

Inria



INSTITUT
DE RECHERCHE
EN INFORMATIQUE
FONDAMENTALE

Sabancı
Üniversitesi

Real World Coq Course

Sabancı University, Istanbul, August 12th 2022

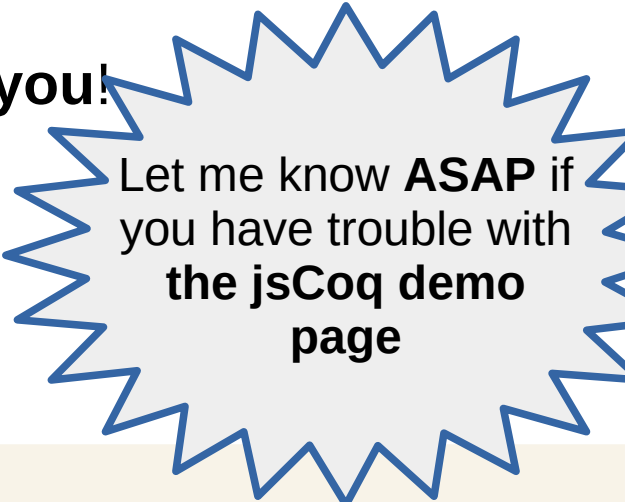
Coq Fundamentals

Emilio Jesús Gallego Arias – Inria Paris

Hoşgeldiniz!

Welcome all to the 2022 Real World Coq course at Sabancı.
Before we start, some **administrativia**:

- Course is split in **two main sessions**:
 - **Morning** (8:40 – 12:10): presentation of core material
 - **Afternoon** (13:10 – 15:30): assisted exercise time
- Both sessions have a **30 mins break**
- Advanced topics for the last day **to be chosen by you!**
- **Day 1**: Coq and Type theory
- **Day 2**: Proof tactics and libraries
- **Day 3**: Mathematics
- **Day 4**: Software verification



Let me know **ASAP** if
you have trouble with
**the jsCoq demo
page**

Online asynchronous help forum

<https://coq.zulipchat.com/#narrow/stream/341461-Sabanc.C4.B1-Coq-Course---Sept-2022>

Software that works: a hard task

Common Compiler Optimisations are Invalid
in the C11 Memory Model and what we can do about it

Ariane 5 rocket

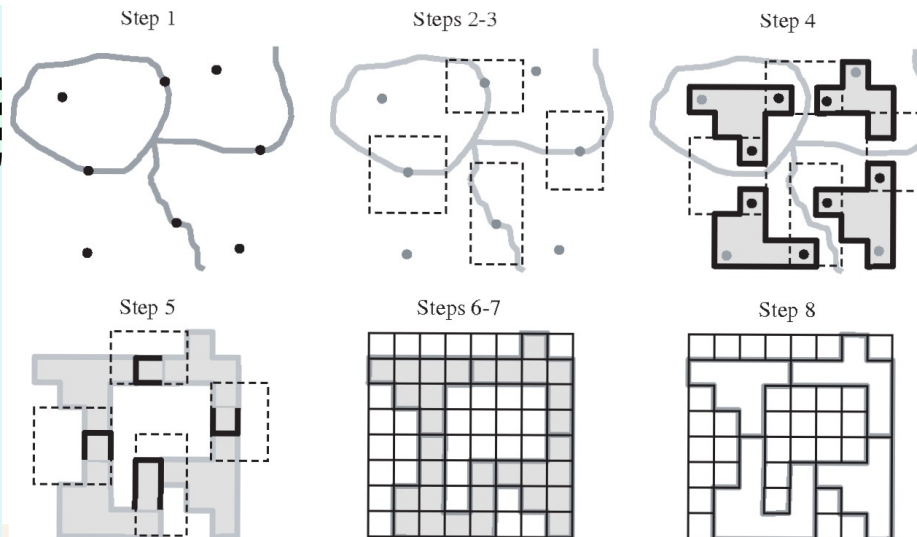
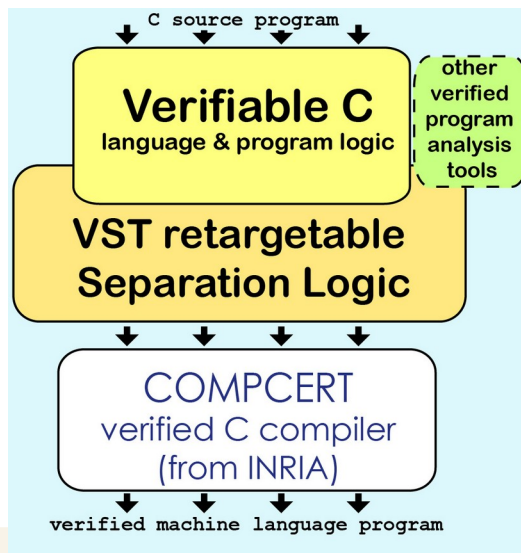


The rocket self-destructed 37 seconds after launch
Reason: A control software bug that went undetected
Conversion from 64-bit floating point to 16-bit signed integer value had an **exception**
The floating point number was larger than 32767 (max 16-bit signed integer)
Efficiency considerations had led to the disabling of the exception handler.
Program crashed rocket crashed
Total Cost: over \$1 billion

Formal Methods: Successes

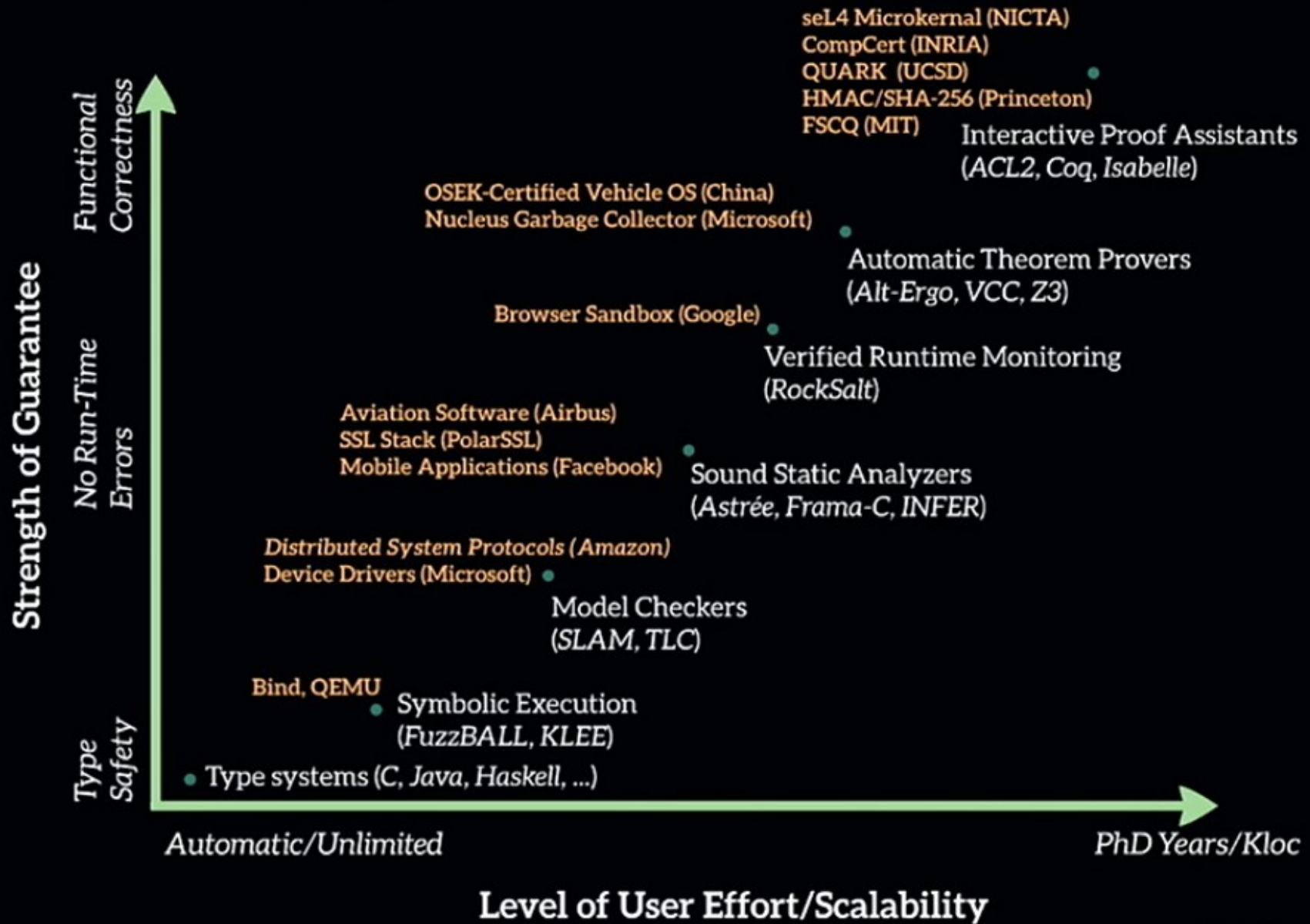
Formal Methods: Application of a broad set of techniques to problems in software and hardware verification and specification.

