INSTITUT
DE RECHERCHE
EN INFORMATIQUE
FONDAMENTALE

LiberAbaci Kick-Off Meeting
Paris, September 20th 2022

JsCoq: Lessons and UI Perspectives

Emilio Jesús Gallego Arias

jsCoq's Early History

*JsCoq's: from toy to **research platform***

2013: Teaching assistant a Upenn, **lit. programming**
2015: Sends Coq to **`js_of_ocaml`** as a **joke**
2015: First version thanks to the **CUDW** *(P. Jouvelot)*
2015: Development of modern interface *(B. Pin)*
2015: Prime numbers example & packages
2016: Current protocol *(C. Pit-Claudel)*

"Opinionated" from the start, experimental

Original Design Philosophy

Interactive Literate Programming & Proving

Document at the center

start from a document, which is then *instrumented*

Lightweight

keep it simple, maintainable, **standards**-based

Server-less

servers **disappear**, self-contained **stays**

Not quite a "computational notebook"

More like *https://eloquentjavascript.net/*

2016-2019: Stagnation

***Ninety–ninety rule**: First **90%** of the code = first **90%** of dev time; last **10%** of the code = other **90%** of dev time*

Issues where varied:

- **Platform:** Browsers + `js_of_ocaml` (not **`wasm`**)
- **Focus:** Not my **main** research project
- **Coq:** Coq's *technical debt* bit hard! Lack of direction.
- **Expertise:** I knew little about UI programming, etc...
- **Users:** Barrier to contributions / lack interest

Quite frustrating "almost ready" state

2019-2022: Shachar's Epoch

"Mega-PR with a slew of new features #40"

[Quite a long list of improvements and fixes]

- Better runtime model, port to **WASM**, **worker** by default
- **UI improvements** (company, inspect, render, completion)
- Better **package management** and meta-data, **lazy loading**
- New **instrumentation** for coqdoc documents
- Node and **npm** support, **jscoq** github organization

We consider the tool "mature" now!

Positives and Negatives

What worked well?
What did not?

UI Work
Is ***not*** *forgiving*
Requires ***deep*** *rework*
Not easy to get ***help***

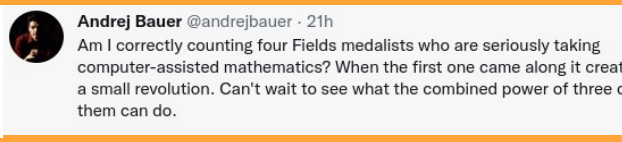
- **(+) Design Principles**: Maintenability, stability, experiments
- **(–) Coq:** API and design still way behind **"modern"** approaches
- **(–) User Workflow:** Not adapted to ***document writing practice***
- **(–) Tooling:** Required a **costly, painful** rewrite (ongoing)
- **(+) Platform/potential**: Very valious **feedback** and ideas

After 7 years, we finally have a good idea on how to do much better

Updating Assumptions: 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**!



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

ANR CoREACT

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, Jupyter, Curve**
- 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 + community feedback = jscoq.wiki!
Documents = math writing + collaboration + verification

- jsCoq **successful** project, but many needs not addressed
- **git-**based, new **rendering engine** to match new model
- Entry point for **H.C.I. research** collaborations and experiments
- Open to **other tools** in the eco-system!
  - **Catala**, formalized tax code and law, **Easycrypt**, formal Crypto, **Lambdapi**, logical framework with rewriting

Scope and problematics of user interaction **data-gathering** still under discussion, previous experiments *smaller scale* ***(CPP2019)***

Navigating `jscoq.wiki`

[jointly with S. Itzhaky, Ali Caglayan, Dune team, Deducteam, Ram R., ...]

