Iterated Processes with Finite Interaction Type
Extended Abstract

Chad Nester

Tallinn University of Technology

Abstract

We propose an interpretation of certain cells of the free cornering of the category of partial recursive functions as iterated processes with finite interaction type. Put another way, we propose a model of composable interactive computation formally grounded in the free cornering of a monoidal category.

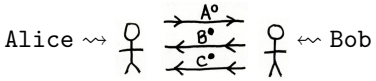
1 The Free Cornering

To begin, we briefly recapitulate the construction and interpretation of the free cornering of a symmetric monoidal category. The free cornering was introduced in [7] as a way to think about interacting processes. It plays an important role in the generalised accounting formalism of [6], and seems to be of foundational importance in the study of bidirectional transformations [1].

Our development is framed by the resource-theoretic understanding of symmetric monoidal categories [2]. Objects are understood of as collections of resources. The tensor product $A \otimes B$ of two objects is the collection consisting of A and B, and the unit I is the empty collection. Morphisms are understood as *transformations*, with $f : A \to B$ giving a way to tranform the resoures of A to the resources of B. We adopt the resource-theoretic perspective here, using the associated vocabulary to elucidate our formal development.

Definition 1 ([7])**.** Let $\mathbb{A}$ be a symmetric monoidal category. Define the monoid $\mathbb{A}^{\circ\bullet}$ of $\mathbb{A}$-*valued exchanges* to be the free monoid on the set of polarized objects of $\mathbb{A}$, as in $\mathbb{A}^{\circ\bullet} = \mathbb{A}_0 \times \{\circ, \bullet\}^*$. We write elements of $\mathbb{A}^{\circ\bullet}$ as in $A^\circ \otimes B^\bullet \otimes C^\bullet$ where A, B, C are objects of $\mathbb{A}$. The unit of $\mathbb{A}^{\circ\bullet}$ (the empty sequence) is denoted I and the monoid operation is sequence concatenation.

Each $\mathbb{A}$-valued exchange $X_1 \otimes \cdots \otimes X_n \in \mathbb{A}^{\circ\bullet}$ involves a left participant and a right participant giving each other resources in sequence, with A° indicating that the left participant should give the right participant an instance of A, and $A^\bullet$ indicating the opposite. For example say the left participant is `Alice` and the right participant is `Bob`. Then we can picture the exchange $A^\circ \otimes B^\bullet \otimes C^\bullet$ as:

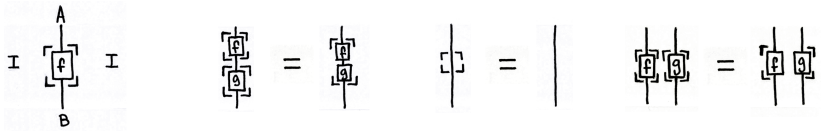


Think of these exchanges as happening *in order*. For example the exchange pictured above demands that first `Alice` gives `Bob` an instance of A, then `Bob` gives `Alice` an instance of B, and then finally `Bob` gives `Alice` an instance of C.

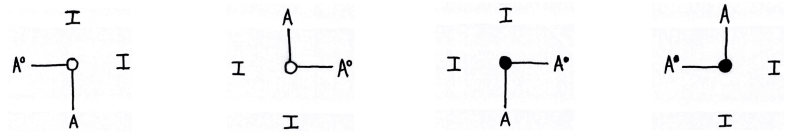
Definition 2 ([7])**.** Let $\mathbb{A}$ be a monoidal category. We define the *free cornering* of $\mathbb{A}$, written $\ulcorner\mathbb{A}\urcorner$, to be the free single-object double category on the following data:

- The horizontal edge monoid $(\mathbb{A}_0, \otimes, I)$ is the object monoid of $\mathbb{A}$
- The vertical edge monoid is $\mathbb{A}^{\circ\bullet}$

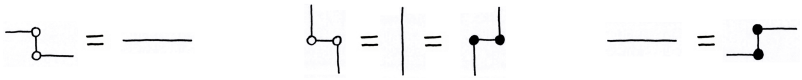
- The generating cells consist of cells $\ulcorner f \urcorner$ for each morphism $f : A \to B$ of $\mathbb{A}$ subject to equations as in:



along with the following *corner cells* for each object A of $\mathbb{A}$:

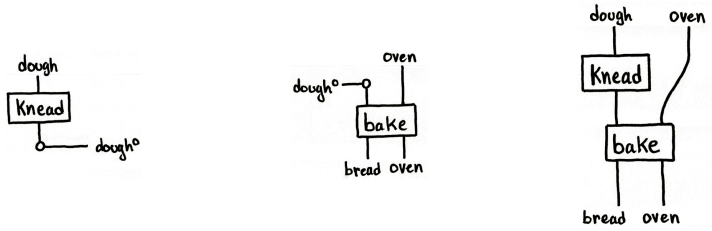


which are subject to the *yanking equations*:



For a precise development of free double categories see [3]. Briefly, cells are formed from the generating cells by horizontal and vertical composition, subject to the axioms of a double category in addition to any generating equations. The corner structure has been heavily studied under various names including *proarrow equipment*, *framed bicategory*, *connection structure*, and *companion and conjoint structure*. A good resource is the appendix of [8].