Many Success Stories: Verified compilers, Algorithms, OS kernels, Hardware, Protocols, Mathematical Proofs, ...



More than 50 years of history, but what's next?

Formal Method Based Tools



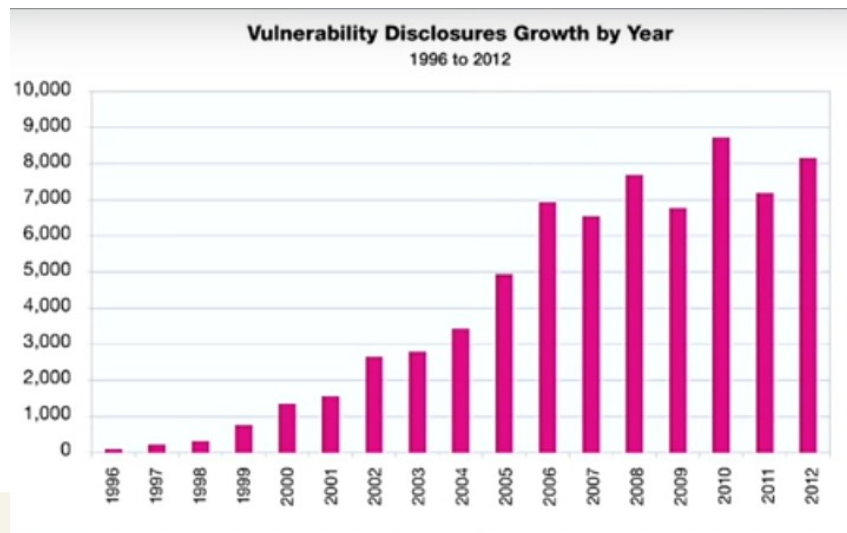
Notional graph

Formal Methods: Challenges

Key Challenges: Deep specification, new computing models (permission-less, quantum), effort, scalability, automation, education.

Common trend:

- *Horizontal* scaling: systems use more components
- *Vertical* scaling: components grow larger



2016-2019: Kernel +3.000.000 lines

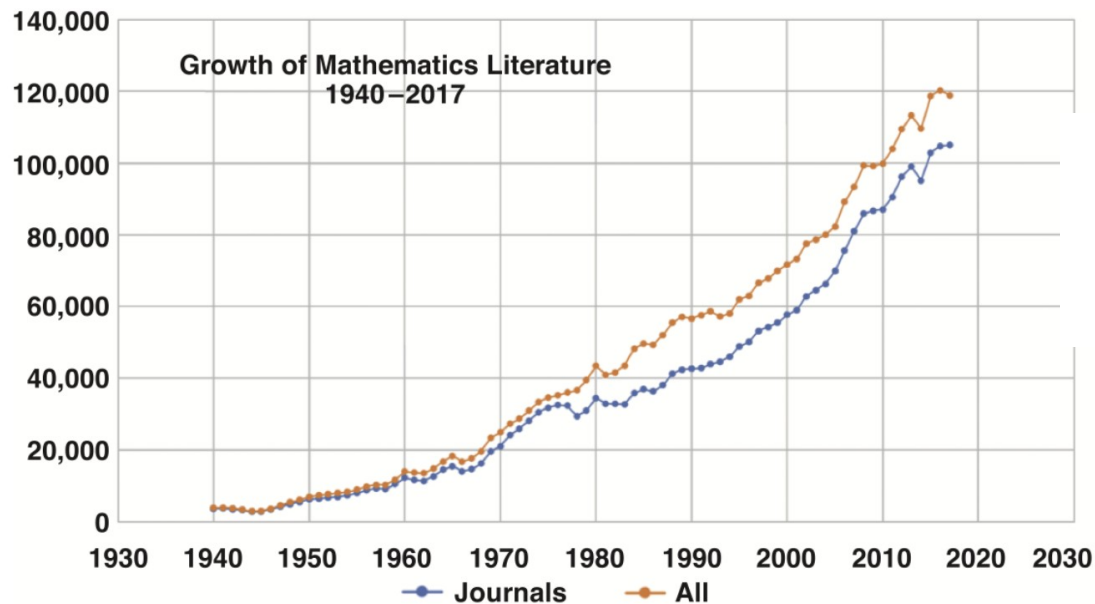
Year	Linux Kernel	GCC
1996	640.000	100.000
2006	5.000.000	2.000.000
2016	20.000.000	15.000.000

Figure – Core Software: Lines per Year

Complete Understanding of Systems Harder and Harder

The Growth of Scientific Knowledge

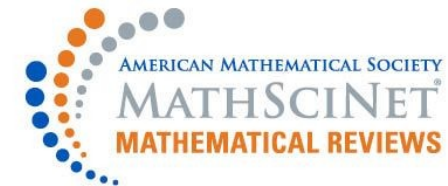
AMS: 3% increase in math production per year
400.000 papers per year by **2045**



E. Dunne, "Looking at the mathematics literature", *Notices of the American Mathematical Society*, vol. **66**, no. 2, pp. 227-230, 2019. ams.org/journals/notices/201902/rnoti-p227.pdf.

The proof that wasn't

Nick Scott explains the story of a mathematical proof that has sparked controversy, questioning how extremely complicated work can be validated if few understand it



4,067,699 publications
1,117,951 authors

About 11,800,000 results (0.41 seconds)

How to Resolve Merge Conflicts in Git?

*"If you think your job is getting **harder**, you are **correct**.
The mathematics literature is growing relentlessly, and
becoming **harder to figure out** along the way"*

E. Dunne

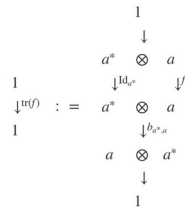
Collaborative Mathematical Writing

WWW: diversification in knowledge production

ncatlab.org/nlab/show/trace

2. Definition

The idea of the trace operation is easily seen in string diagram notation: essentially one takes the endomorphism $a \xrightarrow{a} a$, "bends it around" using the duality and the symmetry and connects its output to its input.



