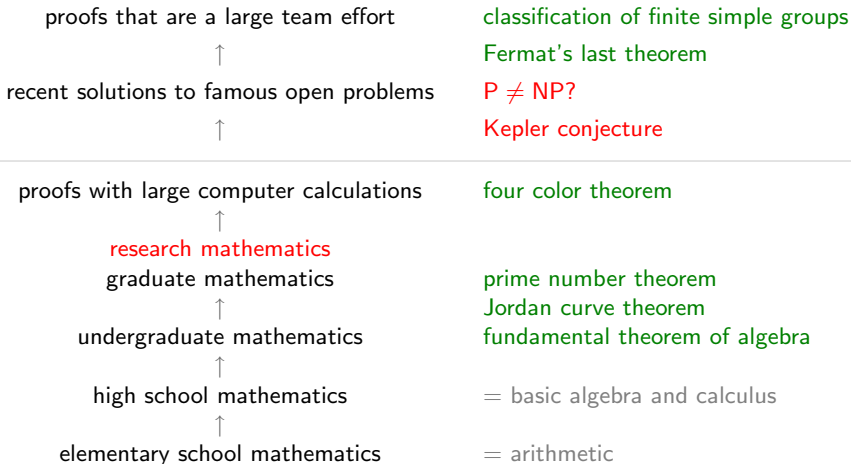
the mathematics spectrum of formal proof



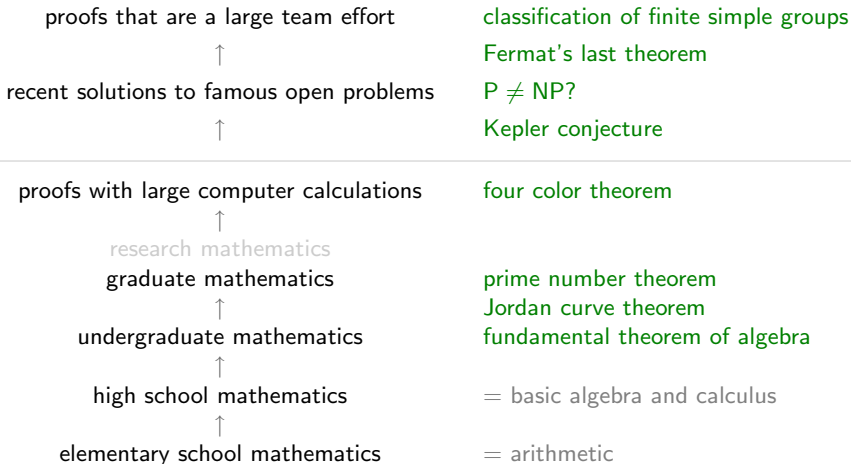
the mathematics spectrum of formal proof



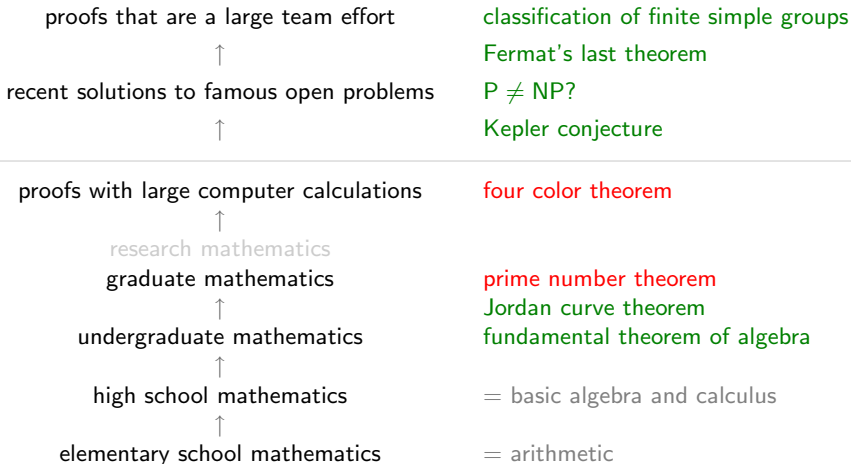
the mathematics spectrum of formal proof



the mathematics spectrum of formal proof



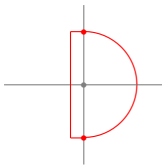
the mathematics spectrum of formal proof



formally proving the prime number theorem

John Harrison, HOL Light,  Intel, 2008

$$2\pi i F(w) = \int_{\Gamma} F(z+w) N^z \left(\frac{1}{z} + \frac{z}{R^2} \right) dz$$

*etcetera*

ALL_TAC] THEN

SUBGOAL_THEN



```
'((\z. f(w + z) * Cx(&N) cpow z * (Cx(&1) / z + z / Cx(R) pow 2))
  has_path_integral (Cx(&2) * Cx pi * ii * f(w))) (A ++ B)'
```

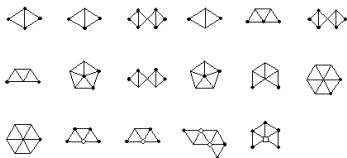
ASSUME_TAC THENL

[MP_TAC(ISPECL

etcetera



formally proving the four color theorem

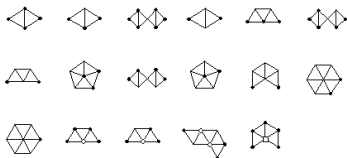
Georges Gonthier, Coq/ssreflect,  Microsoft +  INRIA, 2006



- ▶ Appel & Haken, 1976
assembly program
- ▶ Robertson, Sanders, Seymour & Thomas, 1997
C program
- ▶ Gonthier
purely functional (= no state) OCaml program

formally proving the four color theorem

Georges Gonthier, Coq/ssreflect,  Microsoft +  INRIA, 2006



- ▶ Appel & Haken, 1976
assembly program

- ▶ Robertson, Sanders, Seymour & Thomas, 1997
C program

- ▶ Gonthier
purely functional (= no state) OCaml → Coq program

proof assistants: the next generation

an email from Tobias Nipkow

Message-ID: <4785C81D.2090607@in.tum.de>

Date: Thu, 10 Jan 2008 08:24:13 +0100

From: Tobias Nipkow <nipkow@in.tum.de>

To: Freek Wiedijk <freek@cs.ru.nl>

Subject: Re: [Fwd: free ultrafilters]

[...]

> I personally hate the totality of the HOL logic. Don't
> you?

Occasionally I do. But mostly not.

The next generation of proof assistants will take it into account.

[...]

an email from Tobias Nipkow

Message-ID: <4785C81D.2090607@in.tum.de>

Date: Thu, 10 Jan 2008 08:24:13 +0100

From: Tobias Nipkow <nipkow@in.tum.de>

To: Freek Wiedijk <freek@cs.ru.nl>

Subject: Re: [Fwd: free ultrafilters]

[...]

me



> I personally hate the totality of the HOL logic. Don't
> you?

Occasionally I do. But mostly not.

The next generation of proof assistants will take it into account.

[...]

an email from Tobias Nipkow

Message-ID: <4785C81D.2090607@in.tum.de>

Date: Thu, 10 Jan 2008 08:24:13 +0100

From: Tobias Nipkow <nipkow@in.tum.de>

To: Freek Wiedijk <freek@cs.ru.nl>

Subject: Re: [Fwd: free ultrafilters]

[...]

> I personally hate the totality of the HOL logic. Don't
> you?

Occasionally I do. But mostly not.

The next generation of proof assistants will take it into account.

[...]

me



Tobias



an email from Tobias Nipkow

Message-ID: <4785C81D.2090607@in.tum.de>

Date: Thu, 10 Jan 2008 08:24:13 +0100

From: Tobias Nipkow <nipkow@in.tum.de>

To: Freek Wiedijk <freek@cs.ru.nl>

Subject: Re: [Fwd: free ultrafilters]

[...]

> I personally hate the totality of the HOL logic. Don't
> you?

Occasionally I do. But mostly not.

The **next generation of proof assistants** will take it into account.

[...]

me



Tobias



should the next generation of proof assistants
be based on ZFC set theory?

should the next generation of proof assistants
be based on ZFC set theory?

ACL2
B method
PVS
HOL
Isabelle
Coq
Mizar
Twelf
Metamath

should the next generation of proof assistants
be based on ZFC set theory?

ACL2
B method
PVS
HOL
Isabelle
Coq
Mizar
Twelf
Metamath

should the next generation of proof assistants
be based on ZFC set theory?

ACL2
B method
PVS
HOL
Isabelle
Coq
Mizar
Twelf
Metamath

should the next generation of proof assistants
be based on ZFC set theory?

