Memoryless systems generate the class of all discrete systems (Extended abstract)

Erwan Beurier Dominique Pastor David I. Spivak *

July 1, 2019

Abstract

Discrete systems are automata that receive streams of inputs. They update their internal state and then produce outputs. These constitute the bricks used in engineering and science in general for building complex machines. Discrete systems can be thoughts as simple as needed, and then wired together to produce arbitrarily complex systems. This representation is also thought to be universal, in the sense that a full range of objects may be described as constructions made from more modest elements, from biological systems to computers or any industrial machine. This paper aims at proving that the most complex discrete systems can reduce to a set of simple memoryless automata, that is, automata that do not store any history of their inputs and outputs. Such simplicity makes sense from a biological point of view as well, where complex objects like the brain can be reduced to an interaction of mere cells. From connections between atomic elements emerges the complexity of systems.

Keywords: category theory, discrete systems, memoryless system, monoidal category.

Preprint version: complete version can be found online on the French open-archive HAL. Submitted to Hindawi's International Journal of Mathematics and Mathematical Science.

1 Introduction

This paper uses a representation of dynamical systems made with generalised automata, whose state space is unrestricted (not necessarily finite nor countable). These automata act like stream processors, in the following sense: they take inputs as a stream of data in discrete time, and produces outputs, one per input. We refer to them as discrete systems. In discrete systems, the state space may be seen as a memory of the inputs, a remainder of its history. Each input of the stream changes the current state of the system and has an influence on its future outputs.

2 Discrete systems and their equivalences

This article follows directly from [Spi16]. We recall that $\mathbf{TFS}_{\mathbf{Sets}}$ is the category of $\mathbf{Sets}$-typed finite sets (i.e. pairs (P, τ) where P is finite and $\tau : P \to \mathbf{Sets}$ is a function) and $\mathscr{W}$

*Spivak was supported by AFOSR grants FA9550-14-1-0031 and FA9550-17-1-0058, as well as NASA grant NNH13ZEA001N-SSAT while working on this project.

is the category of **Sets**-boxes (or simply "boxes", i.e. pairs $X = \left(X^{\text{in}}, X^{\text{out}}\right)$ where X^{in} and X^{out} are **Sets**-typed finite sets). Both categories are symmetric monoidal with respect to operations that we will not introduce.

A **Sets**-typed finite set (P, τ) can be seen as a list of sets $\langle \tau(p_1), \ldots, \tau(p_n) \rangle$. An arrow between two list $\gamma : (P, \tau) \to (P', \tau')$ is a function $\gamma : P \to P'$ such that $\tau = \tau' \circ \gamma$, and it corresponds to a reordering / duplication of the elements of the starting list. Then, **Sets**-boxes are pairs of **Sets**-typed finite sets $\left(X^{\text{in}}, X^{\text{out}}\right)$ (the functions $x^{\text{in}} : X^{\text{in}} \to \mathbf{Sets}$ and $x^{\text{out}} : X^{\text{out}} \to \mathbf{Sets}$ are implicit): $\left(X^{\text{in}}, x^{\text{in}}\right)$ is the list of accepted types of the input ports, and $\left(X^{\text{out}}, x^{\text{out}}\right)$ is the list of types of the outputs ports.

If (P, τ) is a **Sets**-typed finite set, we define its dependent product as the product of sets $\prod_{p \in P} \tau(p)$, denoted by $\widehat{P}$. If $X = \left(X^{\text{in}}, X^{\text{out}}\right)$ is a box, we define its dependent product to be the pair $\widehat{X} = \left(\widehat{X^{\text{in}}}, \widehat{X^{\text{out}}}\right)$. The idea is that, if a box X has a list of input types X^{in}, then the dependent product $\widehat{X^{\text{in}}}$ is the mathematical object that contains all its inputs. Similarly, $\widehat{X^{\text{out}}}$ is the set of all the potential outputs.

Thus, a box, or rather, the dependent product of a box, can be seen as the signature of the function, or discrete system, that it will contain.

Definition 2.1 (Discrete systems [Spi16])**.** Let $X = \left(X^{\text{in}}, X^{\text{out}}\right) \in \mathscr{W}$ be a box.