This definition makes sense in any braided monoidal category, but often in non-symmetric cases one wants instead a slightly modified version which requires the extra structure of a balancing.

Wikis / online editors very popular
New writing media brings new opportunities

More than **8.000.000** active **Jupyter** Notebooks on GitHub.com !

Coq and Type Theory: 2019-2022

2019: Environments for Large-Scale Proof Development

Focus on advanced proof engineers, multi-system

- Coq's **Continuous** Integration & **Industrial** Build Systems (creator & maintainer)
> **3 million lines of Specs and Proofs**
- Complex **interop** with *Mach. Learning / Soft. Eng.* : **document matters!**

The many facets of Networked Mathematics

by [Valeria de Paiva](#)



Monday, 18 Apr 2022

FOUNDATIONS OF MATHEMATICS

Building the Mathematical Library of the Future



Andrej Bauer @andrejbauer · 21h

Am I correctly counting four Fields medalists who are seriously taking computer-assisted mathematics? When the first one came along it created a small revolution. Can't wait to see what the combined power of three of them can do.

Online **Collaboration** + **Formal Mathematics** more important!

- From *advanced proof engineers* to **advanced mathematicians**
- Essential *feedback* from **Inria/IRIF, nLab** and **teaching** community

2022: From *mathematical* to **formal** documents

Focus on **collaboration, evolution** of documents

Collaborative Mathematical Writing

WWW: diversification in knowledge production

Template for preparing your...

66 For the authors' names, indicate different affiliations with the symbols: \lasti, \iDaggeri, \iDagger, \iS, after four authors, the symbol's double, \iDagger, \iS, quadleft, and so forth as required.

67 \section{your abstract}

68 In addition to the guidelines provided in the example abstract above, your abstract should:

69 \begin{listofitems}

70 \item provide a synopsis of the entire article;

71 \item begin with the broad context of the study, followed by specific background for the study;

72 \item describe the purpose, methods and procedures, core findings and results, and conclusions of the study;

73 \item emphasize new or important aspects of the research;

74 \item engage the broad readership of GENETICS and be understandable to a diverse audience (avoid using jargon);

75 \item be a single paragraph of less than 250 words;

76 \item contain the full name of the organism studied;

77 \item NOT contain citations or abbreviations.

78 \end{listofitems}

79 \section{Introduction}

80 In individual organisms where a mutant is being studied, the rationale for the study of that mutant must be clear to a geneticist not studying that particular organism. Similarly, study of particular phenotypes should be justified broadly and not on the basis of interest for that organism alone. General background on the importance of the genetic pathway and/or phenotype should be provided in a single, well-reasoned paragraph near the beginning of the introduction.

81 Authors are encouraged to:

Jupyter Lorenz Differential Equations (autosaved)

File Edit View Insert Cell Kernel Help Python 3.0

Exploring the Lorenz System

In this Notebook we explore the Lorenz system of differential equations:

$$\begin{aligned} \dot{x} &= \sigma(y - x) \\ \dot{y} &= \rho x - y - xz \\ \dot{z} &= -\beta z + xy \end{aligned}$$

This is one of the classic systems in non-linear differential equations. It exhibits a range of complex behaviors as the parameters (σ, β, ρ) are varied, including what are known as chaotic solutions. The system was originally developed as a simplified mathematical model for atmospheric convection in 1963.

In [7]: `Interact(Lorenz, N=Fixed(10), angle=(0.,360.), sigma=(0.0,50.0),beta=(0.,5),rho=(0.0,50.0))`

angle 308.2
max_time 12
 σ 10
 β 2.6
 ρ 28

Run some Python

To run the code below:

1. Click on the cell to see
2. Press SHIFT+ENTER

A full tutorial for using the

In []: `!matplotlib inline`
`import pandas as pd`
`import numpy as np`
`import matplotlib`

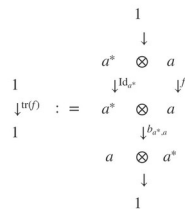
Wikis / online editors very popular
New writing media brings new opportunities

ncatlab.org/nlab/show/trace

Aplicaciones Lista de lectura

2. Definition

The idea of the trace operation is easily seen in [string diagram](#) notation: essentially one takes the endomorphism $a \xrightarrow{1} a$, "bends it around" using the duality and the symmetry and connects its output to its input.



This definition makes sense in any [braided monoidal category](#), but often in non-symmetric cases one wants instead a slightly modified version which requires the extra structure of a [balancing](#).

More than **8.000.000** active **Jupyter** Notebooks on GitHub.com !

Timeline of Type Theory

Goal: Build a foundation for mathematics!

- **19th century:** Cantor, Frege, Peano, Pierce, Brouwer
- **1903:** Russel's **Type Theory**
- **1930s:** Gentzen, Gödel; **consistency and incompleteness**
- **1940:** Church's simple theory of types; **lambda-calculus**
- **1950:** Gödel's System T; **Dialectica Interpretation**
- **1970:** Girard's **System F**, Martin-Lof **Type Theory**
- **1980:** Coquand's **Calculus of Constructions**

A stunning pace of development in a short period of time
(Many others Heyting, Kolmogorov, Kleene, Curry, Howard)

Coq: Foundational Theorem Proving

Foundational: Proofs written in *Minimal yet Expressive* Calculus

Verification: Proofs automatically checked by *trusted* small **kernels**

- A **milestone** of 20th century logic and computer science
- Enabled a **very high degree of confidence** on many **key** mathematical results and **critical** software
- **The Coq Proof Assistant:** field leader (Inria, 1984-now) (myself: 2015-now)
- **2013: ACM System Software Award**
- **2022: Open Science Award** (collab dev)

Proofs as programs: very well suited for both math and software validation

What's next?



Verifying the
Four Colour Theorem

Georges Gonthier

The Coq Proof Assistant

Developed at **Inria** by **T. Coquand**, **G. Huet**, **C. Paulin**

- First usable version in **1985**
- Powerful **logical framework**, programming language
- Goal: mechanically-verified **programs** and **mathematical proofs** in a constructive meta-theory
- See *“Early history of Coq”* in the Coq’s reference manual
- Constructing proofs mainly in **“interactive nature”**

Very successful project

The Calculus of Constructions

Theoretical basis of **Coq**

Higher-order dependently-typed calculus

- $\Gamma \vdash p : T$ “program **p** has type **T** under assumptions Γ ”
- **T** is a **type**, or **proposition** to prove, examples:
 - $\exists (x : \text{nat}) , \text{Turing_machine}_{32}(x) = 33$
 - $\forall (t_1 t_2 : \text{st}) , t_1 \approx t_2 \Rightarrow \exists t'_1 t'_2 , t_1 \mapsto t'_1 \wedge t_2 \mapsto t'_2 \wedge t'_1 \approx t'_2$
- **p** is a **program**, or **proof** for **T**
 - “ $p : A \Rightarrow B$ ” for any **input** of type **A**, output a **B**
 - “ $p : \exists x , P(x)$ ” pair with a witness **w** and proof “**P(w)**”
 - “ $p : A \wedge B$ ” pair with proofs “**p₁ : A**” and “**p₂ : B**”

The Calculus of Constructions

Given a proof **p** and a goal **T**, Coq will check “**p** : **T**”
Type-checking is described using “**inference**” rules