ACL2
B method
PVS
HOL
Isabelle/ZF
Coq
Mizar
Twelf
Metamath

foundations for proof assistants

- ▶ **set theory** (Mizar, Isabelle/ZF, Metamath, B method)
Zermelo-Fraenkel axioms + axiom of Choice
- ▶ **type theory** (Coq, Twelf)
Curry-Howard isomorphism
- ▶ **higher order logic** (HOL, Isabelle/HOL, PVS)
- ▶ **primitive recursive arithmetic** (ACL2)

foundations for proof assistants

- ▶ **set theory** (Mizar, Isabelle/ZF, Metamath, B method)
Zermelo-Fraenkel axioms + axiom of Choice
untyped, not computational, classical, canonical
- ▶ **type theory** (Coq, Twelf)
Curry-Howard isomorphism
- ▶ **higher order logic** (HOL, Isabelle/HOL, PVS)
- ▶ **primitive recursive arithmetic** (ACL2)

foundations for proof assistants

- ▶ **set theory** (Mizar, Isabelle/ZF, Metamath, B method)
Zermelo-Fraenkel axioms + axiom of Choice
untyped, not computational, classical, canonical
- ▶ **type theory** (Coq, Twelf)
Curry-Howard isomorphism
typed, computational, intuitionistic, many variations
- ▶ **higher order logic** (HOL, Isabelle/HOL, PVS)
- ▶ **primitive recursive arithmetic** (ACL2)

foundations for proof assistants

- ▶ **set theory** (Mizar, Isabelle/ZF, Metamath, B method)
Zermelo-Fraenkel axioms + axiom of Choice
untyped, not computational, classical, canonical
- ▶ **type theory** (Coq, Twelf)
Curry-Howard isomorphism, predicative? intensional?
typed, computational, intuitionistic, many variations
- ▶ **higher order logic** (HOL, Isabelle/HOL, PVS)
- ▶ **primitive recursive arithmetic** (ACL2)

foundations for proof assistants

- ▶ **set theory** (Mizar, Isabelle/ZF, Metamath, B method)
Zermelo-Fraenkel axioms + axiom of Choice
untyped, not computational, classical, canonical
- ▶ **type theory** (Coq, Twelf)
Curry-Howard isomorphism, predicative? intensional?
typed, computational, intuitionistic, many variations
- ▶ **higher order logic** (HOL, Isabelle/HOL, PVS)
typed, not computational, classical, canonical
- ▶ **primitive recursive arithmetic** (ACL2)

foundations for proof assistants

- ▶ **set theory** (Mizar, Isabelle/ZF, Metamath, B method)
Zermelo-Fraenkel axioms + axiom of Choice
untyped, not computational, classical, canonical
- ▶ **type theory** (Coq, Twelf)
Curry-Howard isomorphism, predicative? intensional?
typed, computational, intuitionistic, many variations
- ▶ **higher order logic** (HOL, Isabelle/HOL, PVS)
typed, not computational, classical, canonical
- ▶ **primitive recursive arithmetic** (ACL2)
untyped, computational, classical, canonical

foundations for proof assistants

- ▶ **set theory** (Mizar, Isabelle/ZF, Metamath, B method)
Zermelo-Fraenkel axioms + axiom of Choice
untyped, not computational, classical, canonical
- ▶ **type theory** (Coq, Twelf)
Curry-Howard isomorphism, predicative? intensional?
typed, computational, intuitionistic, many variations
as expressive as set theory
- ▶ **higher order logic** (HOL, Isabelle/HOL, PVS)
typed, not computational, classical, canonical
- ▶ **primitive recursive arithmetic** (ACL2)
untyped, computational, classical, canonical

foundations for proof assistants

- ▶ **set theory** (Mizar, Isabelle/ZF, Metamath, B method)
Zermelo-Fraenkel axioms + axiom of Choice
untyped, not computational, classical, canonical
- ▶ **type theory** (Coq, Twelf)
Curry-Howard isomorphism, predicative? intensional?
typed, computational, intuitionistic, many variations
as expressive as set theory
- ▶ **higher order logic** (HOL, Isabelle/HOL, PVS)
typed, not computational, classical, canonical
less expressive
- ▶ **primitive recursive arithmetic** (ACL2)
untyped, computational, classical, canonical

foundations for proof assistants

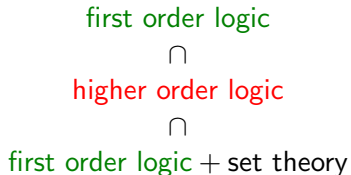
- ▶ **set theory** (Mizar, Isabelle/ZF, Metamath, B method)
Zermelo-Fraenkel axioms + axiom of Choice
untyped, not computational, classical, canonical
- ▶ **type theory** (Coq, Twelf)
Curry-Howard isomorphism, predicative? intensional?
typed, computational, intuitionistic, many variations
as expressive as set theory
- ▶ **higher order logic** (HOL, Isabelle/HOL, PVS)
typed, not computational, classical, canonical
less expressive
- ▶ **primitive recursive arithmetic** (ACL2)
untyped, computational, classical, canonical
even less expressive

first order logic versus higher order logic

'canonical logic' = classical first order predicate logic with equality

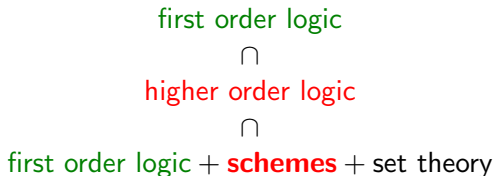
first order logic versus higher order logic

'canonical logic' = classical first order predicate logic with equality



first order logic versus higher order logic

'canonical logic' = classical first order predicate logic with equality



first order logic versus higher order logic

'canonical logic' = classical first order predicate logic with equality

first order logic
 \cap
 higher order logic
 \cap
 first order logic + **schemes** + set theory

binders

$$\{x \in A \mid P(x)\}$$

$$\sum_{i=1}^n a_i$$

$$\lim_{x \rightarrow a} f(x)$$

$$\int f(x) dx$$

||

should the next generation of proof assistants
have an advanced type system?

should the next generation of proof assistants
have an advanced type system?

ACL2
B method
PVS
HOL
Isabelle
Coq
Mizar
Twelf
Metamath

soft types

- ▶ **no types** (Metamath)
- ▶ **hard types** (PVS, HOL, Isabelle, Coq, Twelf)
types are part of the foundation
- ▶ **soft types** (ACL2, B method, Mizar)
types are a layer on top of an untyped foundation

soft types

- ▶ **no types** (Metamath)
- ▶ **hard types** (PVS, HOL, Isabelle, Coq, Twelf)
types are part of the foundation
- ▶ **soft types** (ACL2, B method, Mizar)
types are a layer on top of an untyped foundation

'types as predicates'

type inference = automated predicate proving

$\forall x : A. P(x)$ is syntax for $\forall x. A(x) \Rightarrow P(x)$

soft types

- ▶ **no types** (Metamath)
- ▶ **hard types** (PVS, HOL, Isabelle, Coq, Twelf)
types are part of the foundation
- ▶ **soft types** (ACL2, B method, Mizar)
types are a layer on top of an untyped foundation

'types as predicates'

type inference = automated predicate proving

$\forall x : A(y, \dots). P(x)$ is syntax for $\forall x. A(x, y, \dots) \Rightarrow P(x)$

natural interpretation for **dependent types**

soft types

- ▶ **no types** (Metamath)
- ▶ **hard types** (PVS, HOL, Isabelle, Coq, Twelf)
types are part of the foundation
- ▶ **soft types** (ACL2, B method, Mizar)
types are a layer on top of an untyped foundation

'types as predicates'

type inference = automated predicate proving

$\forall x : A(y, \dots). P(x)$ is syntax for $\forall x. A(x, y, \dots) \Rightarrow P(x)$

natural interpretation for **dependent types**

no natural interpretation for **function types** $A \rightarrow B$

dependent types and empty types

► **dependent types**

- element of a given set
- element of a given algebraic structure
- array of a given length
- normal form of a given lambda term
- vector space of a given dimension
- field extension of a given field by a given degree

dependent types and empty types

► dependent types

element of a given set

element of a **given algebraic structure**

array of a given length

normal form of a given lambda term

vector space of a given dimension

field extension of a given field by a given degree

essential for **implicit arguments** in notations

$x + y$ for $x +_G y$ when x, y are elements of a group G

dependent types and empty types

► dependent types

element of a given set

element of a given algebraic structure

array of a given length

normal form of a given lambda term

vector space of a given dimension

field extension of a given field by a given degree

essential for implicit arguments in notations

$x + y$ for $x +_G y$ when x, y are elements of a group G

► empty types

unavoidable with natural definitions of dependent types

dependent types and empty types

► **dependent types**

element of a given set

element of a given algebraic structure

array of a given length

normal form of a given lambda term

vector space of a given dimension

field extension of a given field by a given degree

essential for implicit arguments in notations

$x + y$ for $x +_G y$ when x, y are elements of a group G

► **empty types**

unavoidable with natural definitions of dependent types

set \leftrightarrow class \leftrightarrow type

should the next generation of proof assistants
take partiality seriously?

should the next generation of proof assistants
take partiality seriously?