Several Independent Components, Unified Architecture
Target a comprehensive solution for the Coq community

Flèche: hybrid document model and validation, **beyond** Coq
[incremental multi-layered model, incremental meta-data handling, whole project]

coq-lsp: Flèche + Coq Layout Engine + SerAPI + Coq.dev
[glue for display, reworked APIs, and document layer, target VsCode (also emacs!)]

jscoq.wiki: coq-lsp + **Curvenote** (prosemirror-based)
[HCI research, collaborative editing, most accessible point for users and educators]

Usable **today**, *first* release in a matter of ***days***

Despite our best practices, still a complex and big project map,
pushing the state of the art at several fronts

LiberAbaci: Research Challenges

Core focus: PhD in Programming Languages
Secondary focus: interactions with HCI

- **PL: Formal** extension of **type theory** towards:

  [All objects are handled uniformly]

  - **Richer mathematical vernacular; logical, meta-logical, human level**
  - **Layered vernacular:** Allow different languages and versions to co-exist
  - ***(Dynamic)*** **Incremental** and **soft** *checking* algorithms (DOM/React)
  - Document **evolution** and **collaboration:** semantic **CRDTs** and merge
- **PL+HCI: Incremental Reactive Elaboration.** Enable exploration!
- **HCI:** Quantify and understand user experience, **A/B testing**

Dynamic model subsumes **different roles** of document **data**
Exercises, **plots** (***à la Jupyter / Coq-Interval***) etc...

LiberAbaci: Engineering Challenges

Development of the Coq System: OCaml
Development of Interfaces: TypeScript

- Meta-data organization: **coq-db**
- Library organization and maintenance: **coq-universe / dune**
- **Web Development:** Curvenote / coq-layout-engine
- **Tool integration:** Build on the dynamic document interpretation
- **Standards**: LSP, Web Components, **OCaml 5**
- **User-support**, coordination beyond Coq, **educator** feedback

Dissemination and **Formation** essential activities too

Demos and Questions!

Thank you!




For more technical details join the upcoming **UI Working Group**





More on Flèche

Result of 5 years of research: still not in final form
Main influences: Isabelle, Dune, Dedukti (thanks to them)

- Essential features: dynamic DAG (== monad)
- All objects live in the same graph, egalitarian
- Good understanding of performance metrics
- Structural view, computational view

Incremental computation has large **tradeoffs** theory/practice

CurveNote

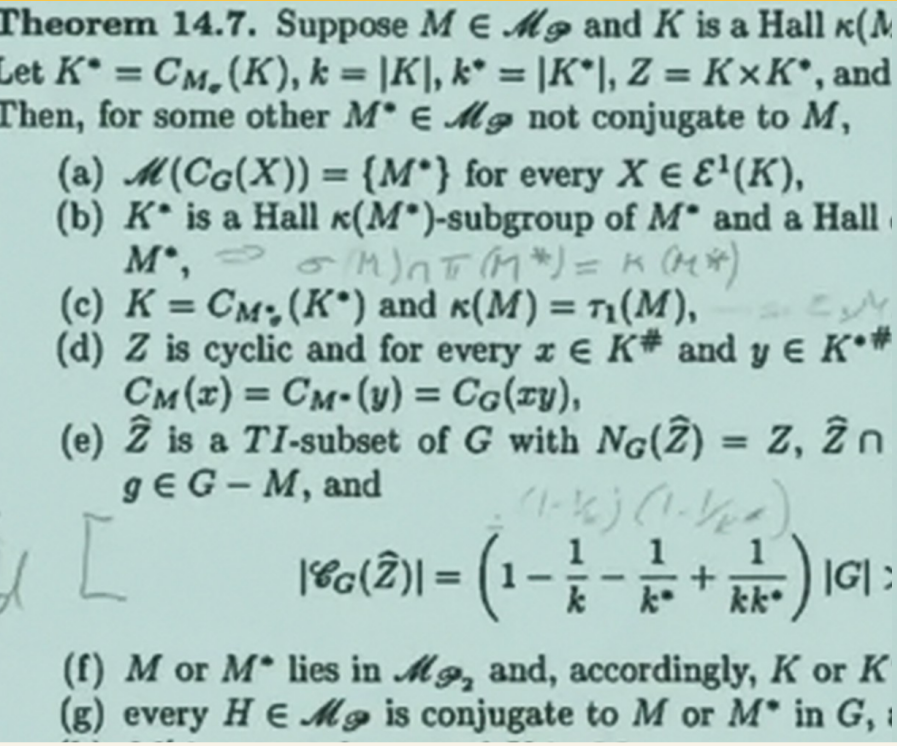
Platform for online scientific writing
Based on ProseMirror, well-proven, already used by us