1.
$$\frac{}{\Gamma \vdash P : T}$$
2.
$$\frac{}{\Gamma, x : A \vdash x : A}$$
3.
$$\frac{\Gamma, x : A \vdash B : K \quad \Gamma, x : A \vdash N : B}{\Gamma \vdash (\lambda x : A. N) : (\forall x : A. B) : K}$$
4.
$$\frac{\Gamma \vdash M : (\forall x : A. B) \quad \Gamma \vdash N : A}{\Gamma \vdash MN : B[x := N]}$$
5.
$$\frac{\Gamma \vdash M : A \quad A =_{\beta} B \quad \Gamma \vdash B : K}{\Gamma \vdash M : B}$$

$e ::= \mathbf{T} \mid \mathbf{P} \mid x \mid ee \mid \lambda x:e. e \mid \forall x:e. e$

The Coq Proof Assistant

Core development team: **12 developers**

- A few dozens of **external contributors**
- Free software, open development model @ Github
- **1.000-10.000** regular users, including all target groups
- 10-100 active **research projects** in the world
- > 3.000.000 lines tested in the CI, in the wild 1 order more
- Fairly high degree of **maturity**, but active development

High-level Goals of Coq for 2020s

Improve how we

Produce
Organize
Interact with
Evolve
Collaborate on
Validate

scientific documents

Core Hypothesis:

**Programming Languages
&
Interactive Theorem Provers**

have reached a maturity point
where we can build upon them
formal, verifiable, hybrid documents
& theory for **collaboration** and **evolution**

We aim to develop a document model is
designed to **enable** many interesting
interactions with other research fields

Proving and verifying in the CoC

Strong points:

- Programs and proofs live at the same level
- Small, reliable kernel, good “Trusted Computing Base”
- **Foundational character**
- PL and theorem proving ideas do apply

Challenges:

- Writing programs proofs is very verbose, requires automation
- Language is low-level, notations and encoding needed
- Underlying logic very general, but not necessarily adapted to all domains

Beyond Programs: Data

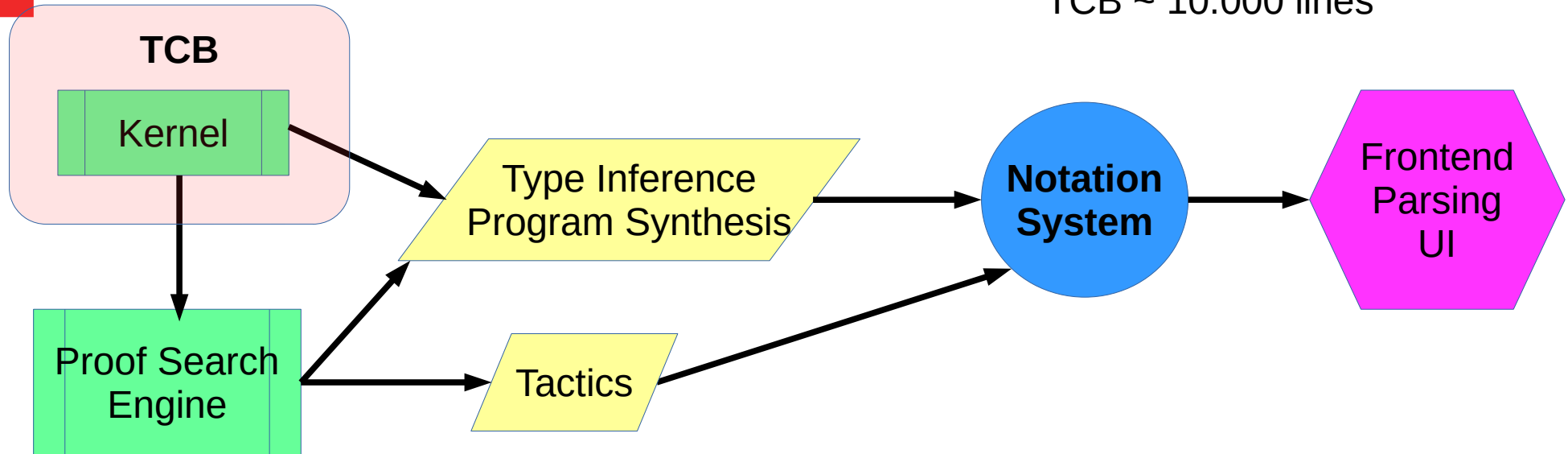
The Calculus of Inductive Constructions

- **Coq** also provides a powerful data-definition mechanism
- “**Inductive**” data types can encode arbitrary relations
- Well beyond safety, etc...
- However they have no **computational** content
- Commonly used to encode **transition systems**, rules, ...
- Also the base for most logical connectives

Key to being friendly for PL verification.

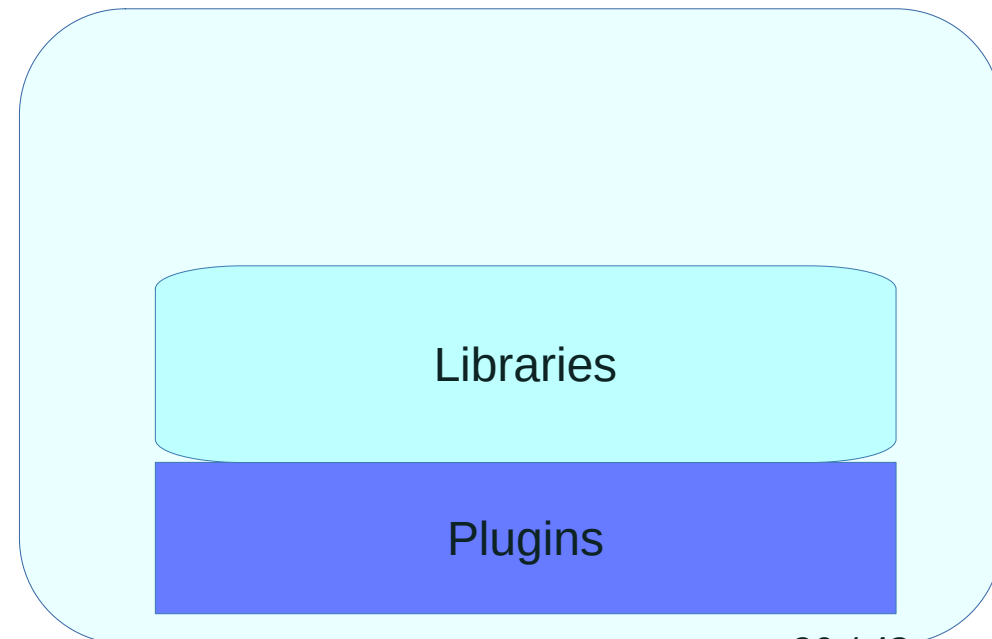
Organization of Coq

200.000 lines of OCaml code
120.000 lines of Coq code
5.000 lines of C
TCB ~ 10.000 lines



The Interactive Proof Cycle

- Users input documents in a high-level mathematical proof language
- High-level language is elaborated to the core calculus
- Tactics and type inference perform program search.
- Kernel checks correctness of the proofs
- System is extensible



Success Stories

- **Fundamental maths:** 4-color theorem, Feit-Thompson

```
Qed.  
  
Theorem Feit_Thompson (gT : finGroupType) (G : {group gT}) :  
  odd #|G| -> solvable G.  
Proof. exact: (minSimpleOdd_ind no_minSimple_odd_group). Qed.  
  
Theorem simple_odd_group_prime (gT : finGroupType) (G : {group gT})  
  odd #|G| -> simple G -> prime #|G|.   
Proof. exact: (minSimpleOdd_prime no_minSimple_odd_group). Qed.
```