ACL2
B method
PVS
HOL
Isabelle
Coq
Mizar
Twelf
Metamath

the value of $\frac{1}{0}$

- ▶ $\frac{1}{0} = 0$ **is provable** (ACL2, HOL, Isabelle/HOL, Mizar)
- ▶ $\frac{1}{0} = 0$ **is disprovable** (Metamath)

- ▶ $\frac{1}{0} = 0$ **is neither provable nor disprovable** (Coq)
- ▶ $\frac{1}{0} = 0$ **is illegal** (ACL2/gold, B method, PVS, Coq/CoRN, Twelf)

the value of $\frac{1}{0}$

- ▶ $\frac{1}{0} = 0$ **is provable** (ACL2, HOL, Isabelle/HOL, Mizar)
- ▶ $\frac{1}{0} = 0$ **is disprovable** (Metamath, **IMPS**)
proof $\frac{1}{0}$ is undefined, 0 is defined **qed**
- ▶ $\frac{1}{0} = 0$ **is neither provable nor disprovable** (Coq)
- ▶ $\frac{1}{0} = 0$ **is illegal** (ACL2/gold, B method, PVS, Coq/CoRN, Twelf)

the value of $\frac{1}{0}$

▶ $\frac{1}{0} = 0$ **is provable** (ACL2, HOL, Isabelle/HOL, Mizar)

▶ $\frac{1}{0} = 0$ **is disprovable** (Metamath, **IMPS**)

proof $\frac{1}{0}$ is undefined, 0 is defined **qed**

$$\frac{1}{0} = \frac{1}{0} ?$$

▶ $\frac{1}{0} = 0$ **is neither provable nor disprovable** (Coq)

▶ $\frac{1}{0} = 0$ **is illegal** (ACL2/gold, B method, PVS, Coq/CoRN, Twelf)

the value of $\frac{1}{0}$

▶ $\frac{1}{0} = 0$ **is provable** (ACL2, HOL, Isabelle/HOL, Mizar)

▶ $\frac{1}{0} = 0$ **is disprovable** (Metamath, **IMPS**)

proof $\frac{1}{0}$ is undefined, 0 is defined **qed**

$$\frac{1}{0} = \frac{1}{0} ?$$

▶ $\frac{1}{0} = 0$ **is neither provable nor disprovable** (Coq)

▶ $\frac{1}{0} = 0$ **is illegal** (ACL2/gold, B method, PVS, Coq/CoRN, Twelf)

the value of $\frac{1}{0}$

▶ $\frac{1}{0} = 0$ **is provable** (ACL2, HOL, Isabelle/HOL, Mizar)

▶ $\frac{1}{0} = 0$ **is disprovable** (Metamath, IMPS)

proof $\frac{1}{0}$ is undefined, 0 is defined qed

$$\frac{1}{0} = \frac{1}{0} ?$$

▶ $\frac{1}{0} = 0$ **is neither provable nor disprovable** (Coq)

▶ $\frac{1}{0} = 0$ **is illegal** (ACL2/gold, B method, PVS, Coq/CoRN, Twelf)

$$\frac{1}{x} = y$$

$\text{div}(1, x, \langle \text{proof object for } x \neq 0 \rangle) = y$

the value of $\frac{1}{0}$

▶ $\frac{1}{0} = 0$ **is provable** (ACL2, HOL, Isabelle/HOL, Mizar)

▶ $\frac{1}{0} = 0$ **is disprovable** (Metamath, IMPS)

proof $\frac{1}{0}$ is undefined, 0 is defined qed

$$\frac{1}{0} = \frac{1}{0} ?$$

▶ $\frac{1}{0} = 0$ **is neither provable nor disprovable** (Coq)

▶ $\frac{1}{0} = 0$ **is illegal** (ACL2/gold, B method, PVS, Coq/CoRN, Twelf)

$$\frac{1}{0} = 0$$

$\text{div}(1, 0, \langle \text{proof object for } 0 \neq 0 \rangle) = 0$

the value of $\frac{1}{0}$

▶ $\frac{1}{0} = 0$ **is provable** (ACL2, HOL, Isabelle/HOL, Mizar)

▶ $\frac{1}{0} = 0$ **is disprovable** (Metamath, IMPS)

proof $\frac{1}{0}$ is undefined, 0 is defined qed

$$\frac{1}{0} = \frac{1}{0} ?$$

▶ $\frac{1}{0} = 0$ **is neither provable nor disprovable** (Coq)

▶ $\frac{1}{0} = 0$ **is illegal** (ACL2/gold, B method, PVS, Coq/CoRN, Twelf)

$$\frac{1}{0} = 0$$

$\text{div}(1, 0, \langle \text{proof object for } 0 \neq 0 \rangle) = 0$

the value of $\frac{1}{0}$

▶ $\frac{1}{0} = 0$ **is provable** (ACL2, HOL, Isabelle/HOL, Mizar)

▶ $\frac{1}{0} = 0$ **is disprovable** (Metamath, IMPS)

proof $\frac{1}{0}$ is undefined, 0 is defined qed

$$\frac{1}{0} = \frac{1}{0} ?$$

▶ $\frac{1}{0} = 0$ **is neither provable nor disprovable** (Coq)

▶ $\frac{1}{0} = 0$ **is illegal** (ACL2/gold, B method, PVS, Coq/CoRN, Twelf)

$$\frac{1}{0} = 0$$

$\text{div}(1, 0, \langle \text{proof object for } 0 \neq 0 \rangle) = 0$

definedness conditions

each function $f(x_1, \dots, x_n)$ has a **domain predicate** $D_f(x_1, \dots, x_n)$

$$D_{\text{div}}(x, y) \Leftrightarrow (y \neq 0)$$

each formula ϕ has a **definedness condition** $\Delta(\phi)$

$$\Delta\left(\frac{1}{0} = 0\right) \Leftrightarrow D_{\text{div}}(1, 0) \Leftrightarrow (0 \neq 0) \Leftrightarrow \perp$$

definedness conditions

each function $f(x_1, \dots, x_n)$ has a **domain predicate** $D_f(x_1, \dots, x_n)$

$$D_{\text{div}}(x, y) \Leftrightarrow (y \neq 0)$$

each formula ϕ has a **definedness condition** $\Delta(\phi)$

$$\Delta\left(\frac{1}{0} = 0\right) \Leftrightarrow D_{\text{div}}(1, 0) \Leftrightarrow (0 \neq 0) \Leftrightarrow \perp$$

$$\Delta(\forall x \in \mathbb{R}. x \neq 0 \Rightarrow \frac{1}{x} \neq 0) ?$$

definedness conditions

each function $f(x_1, \dots, x_n)$ has a **domain predicate** $D_f(x_1, \dots, x_n)$

$$D_{\text{div}}(x, y) \Leftrightarrow (y \neq 0)$$

each formula ϕ has a **definedness condition** $\Delta(\phi)$

$$\Delta\left(\frac{1}{0} = 0\right) \Leftrightarrow D_{\text{div}}(1, 0) \Leftrightarrow (0 \neq 0) \Leftrightarrow \perp$$

$$\Delta(\forall x \in \mathbb{R}. x \neq 0 \Rightarrow \frac{1}{x} \neq 0)$$

$$\Delta(\phi \Rightarrow \psi) \Leftrightarrow (\Delta(\phi) \wedge (\phi \Rightarrow \Delta(\psi)))$$

definedness conditions

each function $f(x_1, \dots, x_n)$ has a **domain predicate** $D_f(x_1, \dots, x_n)$

$$D_{\text{div}}(x, y) \Leftrightarrow (y \neq 0)$$

each formula ϕ has a **definedness condition** $\Delta(\phi)$

$$\Delta\left(\frac{1}{0} = 0\right) \Leftrightarrow D_{\text{div}}(1, 0) \Leftrightarrow (0 \neq 0) \Leftrightarrow \perp$$

$$\Delta(\forall x \in \mathbb{R}. x \neq 0 \Rightarrow \frac{1}{x} \neq 0)$$

$$\Delta(\phi \Rightarrow \psi) \Leftrightarrow (\Delta(\phi) \wedge (\phi \Rightarrow \Delta(\psi)))$$

$$\Delta(\phi \wedge \psi) \Leftrightarrow (\Delta(\phi) \wedge (\phi \Rightarrow \Delta(\psi)))$$

definedness conditions

each function $f(x_1, \dots, x_n)$ has a **domain predicate** $D_f(x_1, \dots, x_n)$

$$D_{\text{div}}(x, y) \Leftrightarrow (y \neq 0)$$

each formula ϕ has a **definedness condition** $\Delta(\phi)$

$$\Delta\left(\frac{1}{0} = 0\right) \Leftrightarrow D_{\text{div}}(1, 0) \Leftrightarrow (0 \neq 0) \Leftrightarrow \perp$$

$$\Delta(\forall x \in \mathbb{R}. x \neq 0 \Rightarrow \frac{1}{x} \neq 0)$$

$$\Delta(\phi \Rightarrow \psi) \Leftrightarrow (\Delta(\phi) \wedge (\phi \Rightarrow \Delta(\psi)))$$

$$\Delta(\phi \wedge \psi) \Leftrightarrow (\Delta(\phi) \wedge (\phi \Rightarrow \Delta(\psi)))$$

Δ does **not** respect logical equivalence

$$\Delta(\phi \wedge \psi) \not\equiv \Delta(\psi \wedge \phi)$$

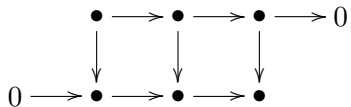
should the next generation of proof assistants
take category theory seriously?

