Adapting Coq-Lsp for Lambdapi

A brief history of Lambdapi Lsp

- *The code is in the same repository of Lambdapi*
- *The Server is in* `src/lsp` *and is written in Ocaml, installs with opam*
- *The clients are in the* `editors` *folder.*
- *The focus is currently on Vscode extension written in Typescript (extension for Emacs and VI exist also).*
- *Can be installed from Vscode marketplace.*

A brief history of Lambdapi Lsp

- *Lambdapi v.2.5.1 released on July, 22nd 2024*
  - *Bug fixes*
  - *Enhance messages in the terminal and navigation in document*
- *Vscode extension : V0.2.2 released on may 16th 2024*
  - *Bug fixes*
  - *Use latest Vscode librairies*
  - *Reopen the goals panel or bring it back to front when navigation proof*

Coq-Lsp

What is it about?

- *Coq-Lsp is an Lsp server for Coq with interesting features to interactively develop proofs in Coq (incremental compilation, Document outline, ...)*
- *LambdaPi and Coq are very similar ...*
- *Why not reuse the Coq-Lsp code so to benefit from the features it implements*
- *Moreover, on the long term, it is more efficient to rely on Coq-Lsp to benefit from the existing support, maintenance, ... the Coq-lsp community offers.*

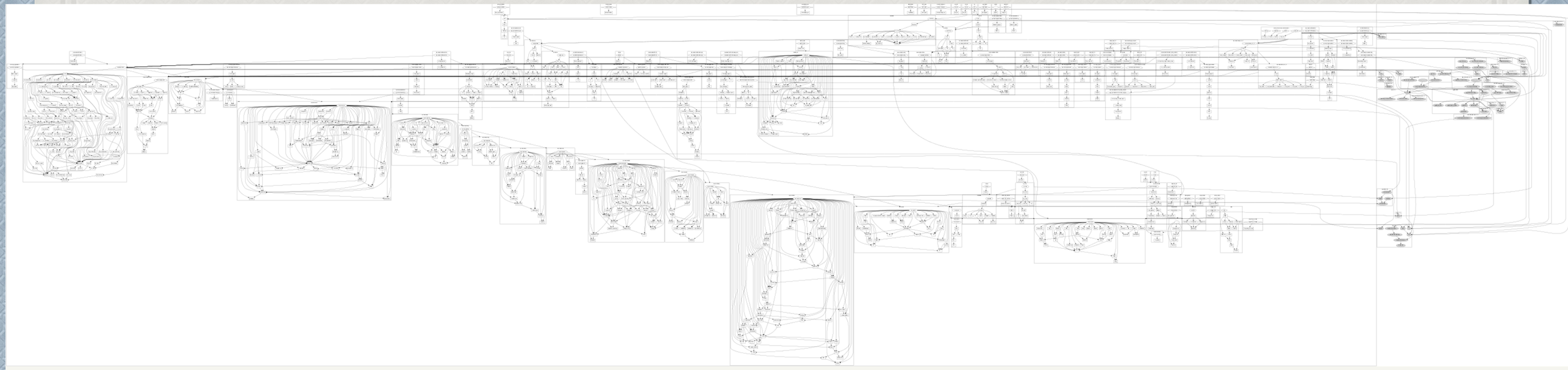
Work plan

- *Identify the modules of Coq-Lsp, their inter-communication API and the global API*
- *Refactor the code to make it adaptable with Lambdapi :*
  - *Separate commun code and specific code*
  - *Ideally, move the specific code to the modules in the bottom layer of architecture (the ones that directly interact with the prover (Coq or Lambdapi)*
- *Write the modules specific to Lambdapi*
- *Write the Glue code : the code that selects the right specific modules*

