Artifact for "A Core Calculus for Documents"

This repository contains the accompanying artifact for the POPL'24 paper "A Core Calculus for Documents" (Crichton and Krishnamurthi).

1 Overview

The core contribution of our paper is a formal model of document languages. This model extends System F with the concept of templates. We provide static and dynamic semantics for templates, and we also model various run-time extensions to the system such as reference labeling and reactivity.

This artifact implements each aspect of the model in OCaml. Our main goal in the implementation is to *correctly* and *clearly* implement the model. Then this model can facilitate authors of document languages to pick and choose ideas that are relevant to their setting, especially those less familiar with PL theory notation. Each of the source files has been liberally commented to explain the purpose of the code, and to point a reader to the relevant pieces.

Claims: Sections 3, 4, and 5 all contain elements that are implemented in the artifact. See "Evaluation Instructions" below for details on which modules correspond to which subsections.

2 Getting Started

2.1 From Docker

If you are downloading a Docker image from Zenodo, first: make sure to pick the image that corresponds to your architecture. That's `arm64` for an ARM machine (e.g., an M-series Mac), and `amd64` otherwise.

Then run the following command, replacing `<ARCH>` with either `arm64` or `amd64`:

```
docker load -i document-calculus-popl24-<ARCH>.tar.gz
```

Then start the image and run the tests as follows:

```
$ docker run -ti document-calculus-popl24:<ARCH> bash
opam@4fb288bee717:/app$ opam exec -- dune test
```

2.2 From Source

2.2.1 I'm in a hurry

You need opam. Run this script:

```
opam switch create 4.13.1 --yes
eval $(opam env --switch=4.13.1)
opam install . --deps-only --locked --yes
opam exec -- dune test
```

2.2.2 I have some time

In greater detail: first, you need the OCaml package manager opam. This artifact was last tested with opam version 2.1.3.

Then, you need OCaml. This artifact was last tested with OCaml version 4.13.1. You can install the version by running:

```
opam switch create 4.13.1
```

Then install the project's dependencies by running:

```
opam install . --deps-only --locked
```

Finally, make sure the tests pass by running:

```
opam exec -- dune test
```

3 Evaluation Instructions

To evaluate this artifact, you should check that the paper representation of the model matches the OCaml representation of the model. I recommend doing so in the following order:

- Section 3.1.1: the `DStrLit` module in `lib/string.ml`.
- Section 3.1.2: the `DStrProg` module in `lib/string.ml`.
- Section 3.1.3: the `DStrTLit` module in `lib/string.ml`.
- Section 3.1.4: the `DStrTProg` module in `lib/string.ml`.
- Section 3.2.1: has no implementation in the code.
- Section 3.2.2: the `DArtProg` module in `lib/article.ml`.
- Section 3.2.3: has no implementation in the code.
- Section 3.2.4: the `DArtTProg` module and the `DArtTProgNested` module in `lib/article.ml`, and the `Node` module in `lib/extensions.ml`.
- Section 4.1: the `References` module in `lib/extensions.ml`.
- Section 4.2: the `Reforestation` module in `lib/extensions.ml`.
- Section 4.3: the `Reactivity` module in `lib/extensions.ml`.
- Section 5.1: the `typecheck_template` functions in `lib/string.ml` and `lib/article.ml`.

4 Additional Artifact Details

The main source files are all in `lib/`. The other files correspond to these categories:

4.1 Tests

To test that the artifact has no basic errors, we have written a number of unit tests in `test/tests.ml`. You can run these tests with the following command:

```
opam exec -- dune test
```

You can also edit `tests.ml` and add your own test if you want.

4.2 Playground

We have provided a `bin/main.ml` file for playing around with the library. There is already a sample program there you can run, like this:

```
$ opam exec -- dune exec bin/main.exe
A sample program:
  "hello" + " world"

It evaluates to:
  "hello world"
```

You can try modifying `main.ml` and rerunning it with different programs.