Success Stories

- **Software Verification:** CompCert, Fiat-Crypto, DeepSpec, IRIS, blockchain, great impact at PL venues

```
( $\exists$  b : bool, l ↦ #b * if b then True else R)%I.
```

```
(** Invariants in Iris are named by a *namespace* so that s  
can be opened at the same time, while guaranteeing that no  
twice at the same time (which would be unsound). Here, this  
since acquiring and releasing a lock only requires to open
```

The namespace [lockN] of the lock invariant:

```
*)  
Definition lockN : namespace := nroot .@ "lock".  
Definition is_lock (lk : val) (R : iProp  $\Sigma$ ) : iProp  $\Sigma$  :=  
  ( $\exists$  l : loc,  $\lceil$  lk = #l  $\lceil$   $\wedge$  inv lockN (lock_inv l R))%I.
```

```
(** The main proofs. *)  
Lemma newlock_spec (R : iProp  $\Sigma$ ):  
  {{{ R }}} newlock #() {{{ lk, RET lk; is_lock lk R }}}.  
Proof.  
  iIntros ( $\Phi$ ) "HR H $\Phi$ ". iApply wp_fupd.  
  wp_lam. wp_alloc l as "Hl".  
  (** Use the Iris rule [inv_alloc] for allocating a lock.  $\rightarrow$   
  resources [HR : R] and the points-to [l ↦ #false] into th  
  iMod (inv_alloc lockN _ (lock_inv l R) with "[HR Hl]") as  
  { iNext. iExists false. iFrame. }  
  iModIntro. iApply "H $\Phi$ ". iExists l. eauto.  
Qed.
```

```
1 subgoal (ID 534)
```

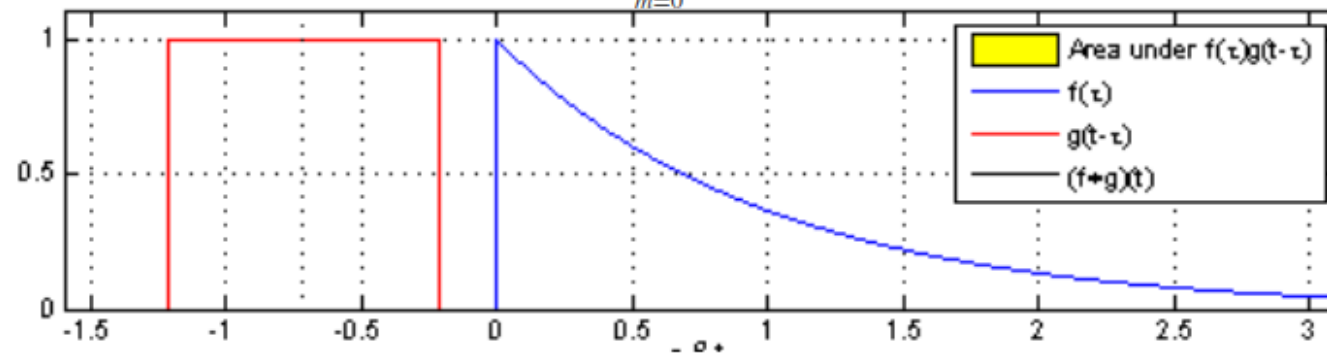
```
 $\Sigma$  : gFunctors  
heapG0 : heapG  $\Sigma$   
R : iProp  $\Sigma$   
 $\Phi$  : val  $\rightarrow$  uPred (iResUR  $\Sigma$ )  
l : loc  
=====  
"Hinv" : inv lockN (lock_inv l R)  
-----  
"H $\Phi$ " :  $\forall$  lk : val, is_lock lk R -*  $\Phi$  lk  
-----  
|={T} $\Rightarrow$   $\Phi$  #l
```

Success Stories

- **Teaching:** Software Foundations, CPDT, many schools and tutorials

Circular Convolution of two Signals

$$(x \otimes y)_n = \sum_{m=0}^{N-1} x(m)y(n - m)$$



```
1 Definition convs x y := \col_n \sum_m x m \theta * y (n-m) \theta.
```

$$\begin{aligned} (x \otimes y)_n &= \sum_{m=0}^{N-1} x(m)y(n - m) = \sum_{l=n}^{n-(N-1)} x(n - l)y(l) \\ &= \sum_{l=0}^{N-1} y(l)x(n - l) \\ &= (y \otimes x)_n \end{aligned}$$

```
1 Lemma convsC : commutative convs.
2 Proof.
3 move=> x y; apply/matrixP=> n k; rewrite !mxE {k}.
4 rewrite (reindex inj (inj comp (addrI n) oppr inj)).
```

Links to resources
In the course
Webpage

Coq vs Other Systems

Coq both a **PL** and an **Interactive Theorem Prover**

- vs **traditional PL**:
Notations, elaboration, implicit arguments, tactics, higher-order unif, partial evaluation, fp, interactive development, slower
- vs **Isabelle**: Different logic ; Isabelle interface much more user-friendly
- vs **Lean**:
different development model and user base, different strengths implementation & compatibility. Punch with math community.
- Vs **Adga**: impredicativity, trust-base, tactics, less experimental features

Development needs to adapt, or risk becoming obsolete.
Huge legacy codebase difficults progress => research topic

Coq's Ecosystem

Large work in the last years to build a community

- **Zulip Forum:** Main forum, both users and devs
- **Coq Community:** collective maintenance
- StackOverflow, mailing lists
- GitHub project
- **Events:** 2 Workshops, 1 user and dev meetings, diversity, misc hackathons, schools....

Development model pretty unique among interactive theorem Provers, has pros and cons.

Coq's Vernacular Language

Type theory is a **very bare** language

Coq provides many **user-level** constructions to do math

```
Record abelian (V : Type) :=
  Mixin {
    zero : V;
    opp  : V -> V;
    add  : V -> V -> V;
    _    : associative add;
    _    : commutative add;
    _    : left_id zero add;
    _    : left_inverse zero opp add
  }.
```

```
Instance abelian int :=
  {
    zero := 0;
    opp  : -;
    add  : +;
    addiA;
    addiC;
    add0i;
    ...
  }.
```

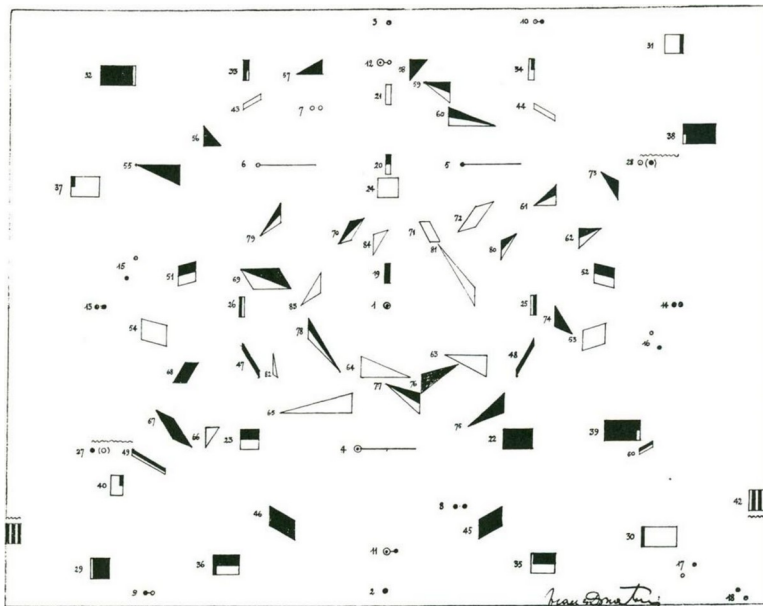
```
Lemma mulrD1 (T:abType) (x y:T) n: {morph (x => x *+ n) : x y / x + y}.
Proof. move=> x y; elim: n => [|n IHn]; rewrite ?addr0 // !mulrS.
by rewrite addrCA -!addrA -IHn -addrCA.
Qed.
```

Of particular interest are **notations, tactics, structures, hints, definitions and modules, ...** (over 200 vernaculars)

Doing Proofs: What is Hard?