How to identify functional modules and API in OCaml

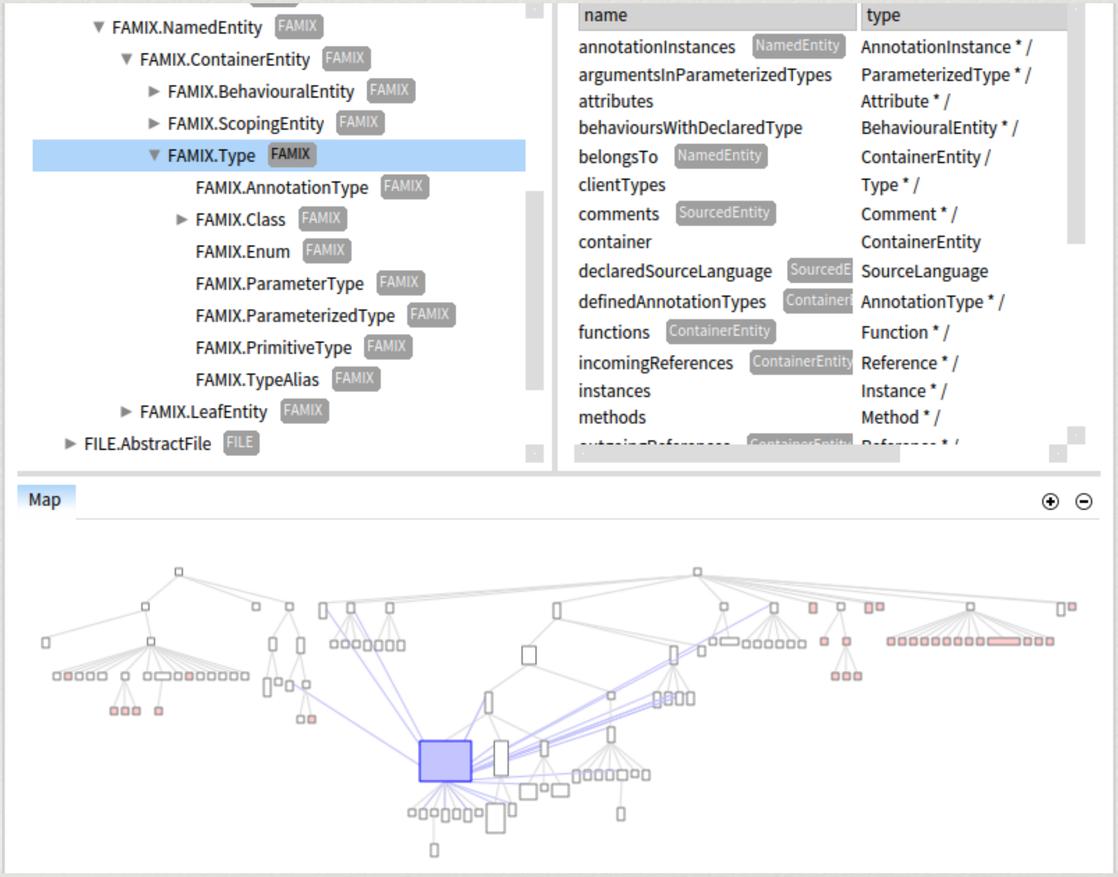
- *Specifically, understanding the Non functional aspects of the code.*
- *Some text-based tools exist to analyse OCaml code and extract information*
  - *dune-deps, not-ocamlfind, depgraph, module-graph, odoc-depgraph, ...*

Odep to understand dependencies



- *Quit rudimentary*
- *Not all useful information (at least the one that interest me) is extracted.*
- *Information is not well presented : no customization, overloaded, ...*

