Getting started guide

flap

A deterministic parser with fused lexing

Artifact evaluation

The flap authors

11th April 2023

Running the artifact

The artifact is a Docker image. We have tested it with Docker version 20.10.14, but we expect it to work with any recent version of Docker.

Carry out the following steps to run the artifact:

1. Unpack the image using `docker load`:

   ```
   docker load < flap.tgz
   ```

   On success Docker displays the image name:

   ```
   Loaded image: flap:latest
   ```

2. Run the Docker image in interactive mode:

   ```
   docker run -it --rm -v local:/local flap}
   ```

   Here `-v local:/local` stipulates that the host directory `local` should be available in the running container as `/local`. Ensure that the `local` directory exists on the host before running the image.

   You should see a prompt like this:

   ```
   opam@b7c56f938d18:~/flap$
   ```

   (The container id, here `b7c56f938d18`, will vary from run to run.)

Basic testing

1. Check that you can read the `flap` directory by typing:

   ```
   ls
   ```

   You should see the contents of the `flap` directory:

   ```
   Makefile benchmarks  dune-project  flap.opam  grammars  lib  tests
   ```

2. Check that you can build `flap` by typing:

   ```
   make
   ```

   The system should display the build instruction:

   ```
   dune build -p flap
   ```

   While the build is running you should see an updating progress report:

   ```
   Done: 90% (152/168, 16 left) (jobs: 2)
   ```

3. Check that you can build one of the `flap` benchmarks by typing:

   ```
   make csv-bench
   ```

   Ignore the `unused variable` warnings. The benchmark will take a little over 2 minutes to run.

   Among other output, the benchmark should display a table showing running times for various implementations (`ocamlyacc`, `menhir_code`, `menhir_fused`, `fused`) and various sizes of data (16, 32, etc.)

   ```
    Name                      Time R^2      Time/Run                 95ci
   ---------------------- ---------- ------------- ---------------------
    ocamlyacc_csv:16            1.00      650.82us       -2.57us +2.60us
    ocamlyacc_csv:32            1.00    1_308.21us       -3.78us +4.28us
    ...
   ```

   If the benchmark finishes successfully the final line of output should be as follows:

   ```
   mkdir -p paper/csv && ./benchmarks/munge.py < benchmark-csv.out > paper/csv/csv.csv
   ```

4. Check that when you create or change a file in the directory `local` on the host, the changes are visible (e.g. via `ls /local`) in the container.

5. Try out the library interactively.

   (a) Start the MetaOCaml toplevel

   ```
   metaocaml
   ```

   (b) Within MetaOCaml load the `topfind` command, then load `flap`:

   ```
   #use "topfind";;
   #require "flap";;
   ```

   On success the last line of output should be as follows:

   ```
   /home/opam/.opam/4.11.1+BER/lib/flap/flap.cma: loaded
   ```

   (c) If you still have time you might like to continue with the example in the *Step by Step* guide, which shows how to create a fused parser with `flap`.