IV

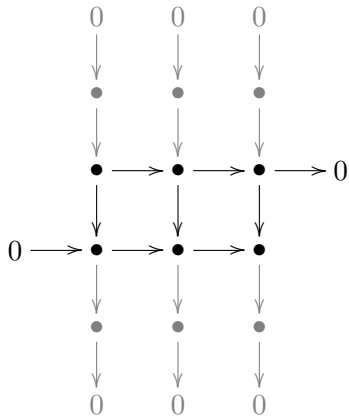
should the next generation of proof assistants
take category theory seriously?

ACL2
B method
PVS
HOL
Isabelle
Coq
Mizar
Twelf
Metamath

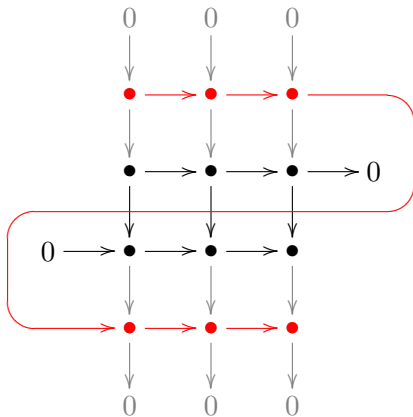
the snake lemma and the category of Abelian groups



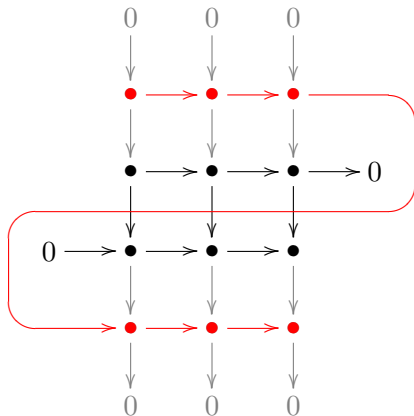
the snake lemma and the category of Abelian groups



the snake lemma and the category of Abelian groups



the snake lemma and the category of Abelian groups



I don't like universes!

Vladimir Voevodsky and the formalization of the real numbers

Vladimir Voevodsky,  Institute for Advanced Study
Fields medal in 2002

Vladimir Voevodsky and the formalization of the real numbers

Vladimir Voevodsky,  Institute for Advanced Study
Fields medal in 2002

homotopy type theory, 2006

Vladimir Voevodsky and the formalization of the real numbers

Vladimir Voevodsky,  Institute for Advanced Study
Fields medal in 2002

homotopy type theory, 2006

I will speak about type systems. It is difficult for a mathematician since a type system is not a mathematical notion.

Vladimir Voevodsky and the formalization of the real numbers

Vladimir Voevodsky,  Institute for Advanced Study
Fields medal in 2002

homotopy type theory, 2006

I will speak about type systems. It is difficult for a mathematician since a type system is not a mathematical notion.

$\mathbb{R} =$

set of Dedekind cuts?

set of equivalence classes of Cauchy sequences?

Vladimir Voevodsky and the formalization of the real numbers

Vladimir Voevodsky,  Institute for Advanced Study
Fields medal in 2002

homotopy type theory, 2006

I will speak about type systems. It is difficult for a mathematician since a type system is not a mathematical notion.

$\mathbb{R} =$

set of Dedekind cuts?

set of equivalence classes of Cauchy sequences?

set of equivalence classes of pairs of *positive* real numbers?

Vladimir Voevodsky and the formalization of the real numbers

Vladimir Voevodsky,  Institute for Advanced Study
Fields medal in 2002

homotopy type theory, 2006

I will speak about type systems. It is difficult for a mathematician since a type system is not a mathematical notion.

$\mathbb{R} =$

set of Dedekind cuts?

set of equivalence classes of Cauchy sequences?

set of equivalence classes of pairs of *positive* real numbers?

any field isomorphic to the real numbers?

Vladimir Voevodsky and the formalization of the real numbers

Vladimir Voevodsky,  Institute for Advanced Study
Fields medal in 2002

homotopy type theory, 2006

I will speak about type systems. It is difficult for a mathematician since a type system is not a mathematical notion.

$\mathbb{R} =$

set of Dedekind cuts?

set of equivalence classes of Cauchy sequences?

set of equivalence classes of pairs of *positive* real numbers?

any field isomorphic to the real numbers?

any **set** of the cardinality of the real numbers?

Vladimir Voevodsky and the formalization of the real numbers

Vladimir Voevodsky,  Institute for Advanced Study
Fields medal in 2002

homotopy type theory, 2006

I will speak about type systems. It is difficult for a mathematician since a type system is not a mathematical notion.

$\mathbb{R} =$

set of Dedekind cuts?

set of equivalence classes of Cauchy sequences?

set of equivalence classes of pairs of *positive* real numbers?

any field isomorphic to the real numbers?

any **set** of the cardinality of the real numbers?

abstract datatypes in mathematics?

Vladimir Voevodsky and the formalization of the real numbers

Vladimir Voevodsky,  Institute for Advanced Study
Fields medal in 2002

homotopy type theory, 2006

I will speak about type systems. It is difficult for a mathematician since a type system is not a mathematical notion.

$\mathbb{R} =$

set of Dedekind cuts?

set of equivalence classes of Cauchy sequences?

set of equivalence classes of pairs of *positive* real numbers?

any field isomorphic to the real numbers?

any **set** of the cardinality of the real numbers?

abstract datatypes in mathematics?

hardwire 'up to isomorphism' in the logical foundations?

V

should the next generation of proof assistants
be based on a logical framework?

should the next generation of proof assistants
be based on a logical framework?

ACL2
B method
PVS
HOL
Isabelle
Coq
Mizar
Twelf
Metamath


logical frameworks

- ▶ **Twelf** = LF
minimal type theory
- ▶ **Isabelle**/Pure
minimal higher order logic
- ▶ **Metamath**
minimal string manipulation


logical frameworks

- ▶ **Twelf** = LF
minimal type theory
- ▶ **Isabelle**/Pure
minimal higher order logic
- ▶ **Metamath**
minimal string manipulation
not HOAS: not invariant under variable renaming

logical frameworks

- ▶ **Twelf** = LF
minimal type theory
- ▶ **Isabelle**/Pure
minimal higher order logic
- ▶ **Metamath**
minimal string manipulation
not HOAS: not invariant under variable renaming
- ▶ **Automath** = AUT-SL = $\Delta\Lambda$
N.G. de Bruijn,  Eindhoven University of Technology, 1987
very similar to LF but more elegant

logical frameworks

- ▶ **Twelf** = LF
minimal type theory
- ▶ **Isabelle**/Pure
minimal higher order logic
- ▶ **Metamath**
minimal string manipulation
not HOAS: not invariant under variable renaming
- ▶ **Automath** = AUT-SL = $\Delta\Lambda$
N.G. de Bruijn,  Eindhoven University of Technology, 1987
very similar to LF but more elegant
- ▶ $\Delta\Lambda$ with a **bounded conversion rule**?

reasons for using a logical framework

- ▶ **varying the logic**

 - taking Gödel's incompleteness theorem seriously

 - tracking what is used in a proof

 - classical logic?

 - K axiom about equality of equality proofs?

 - axiom of choice?

 - large cardinal axioms?

 - Hilbert's ϵ choice operator?

 - universes?

reasons for using a logical framework

- ▶ **varying the logic**

- taking Gödel's incompleteness theorem seriously
 - tracking what is used in a proof

- classical logic?

- K axiom about equality of equality proofs?

- axiom of choice?

- large cardinal axioms?

- Hilbert's ϵ choice operator?

