Automata as Functors

Daniela Petrişan
joint work with Thomas Colcombet *
CNRS, IRIF, Université Paris Diderot – Paris 7, France

> *"It is clear from this introduction that this paper contains nothing that is essentially new, except perhaps for a point of view".*
>
> *Eilenberg and Wright*
> *Automata in General Algebras, 1967*

Abstract

We explain in a systematic way various phenomena from automata theory. The new point of view that we put forward in this work is that automata can be interpreted as functors from an input category specifying the type of the machine, to a category specifying the output values. Minimization of word automata or of weighted automata over a field, syntactic algebras, the correctness of Brzozowski's minimization algorithm or of Choffrut's minimization algorithm for subsequential transducers — all arise from the same generic category theoretic principles.

The relationship between automata and category theory has a long history, starting with the seminal work of Eilenberg who advanced the algebraic point of view on automata theory. It is perhaps no coincidence that Eilenberg is at the same time one of the founding fathers of category theory. Typically, automata are interpreted either as algebras (together with a final map) as put forward in, say, [1] or as coalgebras (together with an initial map), see for example [5]. In this talk I would like to present another category-theoretic point of view in which automata are seen as functors

$$\mathcal{A}\colon \mathcal{I} \to \mathcal{C}$$

from a category $\mathcal{I}$ that specifies the type of the automaton to a category $\mathcal{C}$ that specifies the type of the outputs. For example, for word automata over a finite alphabet A, the category $\mathcal{I}$ is the free category generated from the arrows below, where for each $a \in A$ we have an arrow $a\colon \mathsf{states} \to \mathsf{states}$.

$$\begin{tikzcd} \mathsf{in} \arrow[r, "\triangleright"] & \mathsf{states} \arrow[loop above, "a"] \arrow[r, "\triangleleft"] & \mathsf{out} \end{tikzcd}$$

*This work was supported by the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation programme (grant agreement No.670624), and by the DeLTA ANR project (ANR-16-CE40-0007). The authors also thank the Simons Institute for the Theory of Computing where this work has been partly developped.

By varying the output category one can model various forms of machines:

AUTOMATA	OUTPUT CATEGORY
deterministic automata	Set – the category of sets and functions
non-deterministic automata	Rel – the category of sets and relations
weighted automata	Mod_S – the category of modules for a semiring S
subsequential transducers with output alphabet B	the category of free pointed B^*-actions

Various phenomena from automata theory can be explained in a systematic and unifying way. The tool for this is a simple lemma stating that the process of moving back and forth between various output categories (formally expressed in terms of *adjunctions*) can be lifted to the corresponding categories of automata.

For a trivial and illustrative example, consider the categories Set and Rel as above. These two categories can be related by such an adjunction, where in one direction we simply see a function between two sets as a relation between them, while in the other direction we transform a relation between X and Y into a function $\mathcal{P}(X) \to \mathcal{P}(Y)$:

$$\begin{tikzcd} \mathsf{Set} \arrow[r, bend left] \arrow[r, phantom, "\perp"] & \mathsf{Rel} \arrow[l, bend left] \end{tikzcd}$$

Lifting this connection to categories of automata, we see how the powerset construction turning a non-deterministic automaton into a deterministic one is really the adjoint process of seeing any deterministic automaton as a non-deterministic one.

$$\begin{tikzcd} \mathsf{DFA} \arrow[r, bend left] \arrow[r, phantom, "\perp"] & \mathsf{NFA} \arrow[l, bend left] \end{tikzcd}$$

In this work we push this idea further. We explain in this setting language recognition and how the minimal automaton accepting a language can be obtained provided that we have the following ingredients: an initial and a final automaton for that language, as well as a factorisation of the unique map between them.

We give sufficient conditions on the output category so that minimization is possible. And we see a non-trivial example, the subsequential transducers, where these conditions are not met, yet the minimal transducer à la Choffrut [2] can be constructed following the same recipe.

The work reported here has appeared in [4, 3].

References

[1] M. A. Arbib and E. G. Manes. Adjoint machines, state-behavior machines, and duality. *Journal of Pure and Applied Algebra*, 6(3):313 – 344, 1975.

[2] C. Choffrut. A generalization of Ginsburg and Rose's characterization of G-S-M mappings. In *ICALP*, volume 71 of *Lecture Notes in Computer Science*, pages 88–103. Springer, 1979.

[3] T. Colcombet and D. Petrişan. Automata and minimization. *SIGLOG News*, 4(2):4–27, 2017.

[4] T. Colcombet and D. Petrişan. Automata minimization: a functorial approach. *CALCO*, to appear, 2017.

[5] B. Jacobs and J. Rutten. A tutorial on (co)algebras and (co)induction. *EATCS Bulletin*, 62:62–222, 1997.