Statically typed API for Structured Documents

Gabriel RADANNE – Inria CASH/LIP
Martin BODIN – Inria Grenoble

2025-2026

1 Context

Numerous document formats have a notion of *validity*. For instance, in HTML, a link <a> can't itself contain links, the immediate children of a `<table>` should be rows `<tr>`, etc. There are also constraints on attributes: in SVG, the `transform` attribute can appear on a group <g>, but not on a `<symbol>`.

Schemas languages have been developed to formalize these rules in a way that can be checked automatically. For instance DTDs[1] allow to describe the rules governing an XML document. Numerous schemas languages exist for various data and serialisation formats. Here is an example of a DTD:

```
<!ATTLIST img
   title CDATA  #IMPLIED
   data  ENTITY #IMPLIED
   type  NOTATION (type-image-svg | type-image-gif) #IMPLIED>
```

It would validate a piece of XML such as `<img title="&example1SVGTitle;" data="example1SVG" />`

However, in most cases, documents are not created by hand, they are emitted by programs using libraries. These libraries expose an API which presents the different elements of the language. For instance, TyXML[2] is an OCaml library for creating HTML and SVG documents. It contains functions like `table` or `td`. Its remarkable feature is that it *checks the validity of generated documents statically*: most of the validity constraints are encoded *in the OCaml type system*, by exploiting its rich expressivity. This enables writing programs, such as the following, and ensure that the emitted document is always valid:

```
let mycontent (title : string) (content : string) =
  div ~a:[a_class ["content"]] [
    h1 [txt title] ;
    txt content
  ]
```

For example, if we try to put **bold** text in an HTML title (which is forbidden by the HTML specification), we obtain

```
let mytitle = title (b [txt "A Bold Web Page"])
                    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

Error: This expression has type [> `B ] elt
       but an expression was expected of type [< `PCDATA ] elt
       The second variant type does not allow tag(s) `B
```

In OCaml's typechecker language, this means that you gave an HTML element of type `` `B `` (i.e. the bold tag) and it expects some `` `PCDATA `` (i.e. normal text).

[1] https://en.wikipedia.org/wiki/Document_type_definition
[2] https://github.com/ocsigen/tyxml

LIP – UMR CNRS / ENS Lyon / UCB Lyon 1 / Inria – 69007 Lyon
E-mail : gabriel.radanne@inria.fr and martin.bodin@inria.fr

2 Internship objective

Encoding such validity rules in a type-system requires fairly complex and extensive type annotation. For TyXML, this work was done by hand. This is expensive to maintain, and difficult to reproduce for other family of documents.

The goal of this internship will be to *automatically* generate a TyXML-like library, given a schema (DTD or otherwise). DTD can contain very complex rules, so we will grow the complexity of the encoding progressively. We will start with immediate parent-children constraints and progress towards complex constraints between content and attributes. We may consider several extensions, depending on the taste of the intern:

- A well-designed code-generator for OCaml that can be distributed to users, possibly with the option of controling the complexity (and expressiveness) of the generated code,
- Alternative formats (like JSON) and/or schemas languages (like XSD)
- Encoding in richer type systems (notably, dependent types in Rocq)

3 Internship

The internship will take place either in Lyon (Team CASH, LIP Laboratory) or in Grenoble (Team Spades, Inria Grenoble).

Candidate profile The candidate should ideally be familiar with formal approaches in programming language design. They should also have a taste for functional programming and rich type systems.

From the practical point of view, a basic experience in software programming and usage of collaborative tools such that `git`. Knowledge of OCaml is strongly recommended.