- universes?

- ▶ **simplifying the logical kernel**

- DNA of formal mathematics**

reasons for using a logical framework

► **varying the logic**

taking Gödel's incompleteness theorem seriously
tracking what is used in a proof

classical logic?

K axiom about equality of equality proofs?

axiom of choice?

large cardinal axioms?

Hilbert's ϵ choice operator?

universes?

► **simplifying the logical kernel**

DNA of formal mathematics

$\Delta\Delta$ 'term' = directed graph with four kinds of nodes

out-degrees: 2, 2, 1, 0

should the next generation of proof assistants
have a self-verified kernel?

should the next generation of proof assistants
have a self-verified kernel?

ACL2

B method

PVS

HOL

Isabelle

Coq

Mizar

Twelf

Metamath

state of the art in self-verification

► **Coq in Coq = Coc**

Bruno Barras,  Université Paris 7, 1999

version of Coq kernel as a Coq program, extracted to OCaml
not close to Coq source, can check proofs from real Coq

state of the art in self-verification

▶ **Coq in Coq = Coc**

Bruno Barras,  Université Paris 7, 1999

version of Coq kernel as a Coq program, extracted to OCaml
not close to Coq source, can check proofs from real Coq

▶ **HOL in HOL**

John Harrison,  Intel, 2006

simplified HOL kernel as a HOL program
very close to HOL source, cannot check HOL proofs

state of the art in self-verification

▶ **Coq in Coq = Coc**

Bruno Barras,  Université Paris 7, 1999

version of Coq kernel as a Coq program, extracted to OCaml
not close to Coq source, can check proofs from real Coq

▶ **HOL in HOL**

John Harrison,  Intel, 2006

simplified HOL kernel as a HOL program
very close to HOL source, cannot check HOL proofs

▶ **ACL2 in ACL2 = Milawa**

Jared Davis,  University of Texas, 2009

various approximations to ACL2 as an ACL2 program
not close to ACL2 source, can check proofs from real ACL2

the philosophy of certainty

- ▶ reliability of hardware and software infrastructure?

the philosophy of certainty

- ▶ reliability of hardware and software infrastructure?
- ▶ how can a system validate itself?
if it is incorrect it can falsely claim to be correct

the philosophy of certainty

- ▶ reliability of hardware and software infrastructure?
- ▶ how can a system validate itself?
if it is incorrect it can falsely claim to be correct
- ▶ Gödel's second incompleteness theorem?
no consistent logical system can prove its own consistency

the philosophy of certainty

- ▶ reliability of hardware and software infrastructure?
 - ▶ how can a system validate itself?
if it is incorrect it can falsely claim to be correct
 - ▶ Gödel's second incompleteness theorem?
no consistent logical system can prove its own consistency
- not: **system cannot prove false**
but: **system implements logic**

the philosophy of certainty

- ▶ reliability of hardware and software infrastructure?
- ▶ how can a system validate itself?
if it is incorrect it can falsely claim to be correct
- ▶ Gödel's second incompleteness theorem?
no consistent logical system can prove its own consistency

not: **system cannot prove false**

but: **system implements logic**

just one additional line

logic cannot prove false \vdash system cannot prove false

the philosophy of certainty

- ▶ reliability of hardware and software infrastructure?
- ▶ how can a system validate itself?
if it is incorrect it can falsely claim to be correct
- ▶ Gödel's second incompleteness theorem?
no consistent logical system can prove its own consistency

not: **system cannot prove false**

but: **system implements logic**

just one additional line

logic has a model \vdash system cannot prove false

the philosophy of certainty

- ▶ reliability of hardware and software infrastructure?
- ▶ how can a system validate itself?
if it is incorrect it can falsely claim to be correct
- ▶ Gödel's second incompleteness theorem?
no consistent logical system can prove its own consistency

not: **system cannot prove false**

but: **system implements logic**

just one additional line

inaccessible cardinal axiom \vdash system cannot prove false

VII

should the next generation of proof assistants
be programmed in itself?

VII

should the next generation of proof assistants
be programmed in itself?

ACL2
B method
PVS
HOL
Isabelle
Coq
Mizar
Twelf
Metamath

languages in proof assistants

- ▶ **implementation language**
language the implementer uses to implement the system

languages in proof assistants

- ▶ **implementation language**
language the implementer uses to implement the system

- ▶ **scripting language**
language the user uses to automate proofs in the system

languages in proof assistants

- ▶ **implementation language**
language the implementer uses to implement the system
- ▶ **scripting language**
language the user uses to automate proofs in the system
- ▶ **mathematical language**
language the user uses to write mathematics

languages in proof assistants

- ▶ **implementation language**
language the implementer uses to implement the system
- ▶ **scripting language**
language the user uses to automate proofs in the system
- ▶ **mathematical language**
language the user uses to write mathematics
... and mathematical algorithms

languages in proof assistants

- ▶ **implementation language**
language the implementer uses to implement the system
Coq: OCaml
- ▶ **scripting language**
language the user uses to automate proofs in the system
- ▶ **mathematical language**
language the user uses to write mathematics
... and mathematical algorithms

languages in proof assistants

- ▶ **implementation language**
language the implementer uses to implement the system
Coq: OCaml
- ▶ **scripting language**
language the user uses to automate proofs in the system
Coq: Ltac
- ▶ **mathematical language**
language the user uses to write mathematics
... and mathematical algorithms

languages in proof assistants

- ▶ **implementation language**

language the implementer uses to implement the system

Coq: OCaml

- ▶ **scripting language**

language the user uses to automate proofs in the system

Coq: Ltac

- ▶ **mathematical language**

language the user uses to write mathematics
... and mathematical algorithms

Coq: Gallina

languages in proof assistants

- ▶ **implementation language**
language the implementer uses to implement the system
Coq: OCaml
- ▶ **scripting language**
language the user uses to automate proofs in the system
Coq: Ltac
- ▶ **mathematical language**
language the user uses to write mathematics
... and mathematical algorithms
Coq: Gallina

OCaml, Ltac, Gallina: three very similar programming languages

languages in proof assistants

- ▶ **implementation language**
language the implementer uses to implement the system
Coq: OCaml
- ▶ **scripting language**
language the user uses to automate proofs in the system
Coq: Ltac
- ▶ **mathematical language**
language the user uses to write mathematics
... and mathematical algorithms
Coq: Gallina, Program

OCaml, Ltac, Gallina, Program: *four* similar programming languages

programming language versus mathematical language

► **type theory:**

mathematical language

\cap

programming language

mathematical functions = *terminating* programs

programming language versus mathematical language

► **type theory:**

mathematical language
 \cap
programming language

mathematical functions = *terminating* programs

► **better:**

programming language
 \cap
mathematical language

programs = *executable* function definitions

the mathematical function versus the executable object

all actual computers are finite state machines

the mathematical function versus the executable object

all actual computers are finite state machines

```
let rec fac n = if n=0 then 1 else n*(fac(n-1));;
```

the mathematical function versus the executable object

all actual computers are finite state machines

```
let rec fac n = if n=0 then 1 else n*(fac(n-1));;
```

mathematical function



$\text{fac} : \mathbb{Z}_{\geq 0} \rightarrow \mathbb{Z}$

&

theorem about
the definition of the
mathematical function

the mathematical function versus the executable object

all actual computers are finite state machines

```
let rec fac n = if n=0 then 1 else n*(fac(n-1));;
```

mathematical function

$\text{fac} : \mathbb{Z}_{\geq 0} \rightarrow \mathbb{Z}$

&

theorem about
the definition of the
mathematical function

executable object

```
2 ACCO
3 BNEQ 0, 9
6 CONST1
7 RETURN 1
9 ACCO
10 OFFSETINT -1
12 PUSHOFFSETCLOSURE0
13 APPLY1
14 PUSHACC1
15 MULINT
16 RETURN 1
```

the mathematical function versus the executable object

all actual computers are finite state machines

```
let rec fac n = if n=0 then 1 else n*(fac(n-1));;
```

mathematical function

$\text{fac} : \mathbb{Z}_{\geq 0} \rightarrow \mathbb{Z}$

&

*theorem about
the definition of the
mathematical function*

executable object

```
2 ACCO
3 BNEQ 0, 9
6 CONST1
7 RETURN 1
9 ACCO
10 OFFSETINT -1
12 PUSHOFFSETCLOSURE0
13 APPLY1
14 PUSHACC1
15 MULINT
16 RETURN 1
```

theorem about the relation between the two

VIII

should the next generation of proof assistants
be competitive with commercial computer algebra?

VIII

should the next generation of proof assistants
be competitive with commercial computer algebra?