With **high confidence** comes **high cost**

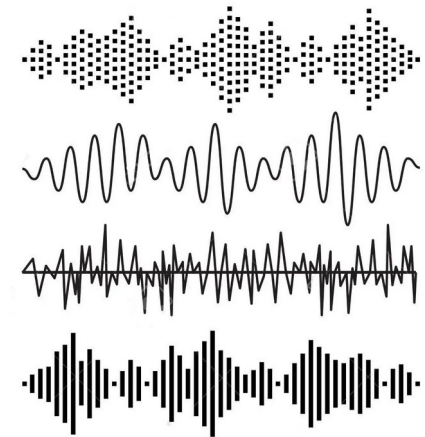
Proof assistants **notoriously difficult to use**



Franco Donatoni – Babàì

Notations (c) John Cage

Interpretation
→



Human vs Machine *impedance*

Doing Proofs: What is Hard?

With **high confidence** comes **high cost**
 Proof assistants **notoriously difficult to use**

Lemma 8.2.3. Let $F : S_1 \rightarrow S_2$ be a 1-morphism of fibred categories over the cate. $U \in \text{Ob}(C)$ and $x, y \in \text{Ob}((S_1)_U)$. Then F defines a canonical morphism of presheave

$$\text{Mor}_{S_1}(x, y) \longrightarrow \text{Mor}_{S_2}(F(x), F(y))$$

on CU .

Proof. By Categories, Definition 4.33.9 the functor F maps strongly cartesian morphisms cartesian morphisms. Hence if $f : V \rightarrow U$ is a morphism in C , then there are canonical morphisms $\alpha_V : f^*F(x) \rightarrow F(f^*x)$, $\beta_V : f^*F(y) \rightarrow F(f^*y)$ such that $f^*F(x) \rightarrow F(f^*x)$ the canonical morphism $f^*F(x) \rightarrow F(x)$, and similarly for β_V . Thus we may define

$$\text{Mor}_{S_1}(x, y)(f : V \rightarrow U) \longlongrightarrow \text{Mor}_{S_2}(f^*x, f^*y)$$

$$\text{Mor}_{S_2}(F(x), F(y))(f : V \rightarrow U) \longlongrightarrow \text{Mor}_{S_2}(f^*F(x), f^*F(y))$$

by $\phi \mapsto \beta_V^{-1} \circ F(\phi) \circ \alpha_V$. We omit the verification that this is compatible with the restriction mappings. □

(c) The Stacks Project

```
(* over the 600-line bar).
Theorem Ptype_embedding M K :
  M \in 'M_'P -> \kappa(M).-Hall(M) K ->
  exists2 Mstar, Mstar \in 'M_'P /\ gval Mstar \notin M :^: G
  & let Kstar := 'C (M' \sigma)(K) in
  let Z := K <*> Kstar in let Zhat := Z :|: (K :|: Kstar) in
  [/\ (*a*) {in 'E^1(K), forall X, 'M('C(X)) = [set Mstar]},
  (*b*) \kappa(Mstar).-Hall(Mstar) Kstar /\ \sigma(M).-Hall(Mstar) Kstar
  (*c*) 'C (Mstar' \sigma)(Kstar) = K /\ \kappa(M) =i \tau(M),
  (*d*) [/\ cyclic Z, M :&: Mstar = Z,
  {in K^#, forall x, 'C_M[x] = Z},
  {in Kstar^#, forall y, 'C_Mstar[y] = Z}
  & {in K^# & Kstar^#, forall x y, 'C[x * y] = Z}]
  & [/\ (*e*) [/\ normedTI Zhat G Z, {in ~: M, forall g, [disjoint Zhat & M :^:
  & (#|G|:R / 2%:R < #|class_support Zhat G|:R >: rat)%R ],
  (*f*) M \in 'M_'P2 /\ prime #|K| \ \ Mstar \in 'M_'P2 /\ prime #|Kstar|
  (*g*) {in 'M_'P, forall H, gval H \in M :^: G :|: Mstar :^: G}
  & (*h*) M^(1) >>| K = M]].
```

Coq Document (Text-based)



```
n : nat =>
  _evar_0 : @eq nat (addn m (addn n 0)) (addn n m) =>
  _ind nat (addn n 0)
  fun _pattern_value_ : nat =>
  @eq nat (addn m _pattern_value_) (addn n m) _evar_0 _n
  (addn n)
  fun _evar_0 : @eq nat (addn n (addn m 0)) (addn n m) =>
  @eq _ind_r nat (addn n (addn m 0))
  (fun _pattern_value_ : nat =>
  @eq nat _pattern_value_ (addn n m) _evar_0
  (addn m (addn n 0)) (addnCA m n 0))
  ((fun _evar_0 : @eq nat (addn n m) (addn n m) =>
  @eq _ind_r nat m
  (fun _pattern_value_ : nat =>
  @eq nat (addn n _pattern_value_) (addn n m) _evar_0
  (addn m 0) (addn0 m)) (@Logic.eq_refl nat (addn n m))))
  :
  @commutative nat nat addn
  : @commutative nat nat addn
```

Kernel-level Proof Term

Manual Translation + Interactive Interpretation

Human side: rich *natural, mathematical, graphical* language
Computer side: minimal “*assembler*” language of *proof terms*

Other Important Challenges

Installing things!

Libraries that don't work / outdated proofs

- **Searching** for things without success
- Bad **display** / **notations**
- Boilerplate / **trivial** proofs
- Synchronization / **merging** problems
- Lack of **documentation**
- **Dumb** or outdated **interfaces**



A mix of Social, Research, and Engineering Problems!

Have we reached a Critical Point?

Recent times have seen a **proliferation** of **formal** and **semi-formal collaborative math writing systems**

- LaTeX / Literate Programming: **Stacks**
- Education for Maths: **Edukera, WaterProof**
- Semantic-Aware, Interactive: **Nota, ScholarPhi**
- Structure-Aware: **Hazelnut, Actema**
- Interactive Documentation: **Alectryon**
- Self-contained formal documents: **jsCoq, Holbert**

How far from an integral solution?

