The Concise Common Workflow Language

Concision and elegance in a workflow language using lisp

Arun Isaac

February 6, 2022

Why workflow languages?

Why not just use shell scripts?

Why workflow languages?

Why not just use shell scripts?

What does this script do?

```
input_file=$1
if [ ! -f "$input_file" ]; then
    echo "Input file does not exist"
    exit 1
fi
output_file=$2
mkdir /tmp/working-directory
cp "$input_file" /tmp/working-directory/input
wc -w /tmp/working-directory/input > /tmp/working-directory/words
if [ $? -ne 0 ]; then
    echo "Command wc -w failed"
    exit 1
fi
rm /tmp/working-directory/input
mv /tmp/working-directory/words "$output_file"
rmdir /tmp/working-directory
```

Why workflow languages?

Why not just use shell scripts?

Seems to take two arguments

```
input_file=$1
if [ ! -f "$input_file" ]; then
    echo "Input file does not exist"
    exit 1
fi
output_file=$2
mkdir /tmp/working-directory
cp "$input_file" /tmp/working-directory/input
wc -w /tmp/working-directory/input > /tmp/working-directory/words
if [ $? -ne 0 ]; then
    echo "Command wc -w failed"
    exit 1
fi
rm /tmp/working-directory/input
mv /tmp/working-directory/words "$output_file"
rmdir /tmp/working-directory
```

Why workflow languages?

Why not just use shell scripts?

Check input files and input types

```
input_file=$1
if [ ! -f "$input_file" ]; then
    echo "Input file does not exist"
    exit 1
fi
output_file=$2
mkdir /tmp/working-directory
cp "$input_file" /tmp/working-directory/input
wc -w /tmp/working-directory/input > /tmp/working-directory/words
if [ $? -ne 0 ]; then
    echo "Command wc -w failed"
    exit 1
fi
rm /tmp/working-directory/input
mv /tmp/working-directory/words "$output_file"
rmdir /tmp/working-directory
```

Why workflow languages?

Why not just use shell scripts?

Report error if command failed

```
input_file=$1
if [ ! -f "$input_file" ]; then
    echo "Input file does not exist"
    exit 1
fi
output_file=$2
mkdir /tmp/working-directory
cp "$input_file" /tmp/working-directory/input
wc -w /tmp/working-directory/input > /tmp/working-directory/words
if [ $? -ne 0 ]; then
    echo "Command wc -w failed"
    exit 1
fi
rm /tmp/working-directory/input
mv /tmp/working-directory/words "$output_file"
rmdir /tmp/working-directory
```

Why workflow languages?

Why not just use shell scripts?

Isolate inputs, outputs and steps

```
input_file=$1
if [ ! -f "$input_file" ]; then
    echo "Input file does not exist"
    exit 1
fi
output_file=$2
mkdir /tmp/working-directory
cp "$input_file" /tmp/working-directory/input
wc -w /tmp/working-directory/input > /tmp/working-directory/words
if [ $? -ne 0 ]; then
    echo "Command wc -w failed"
    exit 1
fi
rm /tmp/working-directory/input
mv /tmp/working-directory/words "$output_file"
rmdir /tmp/working-directory
```

Why workflow languages?

Why not just use shell scripts?

Actual code is just a single command!

```
input_file=$1
if [ ! -f "$input_file" ]; then
    echo "Input file does not exist"
    exit 1
fi
output_file=$2
mkdir /tmp/working-directory
cp "$input_file" /tmp/working-directory/input
wc -w /tmp/working-directory/input > /tmp/working-directory/words
if [ $? -ne 0 ]; then
    echo "Command wc -w failed"
    exit 1
fi
rm /tmp/working-directory/input
mv /tmp/working-directory/words "$output_file"
rmdir /tmp/working-directory
```

The Common Workflow Language (CWL)

What does CWL offer?

The Common Workflow Language (CWL)

What does CWL offer?

- Separate housekeeping code from actual processing

The Common Workflow Language (CWL)

What does CWL offer?

- Separate housekeeping code from actual processing
- Isolate inputs, outputs and steps

The Common Workflow Language (CWL)

What does CWL offer?

- Separate housekeeping code from actual processing
- Isolate inputs, outputs and steps
- Report error on failed steps

The Common Workflow Language (CWL)

What does CWL offer?

- Separate housekeeping code from actual processing
- Isolate inputs, outputs and steps
- Report error on failed steps
- Strong static typing

The Common Workflow Language (CWL)

What does CWL offer?

- Separate housekeeping code from actual processing
- Isolate inputs, outputs and steps
- Report error on failed steps
- Strong static typing
- Automatically handle running in different software and hardware environments (containers, clusters, etc.)

The Common Workflow Language (CWL)

What does CWL offer?

- Separate housekeeping code from actual processing
- Isolate inputs, outputs and steps
- Report error on failed steps
- Strong static typing
- Automatically handle running in different software and hardware environments (containers, clusters, etc.)
- Machine inspectable language

CWL vs ccwl

The Common Workflow Language and the Concise Common Workflow Language

CWL vs ccwl

The Common Workflow Language and the Concise Common Workflow Language

Common Workflow Language

- A CWL YAML specification is too verbose.

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.

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

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

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

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
- An embedded domain-specific language in Guile Scheme

CWL vs ccwl

The Common Workflow Language and the Concise Common Workflow Language

Common Workflow Language

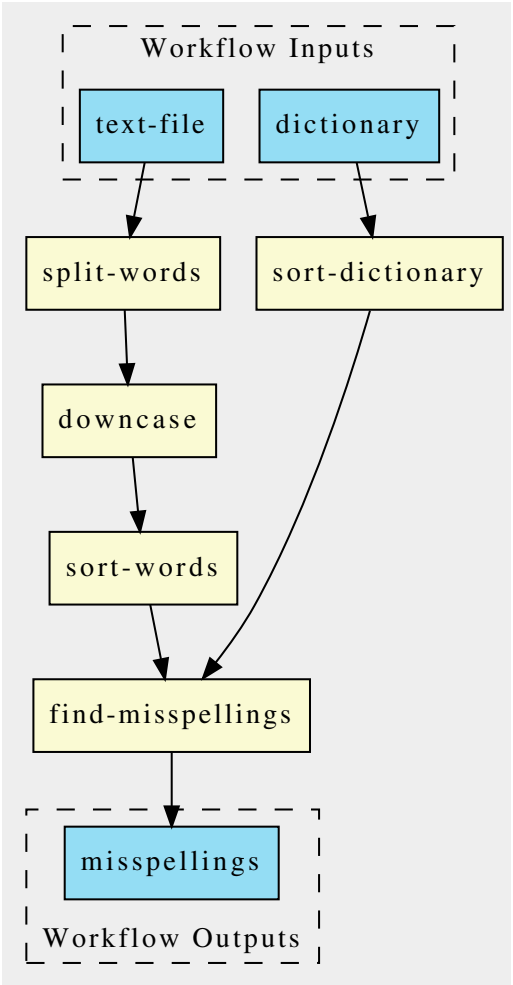
- A CWL YAML specification is too verbose.
- Too many files! Each step has to have its own CWL file. And these need to be wired up together into a workflow CWL file.

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
- An embedded domain-specific language in Guile Scheme
- Only 1800 lines (including comments and blank lines) so far!

Demo

A spell check workflow[1]



[1] Thanks to dgsh (`https://github.com/dspinellis/dgsh`) for inspiring this example

Thank You!

- Website: `https://ccwl.systemreboot.net/`
- Git repo: `https://github.com/arunisaac/ccwl`
- E-mail: arunisaac@systemreboot.net