ACL2
B method
PVS
HOL
Isabelle
Coq
Mizar
Twelf
Metamath

the reliability of computer algebra systems

```
> 2*infinity-infinity;
```

the reliability of computer algebra systems

```
> 2*infinity-infinity;
```

undefined

the reliability of computer algebra systems

```
> 2*infinity-infinity, 2*x-x;
```

undefined, x

the reliability of computer algebra systems

```
> 2*infinity-infinity, 2*x-x;
```

undefined, x

```
> subs(x=infinity, 2*x-x);
```

infinity

the reliability of computer algebra systems

```
> 2*infinity-infinity, 2*x-x;
```

undefined, x

```
> subs(x=infinity, 2*x-x);
```

infinity

```
> 1/(1-x) = simplify(1/(1-x))
```

$$\frac{1}{1-x} = -\frac{1}{-1+x}$$

the reliability of computer algebra systems

```
> 2*infinity-infinity, 2*x-x;
```

undefined, x

```
> subs(x=infinity, 2*x-x);
```

infinity

```
> int(1/(1-x),x) = int(simplify(1/(1-x)),x);
```

$$-\ln(1-x) = -\ln(-1+x)$$

the reliability of computer algebra systems

```
> 2*infinity-infinity, 2*x-x;
```

undefined, x

```
> subs(x=infinity, 2*x-x);
```

infinity

```
> int(1/(1-x),x) = int(simplify(1/(1-x)),x);
```

$$-\ln(1-x) = -\ln(-1+x)$$


```
> evalf(subs(x=-1, %));
```

$$-0.6931471806 = -0.6931471806 - 3.141592654i$$


killer app for proof assistants?

- ▶ reliable **software** and **hardware** development


killer app for proof assistants?

- ▶ reliable **software** and **hardware** development
- ▶ mathematics **education**
teaching students about mathematical proof
Christophe Rafalli,  Université de Savoie, 2002


killer app for proof assistants?

- ▶ reliable **software** and **hardware** development
- ▶ mathematics **education**
teaching students about mathematical proof
Christophe Rafalli,  Université de Savoie, 2002
- ▶ mathematically sound **computer algebra**

killer app for proof assistants?

- ▶ reliable **software** and **hardware** development
- ▶ mathematics **education**
teaching students about mathematical proof
Christophe Rafalli,  Université de Savoie, 2002
- ▶ mathematically sound **computer algebra**
performance? features?


killer app for proof assistants?

- ▶ reliable **software** and **hardware** development
- ▶ mathematics **education**
teaching students about mathematical proof
Christophe Rafalli,  Université de Savoie, 2002
- ▶ mathematically sound **computer algebra**
performance? features?

computers cannot do **high school mathematics**

$$x \neq 0 \wedge |\ln |x|| > 2 \wedge \int_0^{|x|} t dt \leq 1 \Rightarrow -\frac{1}{e^2} < x < \frac{1}{e^2}$$

killer app for proof assistants?

- ▶ reliable **software** and **hardware** development
- ▶ mathematics **education**
teaching students about mathematical proof
Christophe Rafalli,  Université de Savoie, 2002
- ▶ mathematically sound **computer algebra**
performance? features?

computers cannot do **high school mathematics** yet

$$x \neq 0 \wedge |\ln |x|| > 2 \wedge \int_0^{|x|} t dt \leq 1 \Rightarrow -\frac{1}{e^2} < x < \frac{1}{e^2}$$

should the next generation of proof assistants
use a declarative proof style?

should the next generation of proof assistants
use a declarative proof style?

ACL2
B method
PVS
HOL
Isabelle
Coq
Mizar
Twelf
Metamath

proof styles in proof assistants

- ▶ **tactic scripts** (HOL, Isabelle, Coq, PVS, B method, Metamath)
procedural
- ▶ **controlled natural language** (Mizar, Isabelle/Isar, Twelf)
declarative
- ▶ **generated natural language** (ACL2)
- ▶ **actual natural language**

proof styles in proof assistants

- ▶ **tactic scripts** (HOL, Isabelle, Coq, PVS, B method, Metamath)
procedural
- ▶ **controlled natural language** (Mizar, Isabelle/Isar, Twelf)
declarative
- ▶ **generated natural language** (ACL2)
- ▶ **actual natural language**
'computer should be able to parse mathematical \LaTeX '

proof styles in proof assistants

- ▶ **tactic scripts** (HOL, Isabelle, Coq, PVS, B method, Metamath)
procedural
- ▶ **controlled natural language** (Mizar, Isabelle/Isar, Twelf)
declarative
- ▶ **generated natural language** (ACL2)
- ▶ **actual natural language** (not practically possible)
'computer should be able to parse mathematical \LaTeX '

proof styles in proof assistants

- ▶ **tactic scripts** (HOL, Isabelle, Coq, PVS, B method, Metamath)
procedural
only understandable by interactively replaying the proof
- ▶ **controlled natural language** (Mizar, Isabelle/Isar, Twelf)
declarative
- ▶ **generated natural language** (ACL2)
- ▶ **actual natural language** (not practically possible)
'computer should be able to parse mathematical \LaTeX '

proof styles in proof assistants

- ▶ **tactic scripts** (HOL, Isabelle, Coq, PVS, B method, Metamath)
procedural
only understandable by interactively replaying the proof
- ▶ **controlled natural language** (Mizar, Isabelle/Isar, Twelf)
declarative
similar in look to programming language
- ▶ **generated natural language** (ACL2)
- ▶ **actual natural language** (not practically possible)
'computer should be able to parse mathematical \LaTeX '

proof styles in proof assistants

- ▶ **tactic scripts** (HOL, Isabelle, Coq, PVS, B method, Metamath)
procedural
only understandable by interactively replaying the proof
- ▶ **controlled natural language** (Mizar, Isabelle/Isar, Twelf)
declarative
similar in look to programming language
- ▶ **generated natural language** (ACL2)
- ▶ **actual natural language** (not practically possible)
'computer should be able to parse mathematical \LaTeX '

proof styles in proof assistants

- ▶ **tactic scripts** (HOL, Isabelle, Coq, PVS, B method, Metamath)
procedural
only understandable by interactively replaying the proof
- ▶ **controlled natural language** (Mizar, Isabelle/Isar, Twelf)
declarative
similar in look to programming language
- ▶ **generated natural language** (ACL2)
rather verbose: screens and screens of text
- ▶ **actual natural language** (not practically possible)
'computer should be able to parse mathematical \LaTeX '

proof styles in proof assistants

- ▶ **tactic scripts** (HOL, Isabelle, Coq, PVS, B method, Metamath)
procedural
only understandable by interactively replaying the proof
- ▶ **controlled natural language** (Mizar, Isabelle/Isar, Twelf)
declarative
similar in look to programming language
- ▶ **generated natural language** (ACL2, Theorema)
rather verbose: screens and screens of text
- ▶ **actual natural language** (not practically possible)
'computer should be able to parse mathematical \LaTeX '

Mohan Ganesalingam and the language of mathematics

Mohan Ganesalingam,  University of Cambridge
senior wrangler of his year

The Language of Mathematics, 2009
277 page master's thesis → PhD thesis

artificial language as close as possible to **actual** language

Mohan Ganesalingam and the language of mathematics

Mohan Ganesalingam,  University of Cambridge
senior wrangler of his year

The Language of Mathematics, 2009
277 page master's thesis → PhD thesis

artificial language as close as possible to **actual** language

 Andrzej Trybulec, **Mizar**

 Makarius Wenzel, Isabelle/**Isar**

 Andriy Paskevych, SAD/**ForTheL**

Mohan Ganesalingam and the language of mathematics

Mohan Ganesalingam,  University of Cambridge
senior wrangler of his year

The Language of Mathematics, 2009
277 page master's thesis → PhD thesis

artificial language as close as possible to **actual** language

 Andrzej Trybulec, **Mizar**

 Makarius Wenzel, Isabelle/**Isar**

 Andriy Paskevych, SAD/**ForTheL** (= 'Evidence Algorithm')

Mohan Ganesalingam and the language of mathematics

Mohan Ganesalingam,  University of Cambridge
senior wrangler of his year

The Language of Mathematics, 2009
277 page master's thesis → PhD thesis

artificial language as close as possible to **actual** language

 Andrzej Trybulec, **Mizar**

 Makarius Wenzel, Isabelle/**Isar**

 Andriy Paskevych, SAD/**ForTheL** (= 'Evidence Algorithm')

 Steven Kieffer, A language for mathematical knowledge management

 Peter Koepke, Naturalness in formal mathematics

 Claus Zinn, Understanding Informal Mathematical Discourse

 Aarne Ranta, Syntactic categories in the language of mathematics

Theorem 43 (Pythagoras' theorem). $\sqrt{2}$ is irrational.

