jsCoq 2.0: Towards Rich Formal Documents

Emilio J. Gallego Arias & Shachar Itzhaky
(Picube – IRIF, Univ. Paris, Inria Paris)
LiberAbaci, Oct 18th 2024

From mathematical text to formal documents

Theorem 14.7. Suppose $M \in \mathscr{M}_{\mathscr{P}}$ and K is a Hall $\kappa(M)$-subgroup of M. Let $K^* = C_{M_\sigma}(K)$, $k = |K|$, $k^* = |K^*|$, $Z = K \times K^*$, and $\widehat{Z} = Z - (K \cup K^*)$. 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 $\sigma(M)$-subgroup of M^*,
(c) $K = C_{M^*_\sigma}(K^*)$ and $\kappa(M) = \tau_1(M)$,
(d) Z is cyclic and for every $x \in K^\#$ and $y \in K^{*\#}$, $M \cap M^* = Z = C_M(x) = C_{M^*}(y) = C_G(xy)$,
(e) $\widehat{Z}$ is a TI-subset of G with $N_G(\widehat{Z}) = Z$, $\widehat{Z} \cap M^g$ empty for all $g \in G - M$, and

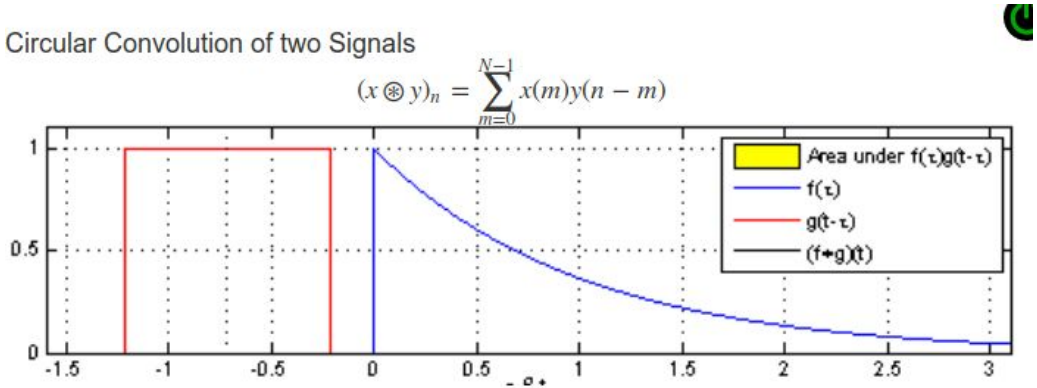
$$|\mathscr{C}_G(\widehat{Z})| = \left(1 - \frac{1}{k} - \frac{1}{k^*} + \frac{1}{kk^*}\right)|G| > \frac{1}{2}|G|,$$

(f) M or M^* lies in $\mathscr{M}_{\mathscr{P}_2}$ and, accordingly, K or K^* has prime order,
(g) every $H \in \mathscr{M}_{\mathscr{P}}$ is conjugate to M or M^* in G, and
(h) M' is a complement of K in M.

```
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).-Hc
     (*c*) 'C_(Mstar`_\sigma)(Kstar) = K /\ \kappa(M) =i \ta
     (*d*) [/\ cyclic Z, M :&: Mstar = Z,
           {in K^#, forall x, 'C_M[x] = Z}, {in Kstar^#, forall y
         & {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]}
```

(Figure credit: G. Gonthier)

From mathematical text to formal documents




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

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

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

jsCoq: history

jsCoq's birth wasn't planned!

- **2013:** Teaching assistant at UPenn, Software Foundations
- **2015:** Send to Coq to js_of_ocaml (joke) but it worked!
- **2015:** First version thanks to first CUDW (P. Jouvelot)
- **2015:** Benoit Pin develops first interface
- **2015:** Prime numbers example + packages
- **2016:** SerAPI, improved protocol (C. Pit-Claudel)
- **2019:** Shachar: worker, WASM, packs, company, print, node…

Very opinionated project from the start

jsCoq: original design philosophy

Interactive Literate Programming and Proving

Document at the center:

Start from existing document, instrument with Coq

Lightweight:

Keep it simple, maintainable, standards-based

Server-less:

Servers disappear, self-contained stays

jsCoq: demos

- https://www.youtube.com/watch?v=COukSOE5utA
- https://www.youtube.com/watch?v=IFCGzBDTpCo
- https://eloquentjavascript.net/11_async.html
- First talk(s)
- https://coq.vercel.app/
- https://github.com/jscoq/jscoq/#examples

jsCoq “1.0” architecture: frontend

Coq manager:
Coordinates events coming from HTML page and Coq

Layout manager:
Controls the jsCoq panel

Package manager:
Manages package loading

CodeMirror Provider:
Presents a set of CM5 editors as single document

The CM provider parses the Coq document and submit sentences to Coq; goals are displayed on request

jsCoq “1.0” protocol

```
type jscoq_cmd =
  | InfoPkg of string * string list
  | LoadPkg of string * string
  | Init    of jscoq_options
  | NewDoc  of doc_options
  | Add     of Stateid.t * Stateid.t * …
  | Cancel  of Stateid.t
  | Exec    of Stateid.t
  | Query   of Stateid.t * route_id * query
  | Ast     of Stateid.t
  | Register of string
  | Put      of string * string
  | GetOpt  of string list
  | InterruptSetup of opaque
  | ReassureLoadPath of lib_path
  | Load    of string
  | Compile of string
```

```
type jscoq_answer =
  | CoqInfo   of string
  | Ready     of Stateid.t
  | Added     of Stateid.t * Loc.t option
  | Pending   of Stateid.t * string list * string list
  | Cancelled of Stateid.t list
  | ModeInfo  of Stateid.t * in_mode
  | GoalInfo  of Stateid.t * Pp.t reified_goal option
  | Ast       of Vernacexpr.vernac_control option
  | CoqOpt    of string list * Goptions.option_value
  | Log       of Feedback.level * Pp.t
  | Feedback  of Feedback.feedback
  | SearchResults of route_id * Qualified_name.t Seq.t
  | Loaded    of string * Stateid.t
  | Compiled  of string
  | CoqExn    of { loc : Loc.t option … }
  | JsonExn   of string
```

jsCoq “1.0”: success and limitations

---

jsCoq proved popular!

- **success:** working reasonably well, real-world proofs
- **success:** many cool features! (auto-load, collab, inspect)
- **success:** serverless approach passed the test of time

- **limitation:** Document model and editor support
- **limitation:** Coq API / STM
- **limitation:** Class preparation workflow
- **limitation:** maintenance / complexity

jsCoq “2.0” and Flèche

- **Context:** looking at the problem since 2016
- **Motivation:** late 2021, SF / CoREACT / Waterproof / others
- **Goal:** New use cases & fix many existing problems
- **How:** Flèche / coq-lsp (long time wish, long time research)
- **Improvements:**
  - Maintainability: Flèche is simple, sound, and extensible
  - Modularity, unification: Reusable components, several editors
  - New features: full-project, native hybrid, incremental, declarative
- **jsCoq 2.0:** platform for **research**, **education**, and **experimentation**

jsCoq “2.0”: Main novelties and changes

- **Frontend:** ported to typescript, greatly simplified
- **Frontend:** support several editors

- **Backend:** custom -> LSP + a few extensions
- **Backend:** built by coq-lsp CI

- **General:** native hybrid document / theory model
- **General:** large improvements on infra, CI
- **General:** improvement on client/server FS model

Flèche: Maintenance, Extensibility

- **Stable** core engine (0 open core bugs)
- Most problems or features: prototypes / workarounds exist

- **CoREACT graph editor** **(A. Lafont)**
- **CoqPilot** **(A. Kozyrev, A. Podkopaev)**
- **ViXZ Visualizer** **(B. Shah)**
- **Waterproof** **(J. Portergies, Waterproof team)**
- **petanque** **(G. Baudart)**
- **jsCoq 2** **(S. Itzhaky)**
- **ERooster** **(S. Itzhakty, E. Singer)**
- **Coq-universe** **(A. Caglayan)**
- **Python, VIM, Emacs, clients** **(several contributors)**

Flèche: interaction modes

Document Updates

- Flèche **scans the workspace** for proof documents and config
- Users / editors **just relay changes** on the documents
- Flèche decides what to do on those, maybe nothing!
- No more need for a build system

Document Queries

- Document-level positions constitute **ground-truth**
- User can query for specific objects in a set of documents
- Then Flèche will decide how to return these objects ASAP
- That’s the base for the upcoming plugin interface

Document interaction as a build system at the core of the philosophy

Do we need jsCoq anymore?