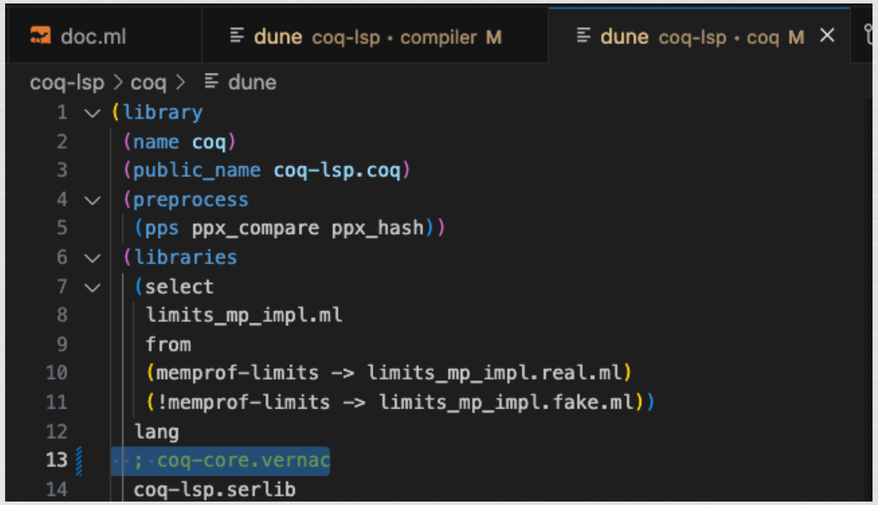
Code Analysis and visualization

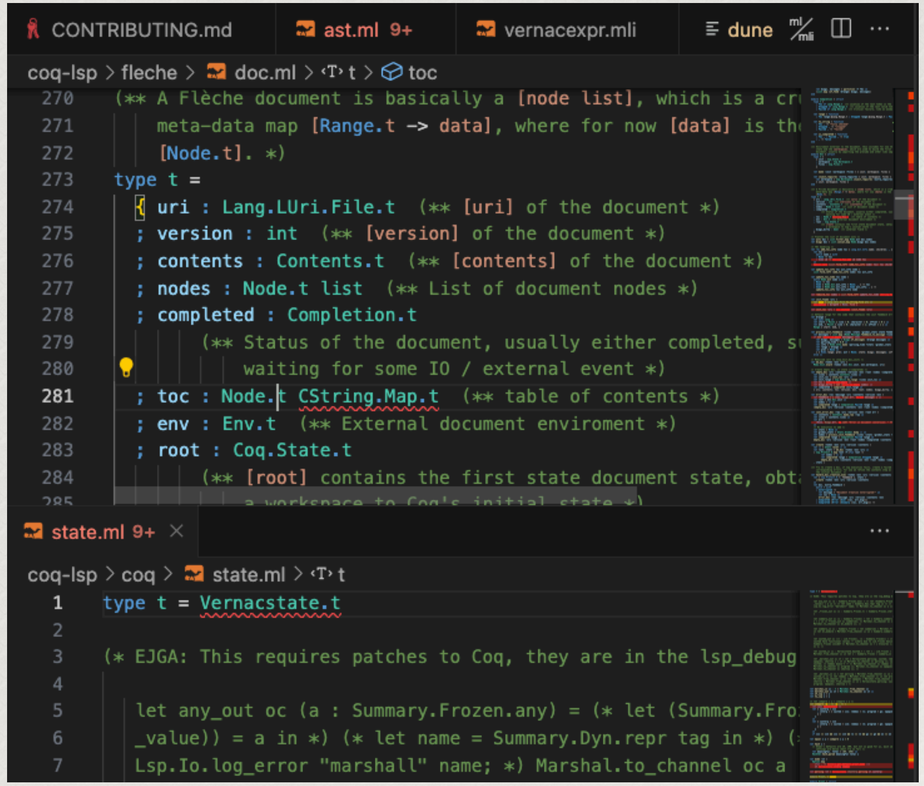


How to identify functional modules and API in OCaml

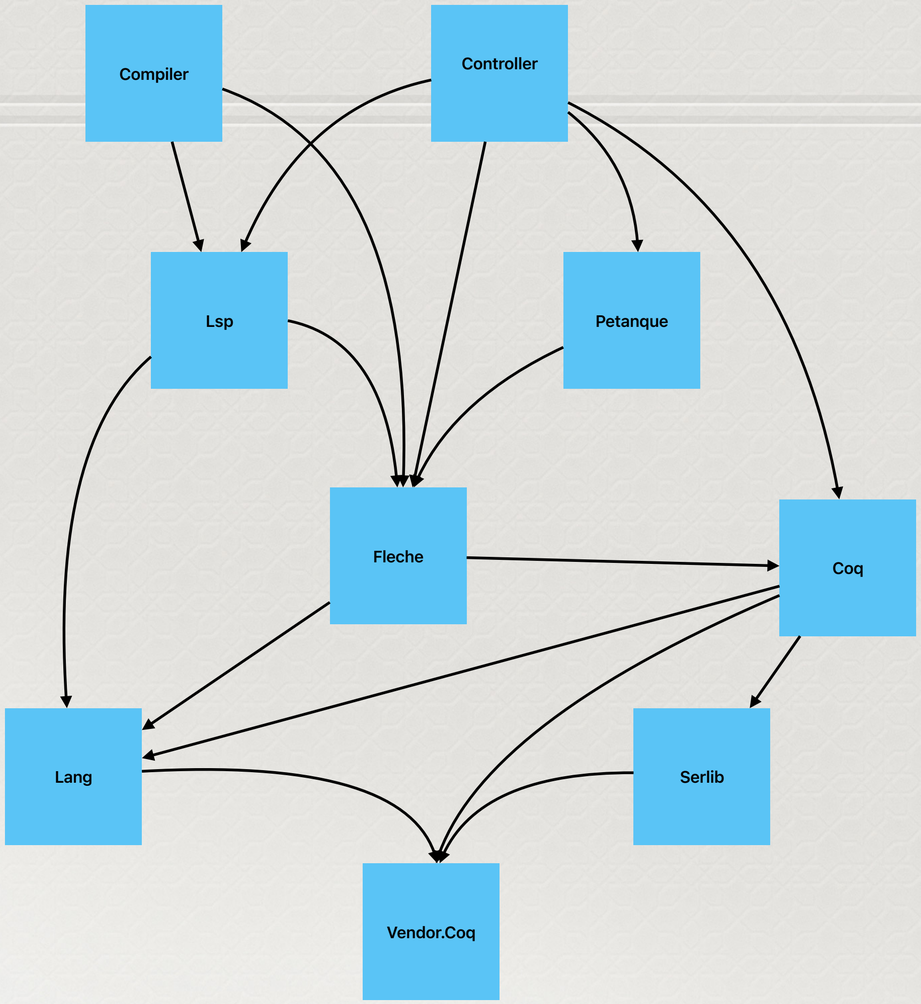
- *By hand*
  - *Read code*
  - *Use .mli files*
  - *Extract APIs and determine dependencies*
  - *Cumbersome for large code*
- *TIP : change the Dune files and let the compiler highlight the dependencies*