The traditional proof ascribed to Pythagoras runs as follows. If $\sqrt{2}$ is rational, then the equation

$$a^2 = 2b^2 \tag{4.3.1}$$

is soluble in integers a, b with $(a, b) = 1$. Hence a^2 is even, and **therefore a is even**. If $a = 2c$, then $4c^2 = 2b^2$, $2c^2 = b^2$, and b is also even, contrary to the hypothesis that $(a, b) = 1$. \square

formal proof sketches

theorem :: *Pythagoras' theorem*

Th43: sqrt 2 is irrational

proof

:: *the traditional proof ascribed to Pythagoras:*

assume sqrt 2 is rational;

consider a, b **such that**

4.3.1: $a^2 = 2 * b^2$ **and**

a, b are_relative_prime;

then a^2 is even;

then a is even;

consider c **such that** $a = 2 * c$;

then $4 * c^2 = 2 * b^2$;

$2 * c^2 = b^2$;

b is even;

hence contradiction;

end;

formal proof sketches

theorem

Th43: sqrt 2 is irrational

:: (Pythagoras' theorem)

proof:: *the traditional proof ascribed to Pythagoras:***assume** sqrt 2 is rational;**consider** a, b **such that**4.3.1: $a^2 = 2 * b^2$ **and**

a, b are_relative_prime;

then a^2 is even;**then** a is even;**consider** c **such that** $a = 2 * c$;**then** $4 * c^2 = 2 * b^2$; $2 * c^2 = b^2$;

b is even;

hence contradiction;**end**;

theorem Th43: *sqrt 2 is irrational* :: (Pythagoras' theorem)

proof

:: *the traditional proof ascribed to Pythagoras:*

assume sqrt 2 is rational;

consider a, b **such that**

4.3.1: $a^2 = 2 * b^2$ **and**

a, b are_relative_prime;

then a^2 is even;

then a is even;

consider c **such that** $a = 2 * c$;

then $4 * c^2 = 2 * b^2$;

$2 * c^2 = b^2$;

b is even;

hence contradiction;

end;

theorem Th43: *sqrt 2 is irrational* :: (Pythagoras' theorem)

proof :: the traditional proof ascribed to Pythagoras:

assume sqrt 2 is rational;

consider a, b **such that**

4.3.1: $a^2 = 2 * b^2$ **and**

a, b are_relative_prime;

then a^2 is even;

then a is even;

consider c **such that** $a = 2 * c$;

then $4 * c^2 = 2 * b^2$;

$2 * c^2 = b^2$;

b is even;

hence contradiction;

end;

theorem Th43: *sqrt 2 is irrational* :: (Pythagoras' theorem)

proof :: the traditional proof ascribed to Pythagoras:
assume sqrt 2 is rational;

consider a, b **such that**

4.3.1: $a^2 = 2 * b^2$ **and**

a, b are_relative_prime;

then a^2 is even;

then a is even;

consider c **such that** $a = 2 * c$;

then $4 * c^2 = 2 * b^2$;

$2 * c^2 = b^2$;

b is even;

hence contradiction;

end;

theorem Th43: *sqrt 2 is irrational* :: (Pythagoras' theorem)

proof :: the traditional proof ascribed to Pythagoras:
assume sqrt 2 is rational; consider a, b such that

```

4.3.1:  $a^2 = 2 * b^2$  and
      a,b are_relative_prime;
then  $a^2$  is even;
then a is even;
consider c such that  $a = 2 * c$ ;
then  $4 * c^2 = 2 * b^2$ ;
       $2 * c^2 = b^2$ ;
      b is even;
hence contradiction;
end;

```

theorem Th43: *sqrt 2 is irrational* :: (Pythagoras' theorem)

proof :: the traditional proof ascribed to Pythagoras:
assume sqrt 2 is rational; consider a, b such that

```

4_3_1: a^2 = 2 * b^2 and
      a,b are_relative_prime;
then a^2 is even;
then a is even;
consider c such that a = 2 * c;
then 4 * c^2 = 2 * b^2;
      2 * c^2 = b^2;
      b is even;
hence contradiction;
end;
```

theorem Th43: *sqrt 2 is irrational* :: (Pythagoras' theorem)

proof :: the traditional proof ascribed to Pythagoras:
assume sqrt 2 is rational; consider a, b such that

4_3_1: $a^2 = 2 * b^2$

and

a, b are relative prime;

then a^2 is even;

then a is even;

consider c **such that** $a = 2 * c$;

then $4 * c^2 = 2 * b^2$;

$2 * c^2 = b^2$;

b is even;

hence contradiction;

end;

theorem Th43: *sqrt 2 is irrational* :: (Pythagoras' theorem)

proof :: the traditional proof ascribed to Pythagoras:
assume sqrt 2 is rational; consider a, b such that

$$4_3_1: \quad a^2 = 2 * b^2$$

and a, b are relative prime;
then a^2 is even;
then a is even;
consider c **such that** $a = 2 * c$;
then $4 * c^2 = 2 * b^2$;
 $2 * c^2 = b^2$;
 b is even;
hence contradiction;
end;

formal proof sketches

theorem Th43: *sqrt 2 is irrational* :: (Pythagoras' theorem)

proof :: the traditional proof ascribed to Pythagoras:
assume sqrt 2 is rational; consider a, b such that

$$4_3_1: \quad a^2 = 2 * b^2$$

and a, b are relative prime;

then a^2 is even;

then a is even;

consider c **such that** $a = 2 * c$;

then $4 * c^2 = 2 * b^2$;

$2 * c^2 = b^2$;

b is even;

hence contradiction;

end;

theorem Th43: *sqrt 2 is irrational* :: (Pythagoras' theorem)

proof :: the traditional proof ascribed to Pythagoras:
assume sqrt 2 is rational; consider a, b such that

$$4_3_1: \quad a^2 = 2 * b^2$$

and a, b are relative prime; then a^2 is even;

then a is even;

consider c **such that** $a = 2 * c$;

then $4 * c^2 = 2 * b^2$;

$2 * c^2 = b^2$;

b is even;

hence contradiction;

end;

theorem Th43: *sqrt 2 is irrational* :: (Pythagoras' theorem)

proof :: the traditional proof ascribed to Pythagoras:
assume sqrt 2 is rational; consider a, b such that

$$4_3_1: \quad a^2 = 2 * b^2$$

and a, b are relative prime; then a^2 is even; then a is even;

consider c **such that** $a = 2 * c$;

then $4 * c^2 = 2 * b^2$;

$2 * c^2 = b^2$;

b is even;

hence contradiction;

end;

theorem Th43: *sqrt 2 is irrational* :: (Pythagoras' theorem)

proof :: the traditional proof ascribed to Pythagoras:
assume sqrt 2 is rational; consider a, b such that

$$4_3_1: \quad a^2 = 2 * b^2$$

and a, b are relative prime; then a^2 is even; then a is even;
consider c such that $a = 2 * c$;

then $4 * c^2 = 2 * b^2$;

$2 * c^2 = b^2$;

b is even;

hence contradiction;

end;

theorem Th43: *sqrt 2 is irrational* :: (Pythagoras' theorem)

proof :: the traditional proof ascribed to Pythagoras:
assume sqrt 2 is rational; consider a, b such that

$$4_3_1: \quad a^2 = 2 * b^2$$

and a, b are relative prime; then a^2 is even; then a is even;
consider c such that $a = 2 * c$; then $4 * c^2 = 2 * b^2$;

$$2 * c^2 = b^2;$$

b is even;

hence contradiction;

end;

theorem Th43: *sqrt 2 is irrational* :: **(Pythagoras' theorem)**

proof :: the traditional proof ascribed to Pythagoras:
assume sqrt 2 is rational; consider a, b such that

$$4_3_1: \quad a^2 = 2 * b^2$$

and a, b are relative prime; then a^2 is even; **then a is even**;
consider c such that $a = 2 * c$; then $4 * c^2 = 2 * b^2$;
 $2 * c^2 = b^2$;

b is even;

hence contradiction;

end;

theorem Th43: *sqrt 2 is irrational* :: **(Pythagoras' theorem)**

proof :: the traditional proof ascribed to Pythagoras:
assume sqrt 2 is rational; consider a, b such that

$$4_3_1: \quad a^2 = 2 * b^2$$

and a, b are relative prime; then a^2 is even; **then a is even**;
consider c such that $a = 2 * c$; then $4 * c^2 = 2 * b^2$;
 $2 * c^2 = b^2$; b is even;

hence contradiction;
end;

theorem Th43: *sqrt 2 is irrational* :: **(Pythagoras' theorem)**

proof :: the traditional proof ascribed to Pythagoras:
assume sqrt 2 is rational; consider a, b such that

$$4_3_1: \quad a^2 = 2 * b^2$$

and a, b are relative prime; then a^2 is even; **then a is even**;
consider c such that $a = 2 * c$; then $4 * c^2 = 2 * b^2$;
 $2 * c^2 = b^2$; b is even; hence contradiction;

end;

theorem Th43: *sqrt 2 is irrational* :: **(Pythagoras' theorem)**

proof :: the traditional proof ascribed to Pythagoras:
 assume sqrt 2 is rational; consider a, b such that

$$4_3_1: \quad a^2 = 2 * b^2$$

and a, b are relative prime; then a^2 is even; **then a is even**;
 consider c such that $a = 2 * c$; then $4 * c^2 = 2 * b^2$;
 $2 * c^2 = b^2$; b is even; hence contradiction; **end**;

Theorem 43 (Pythagoras' theorem). $\sqrt{2}$ is irrational.

