YAML Diagrams

Subjects: Logic in Computer Science (cs.LO); Mathematical Software (cs.MS)

Author: Martin Coll, Buenos Aires

Abstract

This project gives an interoperable string diagram [3] specification based on the YAML language [5]. Using monoidal semantics it enriches this popular data language with programming capabilities. We publish the yaml-diagrams [b] package that uses PyYAML [6] and DisCoPy [1] to generate, draw and compute string diagrams.

Motivation

YAML is a popular data language with uses in application configuration, network definition, security policies, and more. It makes it possible to write friendly yet complex data structures made of lists, dictionaries and scalars.

All mainstream programming languages have native support and developers write primitives that turn YAML into domain-specific languages. These programs can display a wide range of behavior without any changes to code. There is however a constant tension between simple rules and complex behavior.

Implementation

We build this tool with the intention of creating a general-purpose language. We blur the distinction between programs and data in the spirit of functional programming languages and use category theory to support complex compositional patterns.

In its 1.2.2 specification [5], a YAML document has scalars and two composition types: lists and mappings. We make the obvious choice of modeling sequential composition '`;`' with lists and parallel composition '`@`' with mappings.

The following examples show the mathematical, DisCoPy, YAML and diagrammatic representations of two simple boxes. In that order we show two examples.

Example 1

```
box = input ; ( output @ output )
```

```
Box("box", Ty("input"), Ty("output") @ Ty("output"))
```

```
!box [ input, { output: , output: } ]
```

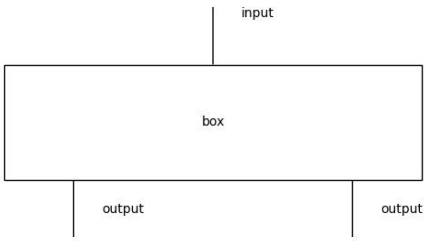

Figure 1: one input and two outputs

Example 2

```
box = (input @ input) ; output
```

```
Box("box", Ty("input") @ Ty("input"), Ty("output"))
```

```
!box [ { input: , input: }, output ]
```

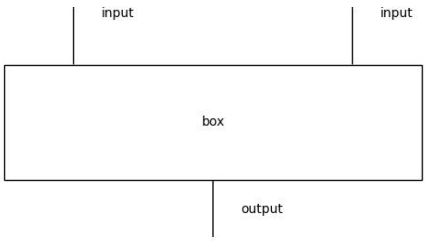

Figure 2: two inputs and one output

We wanted to support all YAML documents as diagrams but monoidal categories alone would not always represent valid equations. We add a Frobenius structure [3] that models any YAML text file into a look-alike diagram. This means users can immediately draw and start tinkering with their existing documents.

Example 3

```
- !box1
  input:
  input:
- !box2
  input: output
```

The diagram above has automatically connected the wires between boxes even when the arities didn't match. Frobenius structure simplifies bureaucracy and achieves our goals.

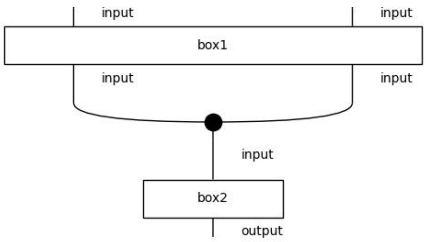

Figure 3: sequential composition

Future work

We want to explore an integration with the filesystem to help building diagrams. We currently handle one file at a time and need to write Python code for composing larger diagrams. This addition can provide a solid foundation to write the compositional parts of a program and restrict traditional programming languages to specific tasks.

The AlgebraicJulia [a] project has many tools for Applied Category Theory including string diagrams. We would like to explore how to make this available in the Julia ecosystem as well.

References

- [1] DisCoPy: Monoidal Categories in Python
- [2] An Introduction to String Diagrams for Coputer Scientists
- [3] String Diagram Rewrite Theory I: Rewriting wih Frobenius Structure
- [4] A survey of graphical languages for monoidal caegories
- [5] YAML specification v1.2.2
- [6] PyYAML processing framework for Python