We have reached a Critical Point

Current solutions don't address current needs

- **Jupyter Notebooks:** Great for computational content, falls short for general verified math and software
- **Overleaf, Wikis, Stacks:** Don't integrate with tools that can understand and validate content
- **Traditional ITPs (Coq, Lean, Isabelle,...):** Lack accessibility, collaboration features

The area has become a **very hot topic** in the last year

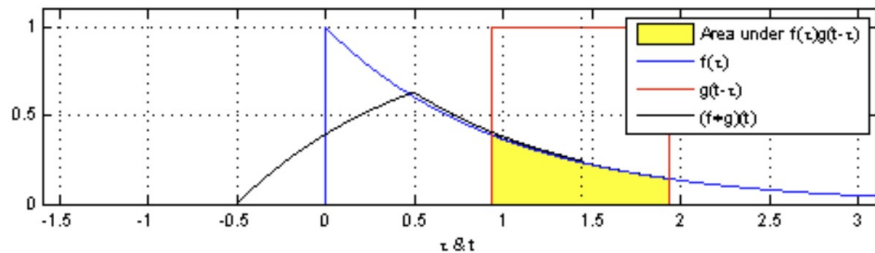
jscoq.wiki: a formally-verifiable Wiki!

jsCoq: Towards Hybrid Interfaces for Theorem Proving (UITP2016)

→ x80.org/rhino-coq/v8.11/examples/dft.html

Circular Convolution of two Signals

$$(x \otimes y)_n = \sum_{m=0}^{N-1} x(m)y(n-m)$$



Credits: Wikipedia/Brian Amberg

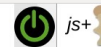
```
57 Definition convs x y := \col_n \sum_m x m 0 * y (n-m) 0.  
58
```

Now we can do our first non-trivial proof using Coq! Let's see how the paper proof compares:

$$\begin{aligned} (x \otimes y)_n &= \sum_{m=0}^{N-1} x(m)y(n-m) = \sum_{l=n}^{n-(N-1)} x(n-l)y(l) \\ &= \sum_{l=0}^{N-1} y(l)x(n-l) \\ &= (y \otimes x)_n \end{aligned}$$

"In the first step we made the change of summation variable $l \equiv n - m$, and in the second step, we made use of the fact that any sum over all N terms is equivalent to a sum from 0 to $N - 1$ ".

```
59 Lemma convsC : commutative convs.  
60 Proof.  
61 move=> x y; apply/matrixP=> n k; rewrite !mxE {k}.  
62 rewrite (reindex_inj (inj_comp (addrI n) oppr_inj)).  
63 by apply/eq_bigr=> m _; rewrite opprD addNKr opprK mulrC.  
64 Qed.
```



js+

Readme @

Goals

1 goal

```
N' : nat  
R : comRingType  
x, y : 'cV_N  
n : 'I_N
```

```
\sum_m x m 0 * y (n - m) 0 = \sum_m y m 0 * x (n - m) 0
```

Messages

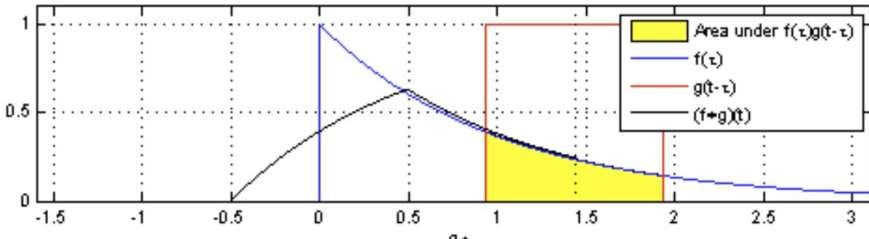
```
mathcomp.solvable.commutator loaded.  
mathcomp.solvable.center loaded.  
mathcomp.solvable.gseries loaded.  
mathcomp.solvable.nilpotent loaded.  
mathcomp.solvable.sylow loaded.  
mathcomp.field.falgebra loaded.  
mathcomp.field.fieldext loaded.  
mathcomp.algebra.polyXY loaded.  
mathcomp.field.separable loaded.  
mathcomp.field.galois loaded.  
mathcomp.field.algebras_fundamentals loaded.  
Dsp.dsp support loaded.  
mathcomp.field.cyclotomic loaded.
```

jscoq.wiki: a formally-verifiable Wiki!

jsCoq: Towards Hybrid Interfaces for Theorem Proving (UITP2016)

→ x80.org/rhino-coq/v8.11/examples/dft.html

Circular Convolution of two Signals

$$(x \otimes y)_n = \sum_{m=0}^{N-1} x(m)y(n-m)$$


Credits: Wikipedia/Brian Amberg

```
57 Definition convs x y := \col_n \sum_m x m 0 * y (n-m) 0.
58
```

Now we can do our first non-trivial proof using Coq! Let's see how the paper

$$\begin{aligned} (x \otimes y)_n &= \sum_{m=0}^{N-1} x(m)y(n-m) = \sum_{l=n}^{n-(N-1)} \\ &= \sum_{l=0}^{N-1} y(l)x(n-l) \\ &= (y \otimes x)_n \end{aligned}$$

"In the first step we made the change of summation variable $l \equiv n - m$, and in the second step, we made use of the fact that any sum over all N terms is equivalent to a sum from 0 to $N - 1$ ".

```
59 Lemma convsC : commutative convs.
60 Proof.
61 move=> x y; apply/matrixP=> n k; rewrite !mxE {k}.
62 rewrite (reindex_inj (inj_comp (addrI n) oppr_inj)).
63 by apply/eq_bigr=> m _; rewrite opprD addNkr opprK mulrC.
64 Qed.
```

Goals

```
1 goal
N' : nat
R : comRingType
x, y : 'cV_N
n : 'I_N

\sum_m x m 0 * y (n - m) 0 = \sum_m y m 0 * x (n - m) 0
```

mathcomp.solvable.commutator loaded.
mathcomp.solvable.center loaded.
mathcomp.solvable.gseries loaded.
mathcomp.solvable.nilpotent loaded.
mathcomp.solvable.sylow loaded.
mathcomp.field.falgebra loaded.
mathcomp.field.fieldext loaded.
mathcomp.algebra.polyXY loaded.
mathcomp.field.separable loaded.
mathcomp.field.galois loaded.
mathcomp.field.algebras_fundamentals loaded.
Dsp.dsp support loaded.
mathcomp.field.cyclotomic loaded.

Try it!
jscoq.github.io

Formal Hybrid Documents

Definition and soundness of *interpretation*

We now **show** that $+$ is **commutative**:

```
Lemma addnC : commutative +.  
Proof. elim=> [//|n iHn]. Qed.
```

Interpretation

Coq's
Kernel
State

The document calculus knows *3 kinds of objects*, and organizes them by **containment**:

- **semi-structured text**: free form, metadata can be updated and extracted
- **meta-logical objects**: objects that are formal, but are not seen by the kernel
- **logical objects**: objects that will be sent to the kernel, after interpretation

Theorem (*soundness*): The interpretation function **respects** the **logical structure** in the document.

We **cannot skip** sending a logical definition or theorem to the kernel. Formally:

$$I(L1 \oplus L2, M) = I(L1, M) \oplus I(L2, M)$$

Note the document is **not checking correctly**, as the proof is incorrect

Formal Hybrid Documents

Gradual Document Interpretation: A **formal** theory of *Error Recovery*

We now **show** that `+` is **commutative**:

```
Lemma addnC : commutative +.  
Proof. elim=> [//|n iHn]. Qed.
```

```
Definition bar := addnC. ✓
```

Interpretation

Coq's
Kernel
State

Proof development is best done by **gradually** refining **human-style specs** to their formal counterpart.

In this example, the proof of `addnC` is replaced by an unknown `?`, which may only error if used. Error propagation can be **contained structurally**, to produce a better user experience. And `bar` can **still be checked**.

Gradual typing for Dependently Typed Systems is a very new area, and we will use it to formally model the **continuous process where a formal document evolves towards full validation**.