- github.dev demo

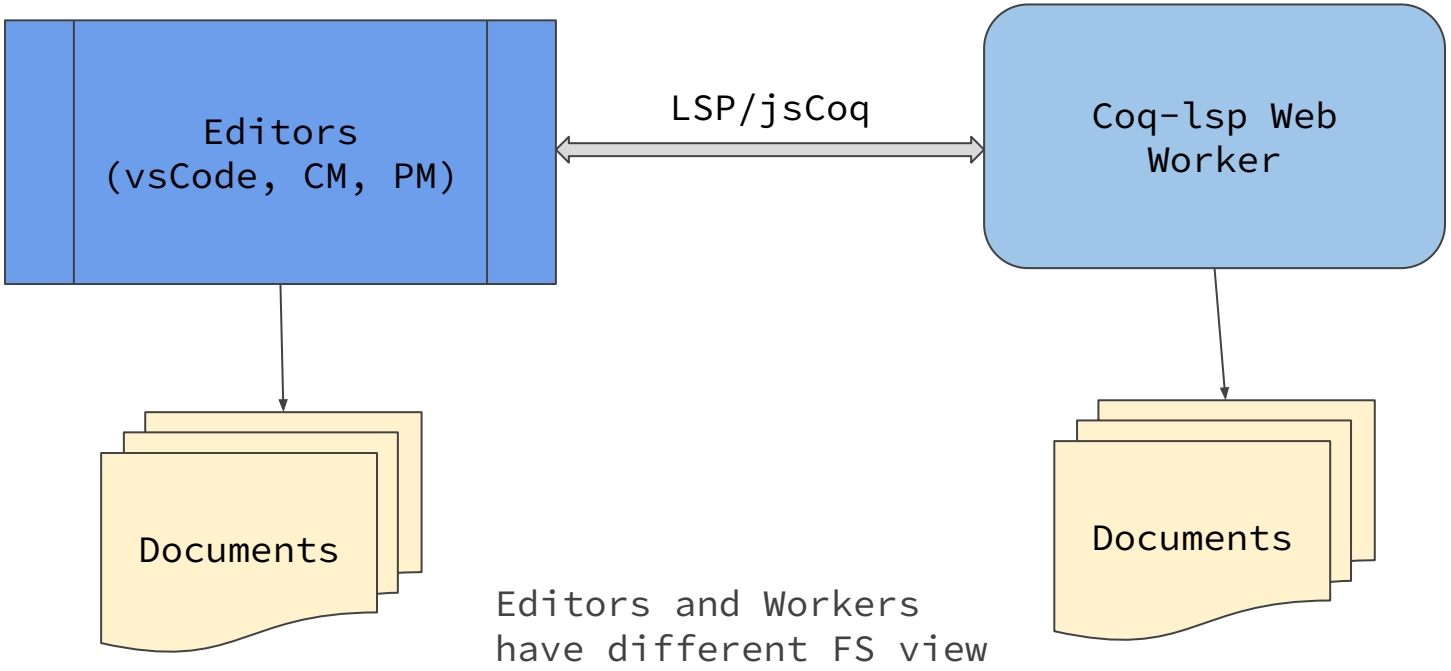
- **Flèche** unifies all the platforms
- All you need now is **to drop a git repos** into Flèche
- **VSCode custom editors:** jsCoq becomes really minimal
- Exploring this question with WaterProof devs

jsCoq “2.0”: protocol extensions

- https://github.com/jscoq/jscoq/issues/377

- Range-based offsets (LSP?)
- Package management
- Interruption setup (LSP?)
- Virtual FS (planned for LSP)

jsCoq “2.0”: virtual FS



jsCoq “2.0”: editors

- Demo of CodeMirror, ProseMirror, Curvenote

jsCoq “2.0”: addons / packages

- As of today we distribute packages in a zip file
- Not clear how to best organize this
- We require some extra metadata to implement **auto-require**

jsCoq “2.0”: TODO

- **Coq patches:** help would be nice
- **64-bit build:** 32-bit builds are going away
- **Package manager:** need to figure out a robust solution
- **Web development:** interface to expose the FS properly
- **Documentation:** seems that this needs a lot of improvement
- **Packaging:** to npm, other forms?
- **OCaml + WASM:** exploring upstream integration

Lots more ideas!

Discussion Time

Thanks !