The Concise Common Workflow Language

Arun Isaac

Department of Computational and Data Sciences
Indian Institute of Science, Bengaluru – 560012

October 8, 2021

Why workflow languages?

Why not just use shell scripts?

- Separate housekeeping code from actual processing
- Isolate inputs, outputs and steps
- Better error reporting on failed steps
- Automatically handle running in different software and hardware environments (containers, clusters, etc.)
- Distinguish between string inputs and file inputs
- Human readable and machine inspectable language

CWL vs ccwl

The Common Workflow Language and the Concise Common Workflow Language

Common Workflow Language

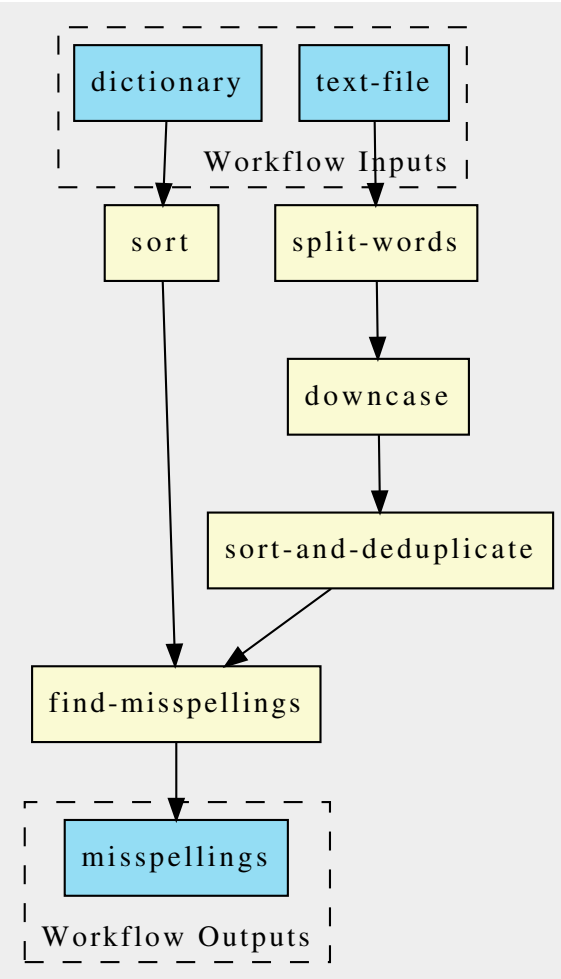
- A CWL YAML specification is too verbose.
- Too many files! Each step has to have its own CWL file. And these need to be wired up together into a workflow CWL file.

Concise Common Workflow Language

- Reduce the verbosity of CWL by auto-generating most of it
- To the user, writing ccwl should be as simple as writing a shell script, or at least, a Makefile
- Good compile-time warnings so errors can be caught early

Demo

A spell check workflow[1]



[1] Example from dgsh; see `https://github.com/dspinellis/dgsh`

Prior art

Spell check in dgsh, the directed graph shell

```
{{
    tr -cs 'A-Za-z' '\\n' |
    tr 'A-Z' 'a-z' |
    sort -u
    sort /usr/share/dict/words
}} |
comm -23
```

Summary

Rationale

Why workflow languages? Automatically handle housekeeping tasks (managing intermediate files, checking for success of sub-steps, etc.)

Why CWL? Ability to reason about workflows, generate graphical representations without running them

Why ccwl? Much more concise, and easier to write than CWL. Aims to be as easy to write as a shell script or Makefile.

```
(define split-words
  (command #:inputs file
           #:run "tr" "--complement"
           "--squeeze-repeats" "A-Za-z" "\\n"
           #:stdin file
           #:outputs (words #:type stdout)))

(define downcase
  (command #:inputs file
           #:run "tr" "A-Z" "a-z"
           #:stdin file
           #:outputs (downcased #:type stdout)))
```

```
(define sort-and-deduplicate
  (command #:inputs file
           #:run "sort" "--unique"
           #:stdin file
           #:outputs (sorted-and-deduplicated
                      #:type stdout)))

(define sort
  (command #:inputs file
           #:run "sort" file
           #:outputs (sorted #:type stdout)))
```

```
(define find-misspellings
  (command #:inputs words dictionary
           #:run "comm" "-23" words dictionary
           #:outputs (misspellings #:type stdout)))
```

```
(workflow (text-file dictionary)
  (pipe (tee
          (pipe (split-words #:file text-file)
                (downcase #:file words)
                (sort-and-deduplicate
                 #:file downcased))
          (sort #:file dictionary))
         (find-misspellings
          #:words sorted-and-deduplicated
          #:dictionary sorted)))
```