The traditional proof ascribed to Pythagoras runs as follows. If $\sqrt{2}$ is rational, then the equation

$$a^2 = 2b^2 \tag{4.3.1}$$

is soluble in integers a, b with $(a, b) = 1$. Hence a^2 is even, and **therefore a is even**. If $a = 2c$, then $4c^2 = 2b^2$, $2c^2 = b^2$, and b is also even, contrary to the hypothesis that $(a, b) = 1$. \square

Theorem 43 (Pythagoras' theorem). $\sqrt{2}$ is irrational.

The traditional proof ascribed to Pythagoras runs as follows. If $\sqrt{2}$ is rational, then the equation

$$a^2 = 2b^2 \tag{4.3.1}$$

is soluble in integers a, b with $(a, b) = 1$. Hence a^2 is even, and **therefore a is even**. If $a = 2c$, then $4c^2 = 2b^2$, $2c^2 = b^2$, and b is also even, contrary to the hypothesis that $(a, b) = 1$. \square

formal proof sketches


theorem Th43: *sqrt 2 is irrational* :: (Pythagoras' theorem)

proof :: the traditional proof ascribed to Pythagoras:
assume sqrt 2 is rational; consider a, b such that

$$4_3_1: \quad a^2 = 2 * b^2$$

and a, b are relative prime; then a^2 is even; then a is even;
consider c such that $a = 2 * c$; then $4 * c^2 = 2 * b^2$;
 $2 * c^2 = b^2$; b is even; hence contradiction; end;


integrating the procedural and declarative proof styles

Freek Wiedijk,  Radboud University Nijmegen, 2009

Mizar Light =

Mizar proof language on top of HOL Light system

integrating the procedural and declarative proof styles

Freek Wiedijk,  Radboud University Nijmegen, 2009


Mizar Light =

Mizar proof language on top of HOL Light system

three ways to create Mizar Light texts

- ▶ **editing** manually = declarative proof style
- ▶ **growing** by executing tactics = procedural proof style

integrating the procedural and declarative proof styles

Freek Wiedijk,  Radboud University Nijmegen, 2009


Mizar Light =

Mizar proof language on top of HOL Light system

three ways to create Mizar Light texts

- ▶ **editing** manually = declarative proof style
check-fix-extend cycle
- ▶ **growing** by executing tactics = procedural proof style

integrating the procedural and declarative proof styles

Freek Wiedijk,  Radboud University Nijmegen, 2009


Mizar Light =

Mizar proof language on top of HOL Light system

three ways to create Mizar Light texts

- ▶ **editing** manually = declarative proof style
check-fix-extend cycle
- ▶ **growing** by executing tactics = procedural proof style
lines without correct justification are the subgoals

integrating the procedural and declarative proof styles

Freek Wiedijk,  Radboud University Nijmegen, 2009

Mizar Light =

Mizar proof language on top of HOL Light system

three ways to create Mizar Light texts

- ▶ **editing** manually = declarative proof style
check-fix-extend cycle
- ▶ **growing** by executing tactics = procedural proof style
lines without correct justification are the subgoals
- ▶ **converting** existing HOL Light scripts

X

how should the next generation of proof assistants
be arrived at?

evolution versus revolution

- ▶ next generation **of an existing system?**

HOL Light → Mizar Light was an attempt at this

- ▶ new system **from scratch?**

evolution versus revolution

- ▶ next generation **of an existing system?**

HOL Light → Mizar Light was an attempt at this

best starting points = systems that answered 'yes' most

 Isabelle

 Coq

- ▶ new system **from scratch?**

evolution versus revolution

- ▶ next generation **of an existing system?**

HOL Light → Mizar Light was an attempt at this

best starting points = systems that answered 'yes' most

  Isabelle

 Coq

- ▶ new system **from scratch?**

throw away all that existing work?

evolution versus revolution

- ▶ next generation **of an existing system?**

HOL Light → Mizar Light was an attempt at this

best starting points = systems that answered 'yes' most

 Isabelle

 Coq

- ▶ new system **from scratch?**

throw away all that existing work?

many interoperable systems, or let the best system win?

evolution versus revolution

- ▶ next generation **of an existing system?**

HOL Light → Mizar Light was an attempt at this

best starting points = systems that answered 'yes' most

  Isabelle

 Coq

- ▶ new system **from scratch?**

throw away all that existing work?

many interoperable systems, or **let the best system win!**

the three bottlenecks

main improvements needed for wide adaptation of proof assistants

- ▶ better **automation**
- ▶ better mathematical **libraries**
- ▶ better match with existing mathematical **culture**

main improvements needed for wide adaptation of proof assistants

- ▶ better **automation**
- ▶ better mathematical **libraries**
- ▶ better match with existing mathematical **culture**
- ▶ better **visual** reasoning

main improvements needed for wide adaptation of proof assistants

- ▶ better **automation**
- ▶ better mathematical **libraries**
- ▶ better match with existing mathematical **culture**
- ▶ better **visual** reasoning

theorem $\arctan \frac{1}{2} + \arctan \frac{1}{3} = \frac{\pi}{4}$

proof

main improvements needed for wide adaptation of proof assistants

- ▶ better **automation**
- ▶ better mathematical **libraries**
- ▶ better match with existing mathematical **culture**
- ▶ better **visual** reasoning

theorem $\arctan \frac{1}{2} + \arctan \frac{1}{3} = \frac{\pi}{4}$

proof

> `simplify(arctan(1/2) + arctan(1/3) - Pi/4);`

main improvements needed for wide adaptation of proof assistants

- ▶ better **automation**
- ▶ better mathematical **libraries**
- ▶ better match with existing mathematical **culture**
- ▶ better **visual** reasoning

theorem $\arctan \frac{1}{2} + \arctan \frac{1}{3} = \frac{\pi}{4}$

proof

```
> simplify(arctan(1/2) + arctan(1/3) - Pi/4);
```

0



main improvements needed for wide adaptation of proof assistants

- ▶ better **automation**
- ▶ better mathematical **libraries**
- ▶ better match with existing mathematical **culture**
- ▶ better **visual** reasoning

theorem $\arctan \frac{1}{2} + \arctan \frac{1}{3} = \frac{\pi}{4}$

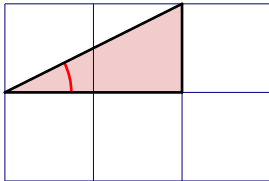
proof

main improvements needed for wide adaptation of proof assistants

- ▶ better **automation**
- ▶ better mathematical **libraries**
- ▶ better match with existing mathematical **culture**
- ▶ better **visual** reasoning

theorem $\arctan \frac{1}{2} + \arctan \frac{1}{3} = \frac{\pi}{4}$

proof

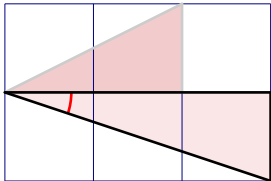


main improvements needed for wide adaptation of proof assistants

- ▶ better **automation**
- ▶ better mathematical **libraries**
- ▶ better match with existing mathematical **culture**
- ▶ better **visual** reasoning

theorem $\arctan \frac{1}{2} + \arctan \frac{1}{3} = \frac{\pi}{4}$

proof

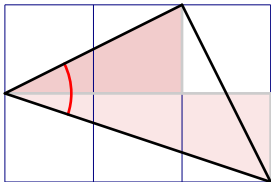


main improvements needed for wide adaptation of proof assistants

- ▶ better **automation**
- ▶ better mathematical **libraries**
- ▶ better match with existing mathematical **culture**
- ▶ better **visual** reasoning

theorem $\arctan \frac{1}{2} + \arctan \frac{1}{3} = \frac{\pi}{4}$

proof



main improvements needed for wide adaptation of proof assistants

- ▶ better **automation**
- ▶ better mathematical **libraries**
- ▶ better match with existing mathematical **culture**
- ▶ better **visual** reasoning

theorem $\arctan \frac{1}{2} + \arctan \frac{1}{3} = \frac{\pi}{4}$

proof

