Functor generators for Cartesian closed categories

Stanislav Kapulkin, Viete
stask@viete.io

Abstract

Cartesian closed categories are suitable for describing evaluation logic. A category theory editor with appropriate visual syntax for CCC can be practically used as a programming language. This work demonstrates that expanding the editor's syntax with functors in the form of generators for objects and morphisms enhances programming capabilities through automation and decomposition, making programming more efficient and appealing. The paper presents basic syntax for functor generators. Further development and improvement of the syntax is the subject of future work.

Introduction

Monoidal categories represented as string diagrams have become prevalent in evaluation frameworks, such as DisCoPy. These frameworks provide a syntax for describing evaluations as a series of states and (quite tricky) transitions between them. Subdiagrams can also be enclosed within larger diagrams, akin to extracting code into functions in traditional programming. Subdiagrams are visually displayed within a box that represents the external diagram's morphism.

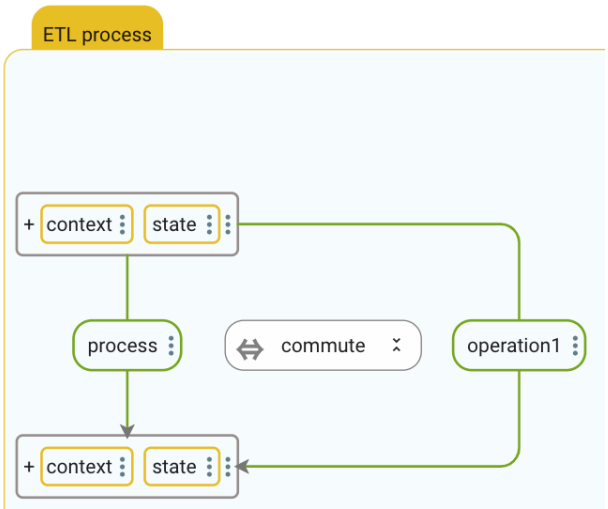
In this paper, we introduce a syntax for functor generators that enables the creation of transformed versions of existing diagrams. We developed a special syntax for subdiagrams compatible with functor generators. A subdiagram is displayed to the right from the external diagram's morphism. That establishes a common directionality that is utilized when placing content generated by functors.

The proposed syntax and implementation details of functor generators are discussed in detail below.

Editor's syntax

Decomposition

Every diagram starts with a single morphism, which represents the evaluation task. This is

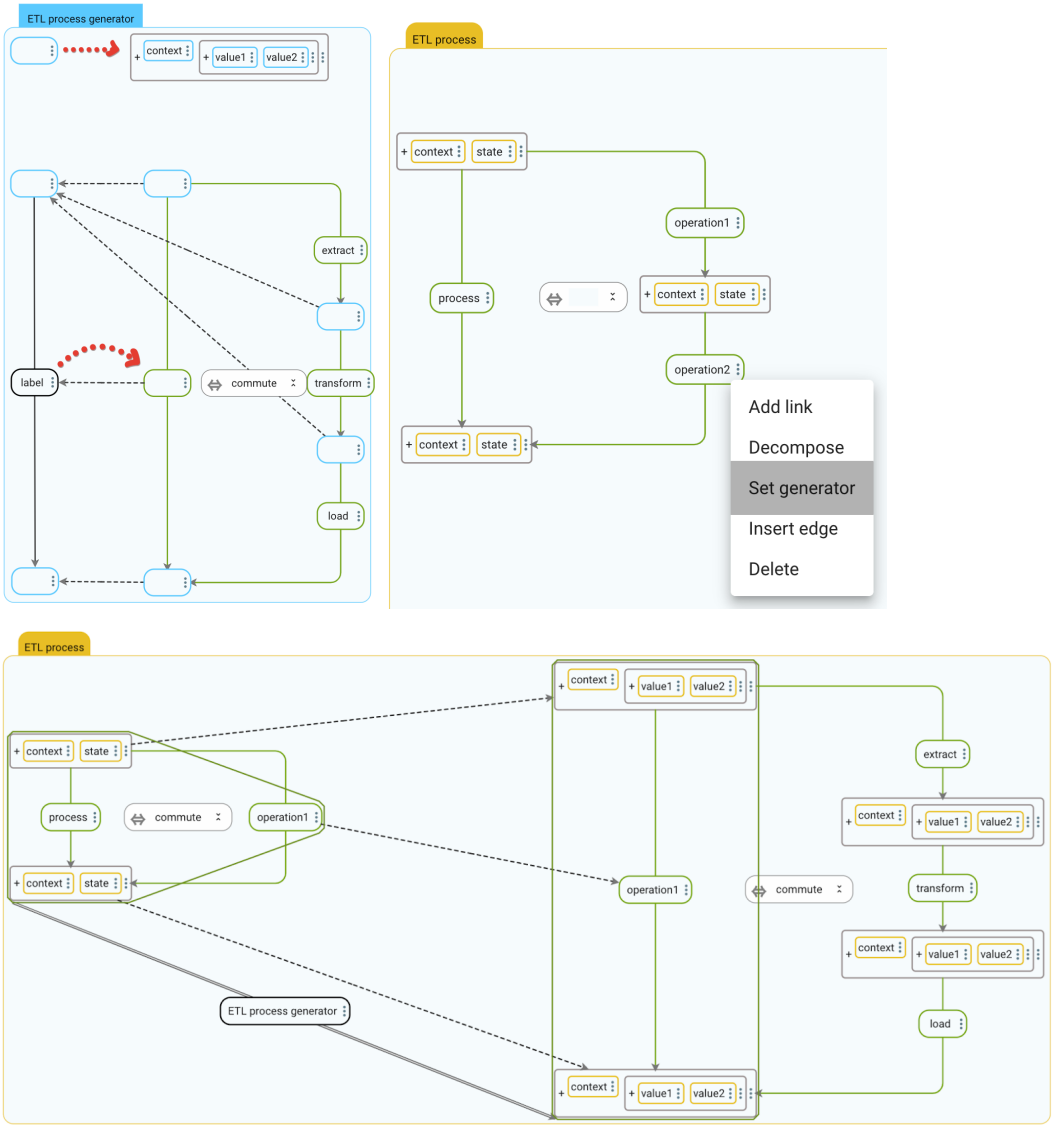


then further decomposed into a chain of morphisms, representing the editing process. The editor's syntax supports both morphism products and coproducts, although these concepts are beyond the scope of this article. In the following example, we consider an ETL (extract-transform-load) process that we wish to describe in detail. Specifically, we aim to represent the morphism "operation1" as a composition of three morphisms: "extract", "transform", and "load". To automate this process, functors are a convenient tool.

Functor generator

The use of a functor generator facilitates the decomposition of ETL operations into a three-step evaluation process. An object generator generates new objects from the initial diagram's ones, while another generator produces morphisms.

When the functor generator is applied to a morphism “operation1”, a detailed diagram is produced on the right-hand side.



Our implementation of functor generators is supported by interactive features. Expanding the initial diagram to include an additional morphism“operation2” leads to the generation of a second three-step evaluation diagram.