Theoretical Challenge: relation with interpretation soundness

Formal Hybrid Documents

Incremental Interpretation: Avoid Re-Doing Work

We now **show** that + is **commutative**:

```
Lemma addnC : commutative +.  
Proof. elim=> [//|n iHn]. Qed.  
Definition bar := addnC. ✓
```

$\Delta=[iHn,->]$

We now **show** that + is **commutative**:

```
Lemma addnC : commutative +.  
Proof. elim=> [//|n ->]. Qed. ✓  
Definition bar := addnC.
```

Interpretation

Coq's
Kernel
State

Incremental checking an **essential**
property for all our applications.

Soundness comes from the
commuting diagram.

Open: Incrementality + Structure

$I(\Delta=[iHn,->])$

Coq's
Kernel
State

Shift from **study of proofs** to the **study of evolution of proofs**

Enabling Document-based Research

Document theory: **validity, distance**

"Mutation Testing for Coq"

Use checker as **Oracle** in "soft" experiments

(ASE2019)

- Documents as **source**: Indexing, Dataset Extraction
- Documents as **target**: Automatic "fuzzy" translation of mathematical texts, with feedback!
- More: **Structured access** provides an abstraction layer
- Foundation for **M.L. / S.E.** collaboration

A more fancy example: **constraint-aided conflict resolution**,
SMT finds the **best resolution** w.r.t. doc soundness

SerAPI: Communicating with Coq

Enable other tools to interact easily with Coq

Not easy due to extensible nature; design constraints:

- **Low-effort:** cannot justify a large time sink
- **Lightweight:** neither can the users
- **Maintenable:** no use if it will stop working in 6 months
- **Robust:** API for clients should “resist change”
- **Machine-oriented:** Main use case is to talk to tools
- **User-driven:** convenience for users triumphs ideology
- Should be **easy to install**, work on unmodified Coq

Extensive use of OCaml’s meta-programming system

PPX

SerAPI: Interaction Protocol

control and query protocols

```
type cmd =  
  NewDoc   of newdoc_opts  
  Add      of add_opts * string  
  Cancel   of Stateid.t list  
  Exec     of Stateid.t  
  Query    of query_opt * query_cmd  
  Print    of print_opt * coq_object
```

```
type coq_object =  
  CoqPp    of Pp.t  
  CoqLoc   of Loc.t  
  CoqTok   of Tok.t list  
  CoqAst   of Vernacexpr.vernac_control Loc.located
```

```
type query_cmd =  
  Option (** List of options Coq knows about *)  
  Goals  (** Current goals, in kernel form *)  
  Ast    (** Ast for the current sentence *)  
  TypeOf of string
```

Then, these object definitions are serialized to JSON or Sexps

```
(Add ((ontop 3) (limit 3)) "Definition foo := 3.")    (Query ((sid 3)) Ast)
```

Improving Coq's Printing

Coq's current printing system still **textual**
 Roots on **console-based** interaction

Theorem 14.7. Suppose $M \in \mathcal{M}_{\mathcal{G}}$ and K is a Hall $\kappa(M)$. Let $K^* = C_{M^*}(K)$, $k = |K|$, $k^* = |K^*|$, $Z = K \times K^*$, and Then, for some other $M^* \in \mathcal{M}_{\mathcal{G}}$ not conjugate to M ,

- (a) $\mathcal{M}(C_G(X)) = \{M^*\}$ for every $X \in \mathcal{E}^1(K)$,
- (b) K^* is a Hall $\kappa(M^*)$ -subgroup of M^* and a Hall M^* , $\Rightarrow \sigma(M) \cap \tau(M^*) = \kappa(M^*)$
- (c) $K = C_{M^*}(K^*)$ and $\kappa(M) = \tau_1(M)$, $\rightarrow \sigma \cup \tau$
- (d) Z is cyclic and for every $x \in K^{\#}$ and $y \in K^{*\#}$ $C_M(x) = C_{M^*}(y) = C_G(xy)$,
- (e) \hat{Z} is a TI -subset of G with $N_G(\hat{Z}) = Z$, $\hat{Z} \cap g \in G - M$, and

$$|\mathcal{C}_G(\hat{Z})| = \left(1 - \frac{1}{k} - \frac{1}{k^*} + \frac{1}{kk^*}\right) |G|$$

- (f) M or M^* lies in $\mathcal{M}_{\mathcal{G}_2}$ and, accordingly, K or K^*
- (g) every $H \in \mathcal{M}_{\mathcal{G}}$ is conjugate to M or M^* in G ,

```
Theorem Ptype_embedding : forall M K,
  M \in 'M_'P -> \kappa(M).-Hall(M) K ->
  exists2 Mstar, Mstar \in 'M_'P /\ gval Mstar \notin M :^: G
  & let Kstar := 'C_(M`_\sigma)(K) in
  let Z := K <*> Kstar in let Zhat := Z :\ (K :|: Kstar) in
  [/\ (*a*) (in 'E^1(K), forall X, 'M('C(X)) = [set Mstar]],
  (*b*) \kappa(Mstar).-Hall(Mstar) Kstar /\ \sigma(M).-Hall(Mstar) Kstar,
  (*c*) 'C_(Mstar`_\sigma)(Kstar) = K /\ \kappa(M) = i \tau_1(M),
  (*d*) [/\ cyclic Z, M :&: Mstar = Z,
  {in K^{\#}, forall x, 'C_M[x] = Z}, {in Kstar^{\#}, forall y, 'C_Mstar[y] = Z}
  & {in K^{\#} & Kstar^{\#}, forall x y, 'C[x * y] = Z}]
  & [/\ (*e*) [/\ trivIset (Zhat :^: G), 'N(Zhat) = Z,
  {in ~: M, forall g, [disjoint Zhat & M :^: g]}
  & (#|G|:%R / 2:%R < #|class_support Zhat G|:%R => qnum)%R ],
  (*f*) M \in 'M_'P2 /\ prime #|K| \vee Mstar \in 'M_'P2 /\ prime #|Kstar|,
  (*g*) {in 'M_'P, forall H, gval H \in M :^: G :|: Mstar :^: G}
  & (*h*) M^(1) <*> K = M]].
```

Main problems: **1-dimensional layout**, lack of **meta-data**

The BoxModel.t printer

Adopt as **output** a **LaTeX/HTML box model**

Plus **attach semantic information** *à la Isabelle*

```
type t =
  Variable of string
  Constant of string
  Identifier of Id.t
  Sort of string list
  App of { fn : t
           ; impl : t list
           ; argl : t list
         }
  Abs of { kind : abs_kind; binderl : t list; v : t }
  Let of { lhs : t; rhs : t; typ : t option; v : t }
  Notation of
    { key : string
      ; args : t list
      ; raw : t
    }

module Id : sig
  type t =
    { relative : string
      ; absolute : string option
    }
end
```

Rendering to Web Components

Standard by Google, 2015, well **supported**

Allows to define **custom tags** in the DOM

- `<coq-notation raw="..."></coq-notation>`
- `<coq-app>...</coq-app>`
- `<coq-binder-list> ... </coq-binder-list>`
- Reusable **components, shadow-DOM**
- Class based: extend `<coq-notation>` for your purposes!
- Programmable with **JavaScript / TypeScript**

In alpha stage, **collaboration** with **Actema** as to define an interactive, **2-way model**

Summary

A **wide-scope** project, potentially **large impact**
Great opportunity to **collaborate** with several CS areas

- Bringing together **mathematical writing, formal logic, and collaboration research**
- Trying to match **today's** demands
- Important little step to **improve validation in science**
- Co-enrichment between PL and a **few other fields**
- **State of the art** PL proposals (gradual, incremental, **differential**)...

Reflecting the **reality** of a more and more **multidisciplinary** setup in computer science.