Cells of $\ulcorner \mathbb{A} \urcorner$ can be understood as *interacting* morphisms of $\mathbb{A}$. Each cell is a method of obtaining the bottom boundary from the top boundary by participating in $\mathbb{A}$-valued exchanges along the left and right boundaries in addition to using the arrows of $\mathbb{A}$. For example, if the morphisms of $\mathbb{A}$ describe processes involved in baking bread, we might have the following cells of $\ulcorner \mathbb{A} \urcorner$:



The cell on the left describes a procedure for transforming `dough` into nothing by `kneading` it and sending the result away along the right boundary, and the cell in the middle describes a procedure for transforming an `oven` into `bread` and an `oven` by receiving `dough` along the left boundary and then using the `oven` to `bake` it. Composing these cells horizontally results in the cell on the right via the yanking equations. In this way the free cornering models concurrent interaction, with the corner cells capturing the flow of information across different components.

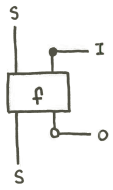
2 Iterated Processes with Finite Interaction Type

Let $\mathcal{K}_1$ be the category of partial recursive functions. Specifically, objects are powers of $\mathbb{N}$ and arrows $\mathbb{N}^n \to \mathbb{N}^m$ are partial recursive functions f such that if $f(x_1, \ldots, x_n)\downarrow$ then $f(x_1, \ldots, x_n) = (y_1, \ldots, y_m)$. Composition and identities are as with functions.

Cells of $\ulcorner\mathcal{K}_1\urcorner$ may be thought of as *processes with finite interaction type*. Operationally, we ask that each such process be equipped with two input streams and two output streams, one of each sort associated with both the left and right boundary of the corresponding cell. Now evaluation proceeds by computation, with the values received and sent out along the boundaries coming from and being sent out along the appropriate input or output stream. To compose cells of $\ulcorner\mathcal{K}_1\urcorner$ is to compose processes. When processes are composed horizontally, the input stream associated with each component is generated by the output stream associated with the other component, which is compatible with the yanking equations.

The finitary nature of these processes is a significant restriction. This can be overcome by considering only those cells of $\ulcorner\mathcal{K}_1\urcorner$ with equal top and bottom boundary. Operationally, we treat these cells as *iterated* processes, which repeat the action specified by the cell forever. The top/bottom boundary allows the state of the system to persist across multiple iterations of a process. Composition of cells of $\ulcorner\mathcal{K}_1\urcorner$ is now composition of (iterated) processes.

Example 1. A *persistent Turing machine* is a Turing machine with three tapes: a read-only input tape, a write-only output tape, and a read-write tape for keeping track of internal state [4]. To operate a persistent Turing machine we require input and output streams corresponding to the input and output tapes. At each stage of computation, the head of the input stream is written to the input tape, the machine performs some computation, and then the contents of the output tape become the next value of the output stream. This is then repeated indefinitely, giving a model of interactive computation. Persistent Turing machines correspond to the cells of $\ulcorner\mathcal{K}_1\urcorner$ (understood as iterated processes) with the following form:



where the wires labelled I, O, and S correspond to the input tape, output tape, and internal state tape, respectively.

We propose cells of $\ulcorner\mathcal{K}_1\urcorner$ with equal top and bottom boundary, understood as iterated processes with finite interaction type, as a model of composable interactive computation. A great deal of further research is necessary to validate this proposal. In particular it remains to implement $\ulcorner\mathcal{K}_1\urcorner$ with this interpretation, although we imagine there are no significant obstacles to this. A fully formal version of the proposed operational semantics is also desirable, as it would allow us to consider the question of whether or not the equations of the free cornering are enough to characterise operational equivalence of processes. We note that there is a satisfying notion of *process motion* in the sense of [5], in which a process α with internal state of type A may move to a process β with internal state of type B through an intermediary arrow $f : A \to B$. Operationally, we apply f to the state of α at the end of an iteration, and continue as β. Another important question is how we might formally specify patterns of motion that an iterated process may exhibit.

While this research is at an early stage, we find the interpretation of $\ulcorner\mathcal{K}_1\urcorner$ as iterated processes with finite interaction type compelling, and would welcome the opportunity to discuss these ideas with the wider research community.

References

[1] G. Boisseau, C. Nester, and M. Roman. Cornering optics. In *Applied Category Theory (ACT2022), to appear*.

[2] B. Coecke, T. Fritz, and R.W. Spekkens. A mathematical theory of resources. *Information and Computation*, 250:59–86, 2016.

[3] M. Fiore, S. Paoli, and D. Pronk. Model structures on the category of small double categories. *Algebraic and Geometric Topology*, 8(4):1855–1959, 2008.

[4] D.Q. Goldin. Persistent turing machines as a model of interactive computation. In *Foundations of Information and Knowledge Systems*, pages 116–135. Springer Berlin Heidelberg, 2000.

[5] R. Milner. *The Space and Motion of Communicating Agents*. Cambridge University Press, 2009.

[6] C. Nester. Situated transition sytems. In *Applied Category Theory (ACT 2021), to appear*, 2021.

[7] C. Nester. The structure of concurrent process histories. In *International Conference on Coordination Models and Languages*, pages 209–224, 2021.

[8] M. Shulman. Framed bicategories and monoidal fibrations. *Theory and Applications of Categories*, 20(18):650–738, 2008.