- Extensible document schema
- Collaboration built-in, in a “classical sense”
- Fits very well with our goals
- Provides more than one document workflow
- Several import / export methods, main one MysT Markdown

Even if pretty minimalistic, still a complex piece of SW

Improving Coq's Printing

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



Theorem 14.7. Suppose $M \in \mathscr{M}_{\mathscr{P}}$ 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 $M^* \in \mathscr{M}_{\mathscr{P}}$ not conjugate to M,

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

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

(f) M or M^* lies in $\mathscr{M}_{\mathscr{P}_2}$ and, accordingly, K or K
(g) every $H \in \mathscr{M}_{\mathscr{P}}$ 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 \tau1(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| \/ 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

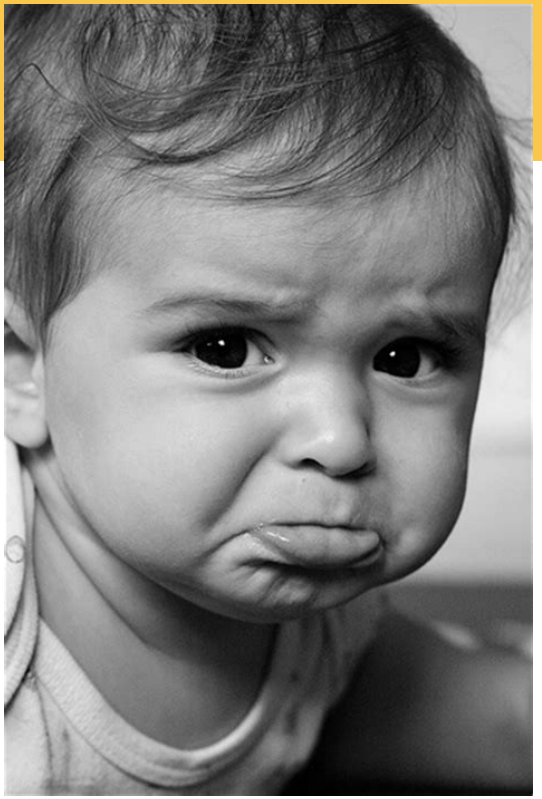
Standard by Google, 2015, well **supported**
Allows to define **custom tags** in the DOM

- **`<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**

Other (Applied) Challenges

Installing things!
Libraries that don't work / outdated proofs

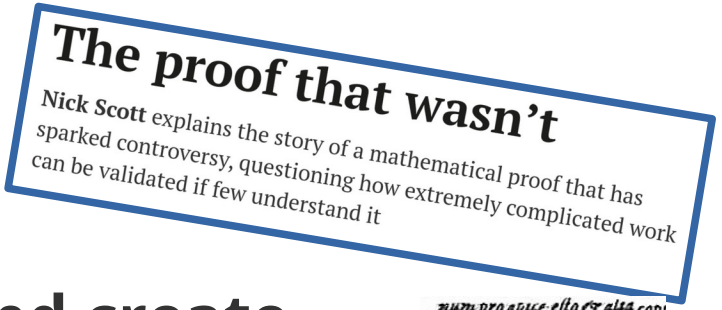


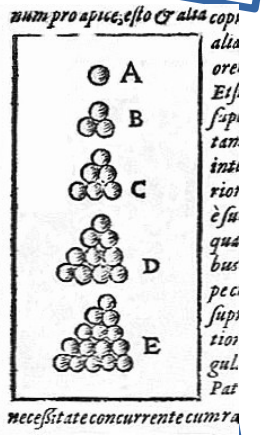
- **Searching** for things without success
- Bad **display** / **notations**
- Boilerplate / **trivial** proofs
- Synchronization / **merging** problems
- Lack of **documentation**
- **Dumb** or outdated **interfaces**

A mix of **Social**, **Research**, and **Engineering** Problems!