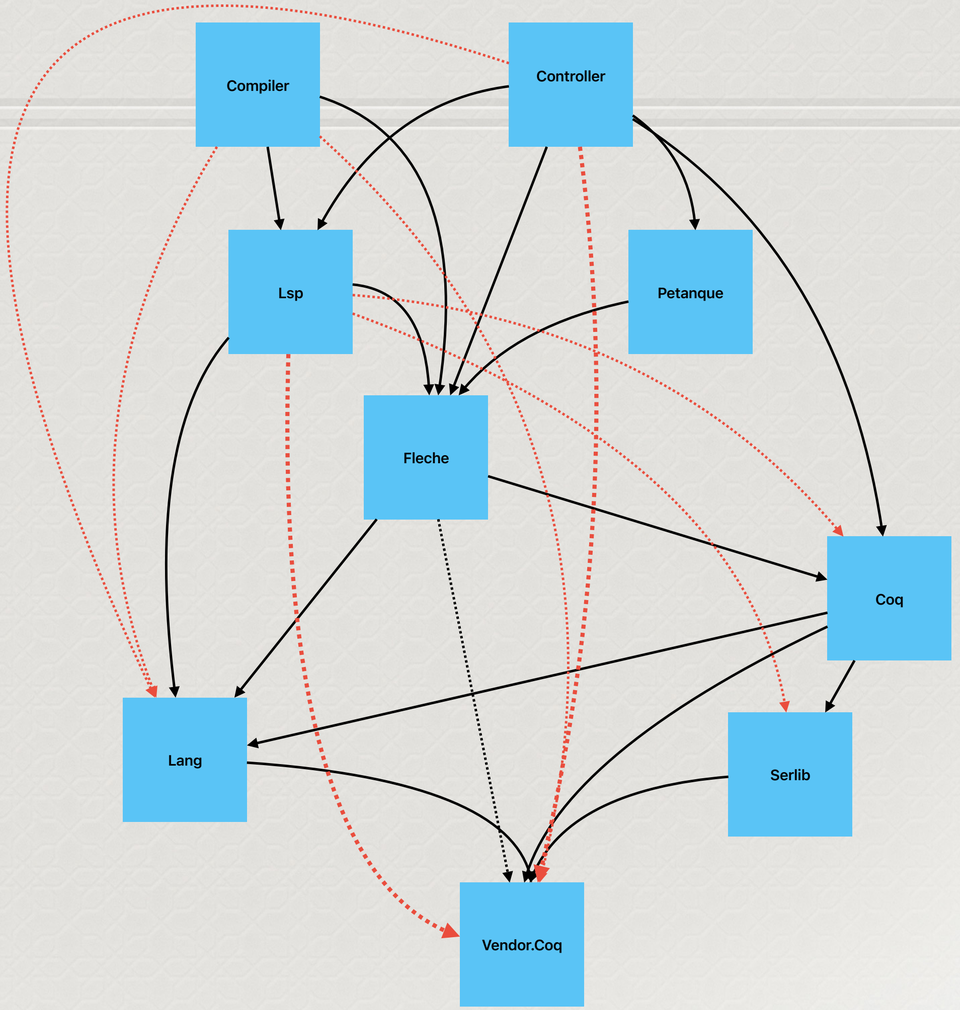
Detecting dependencies





The package diagram of Coq-Lsp





Refactoring the code

```
type fleche_document_type = {
  ... (* independent from prover *)
}
module type ProverDocument = sig
  type fleche_document_type
  type prover_document_type
  val specific_function :
end
module FlecheDocumentFunctor =
functor(P:ProverDocument) -> struct
  type t = P.t
  type prover_document_type = P.prover_document_type
  let specific_function = P.specific_function
  let commun_function = (* Commun code here *)
end
```

```
type lambdapi_specific_type = {
  ...
}

module LambdapiDocument = struct
  type t = commun_t
  type source = Lambdapi.LambdapiDocument.t
  let specific_lambdapi_function = ...
end
```

```
module FlecheDocumentForLambdaPi = FlecheDocumentFunctor(LambdapiDocument)
FlecheDocumentForLambdaPi.specific_function ...
FlecheDocumentForLambdaPi.commun_function ...
```

Lessons learnt

- *Open projects are great*
- *A gap may exist between what the Readme says and what the code looks like.*
- *Investing in open code can pay at the mid and long terms. But,*
  - *Can be non negligible.*
  - *Many parameters determine if it worths it :*
    - *Quality of code*
    - *Documentation*
    - *Community engagement in the project and openness*

Future of development

- *Coq-Lsp developers are willing to evolve it to work with other Provers*
- *What can be done on our side :*
  - *Leave it to them and focus on LambdapiPi specific code*
  - *Contribute to the refactoring : Documentation (models), specific and non specific code.*
  - *Fork the project.*

Questions?