Nested wiring diagrams in Semagrams

Owen Lynch — Topos Institute
Spencer Breiner — NIST

Semagrams is a scalable platform for developing graphical editors of non-linear syntax. By non-linear syntax, we mean syntax which does not fit easily into an abstract syntax tree, and has a natural two-dimensional structure. Such syntax includes, but is not limited to, Petri nets, directed and undirected wiring diagrams, string diagrams, and graphs. Semagrams is built to edit such syntax in a graphical way, with the aim of not just *drawing* the syntax but actually using it via functorial semantics implemented in an attached backend (for example, Catlab). Thus, the name, which comes from **sema**ntic dia**grams**.

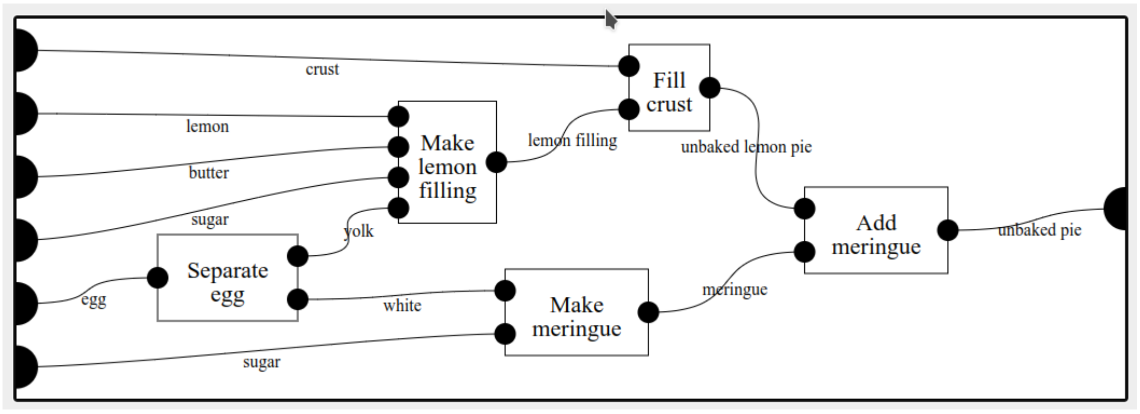

Figure 1. The classic pie recipe wiring diagram as drawn in Semagrams

The goal of Semagrams is to serve two classes of users. The first class of users is developers who would like to make graphical "REPLs" (read-evaluate-print-loops) for their software. Semagrams provides a library of functionality which facilitates much of the hard, repetitive work in such user interfaces, like layout, user interaction, serialization, backend communication, state management, undo/redo, etc. The second class of users are the people who use the user interfaces created by the first class of users. Both classes of users include the authors.

A prototype of Semagrams was demoed at ACT2021; over the last two years there has been a lot of work on Semagrams, including a full rewrite from TypeScript to Scala.js and the subsequent development of more mature and decoupled abstractions for handling a variety of use cases.

In particular, one feature that we have developed is the ability to work with *nested* wiring diagrams, i.e. wiring diagrams where each box contains an entire sub-wiring diagram. Development of this feature required a rethinking of fundamental data structures within Semagrams in order to support recursive editing.

We plan to demonstrate these new capabilities of Semagrams, and additionally to discuss using code samples some of the more general features on which these capabilities rest.