A *discrete system for the box* X, or simply *discrete system*, consists of a 4-tuple $F = \left(S_F, f^{\text{rdt}}, f^{\text{upd}}, s_{F,0}\right)$ where:

- $S_F \in \mathbf{Sets}$ is called the *state set*
- $f^{\text{rdt}} : S_F \to \widehat{X^{\text{out}}}$ is called the *readout function*
- $f^{\text{upd}} : \widehat{X^{\text{in}}} \times S_F \to S_F$ is called the *update function*
- $s_0 \in S_F$ is called the *initial state*

A box X can be seen as an empty frame where discrete systems fit. For a given box X, the set of all discrete systems that live inside is denoted $\mathrm{DS}(X)$.

Such discrete systems do not recognize languages like standard automata, but are rather automata that take and return streams of data.

Theorem 2.2 ([Spi16])**.** *$DS : \mathscr{W} \to \mathbf{Sets}$ is a lax monoidal functor.*

3 Main results

Definition 3.1 (Closure)**.** Let $A : \mathscr{W} \to \mathbf{Sets}$ be a lax monoidal functor. If $B : \mathrm{Ob}_{\mathscr{W}} \to \mathrm{Ob}_{\mathbf{Sets}}$ is a map (not necessarily a functor) such that $\forall X \in \mathrm{Ob}_{\mathscr{W}}$, $B(X) \subseteq A(X)$, then the closure of B in A, written $\mathrm{Clos}(B)$, is the smallest lax monoidal subfunctor of A containing B.

We now define a sub-map of DS that build memoryless (or reactive) systems. Our main result involves showing that the closure of this sub-map is DS. In substance, we show that any discrete system can be reduced to a memoryless system.

Definition 3.2 (Memoryless systems)**.** In a box $X = \left(X^{\text{in}}, X^{\text{out}}\right)$, a *memoryless (or reactive) system* is a discrete system $\left(S_F, f^{\text{rdt}}, f^{\text{upd}}, s_{F,0}\right) \in \mathrm{DS}(X)$ such that $f^{\text{upd}} : \widehat{X^{\text{in}}} \times S_F \to S_F$ verifies $f^{\text{upd}} = f^u \circ \pi_{\widehat{X^{\text{in}}}}$ for some $f^u : \widehat{X^{\text{in}}} \to S_F$.

For a box X, the set of all memoryless systems that live inside is written $\mathrm{DS}^{\mathrm{ML}}(X)$.

Those systems are *memoryless* in that the state space is seen as a kind of memory, the "life experience" of the system. A memoryless system is a discrete system that ignores its memory when it transitions to a new state. In a sense, it only *reacts* to its input regardless of its past experience. An example could be a trained neural network, which, after learning its parameters, will only accomplish its function without updating them.

We can now state the main result of this paper: every discrete system is equivalent to a memoryless system and a feedback loop. Here, we consider that two systems are equivalent when, given the same input stream, they return the same output stream. The feedback loop does the trick of reminding the previous output, and thus, somehow emulate a state space.

Theorem 3.3. $\mathrm{Clos}\left(\mathrm{DS}^{\mathrm{ML}}\right) \cong \mathrm{DS}$.

Proof. The proof is constructive. We build the equivalent memoryless discrete system by expanding its state space, so that the current state consists in the previous output (feedback loop). □

This result and its proof have two interesting interpretations. First, memory emerges from connections, and second, the only information one needs to build a state space, and thus memory, is the last output; in particular, we do not need to have the whole list of previous outputs.

4 Conclusion

The categorical framework was used in order to prove that a discrete system is equivalent to a memoryless system with the right wiring, so that all the system needs to know is its last output. With their nesting property, discrete systems are complex systems, and this result may be seen as an emergence property.

The proof is indeed constructive, but the equivalent memoryless system we exhibit is probably far from being "optimal" (for example, it is probably not the most parsimonious in terms of state space). Finally, could we extend the result to other types of dynamical systems, for example those introduced in [Spi16], such as measurable dynamical systems, or continuous ones?

References

[Spi16] David I. Spivak. The steady states of coupled dynamical systems compose according to matrix arithmetic. Available online: https://arxiv.org/abs/1512.00802, 2016.