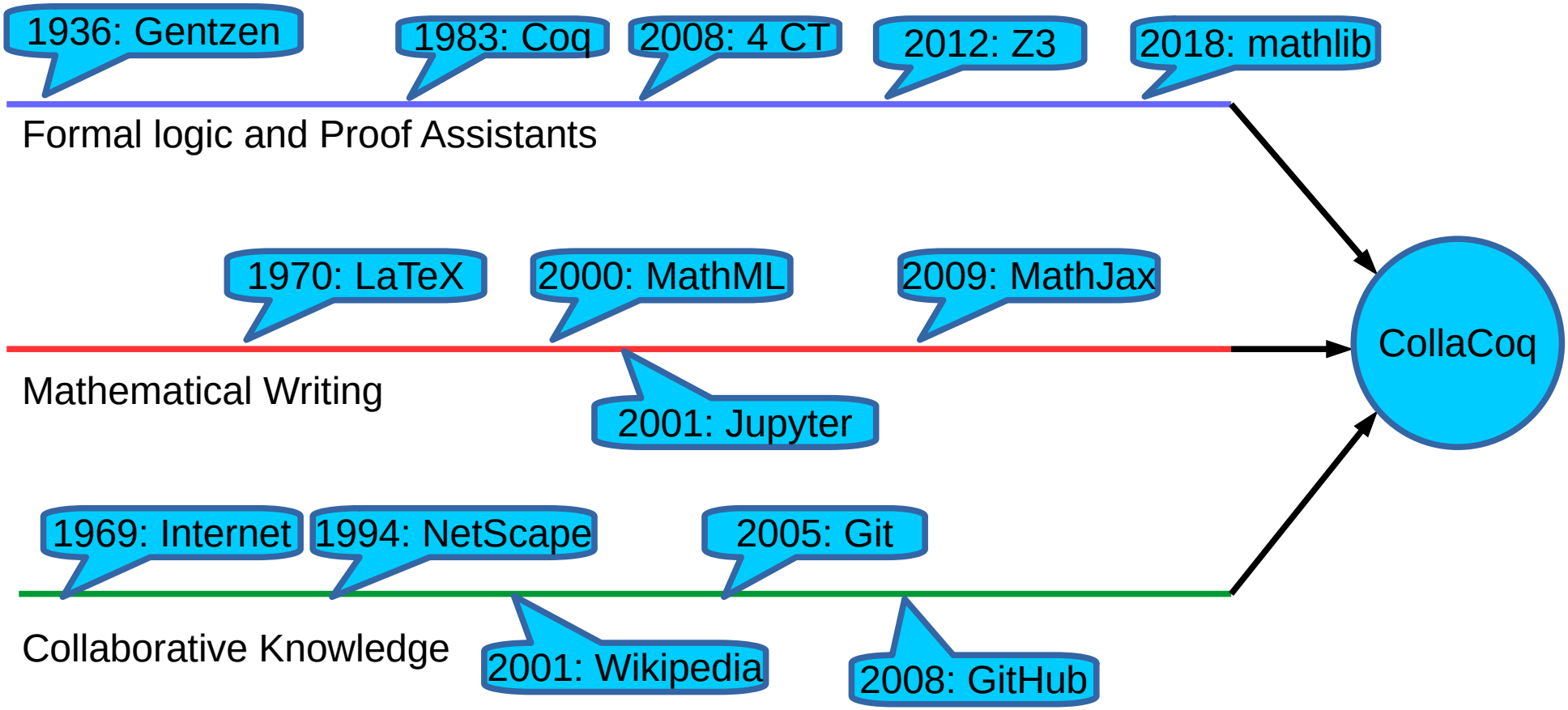
Scaling Formal Knowledge is Hard

- **Validity: complexity beyond human reach**
  - reviewer time
  - definition size and spread
- **Accessibility: hard to understand and create**
  - organization of knowledge
  - complex tools, complex content
- **Coordination: The Mythical Man-Month**
  - larger teams become less effective
  - social issues become apparent





Our Answer: jsCoq.wiki



Our Answer: jsCoq.wiki