Function spaces for orbit-finite sets

Mikołaj Bojańczyk
University of Warsaw, Poland

Lê Thành Dũng (Tito) Nguyễn
École normale supérieure de Lyon, France

Rafał Stefański
University of Warsaw, Poland

Abstract

Orbit-finite sets are a generalisation of finite sets, and as such support many operations allowed for finite sets, such as pairing, quotienting, or taking subsets. However, they do not support function spaces, i.e. if X and Y are orbit-finite sets, then the space of finitely supported functions from X to Y is not orbit-finite. We propose a solution to this problem inspired by linear logic.

2012 ACM Subject Classification Theory of computation → Formal languages and automata theory

Keywords and phrases Orbit-finite sets, automata, linear types, game semantics

Related Version *Full version with appendix*: `arXiv:2404.05265`

Acknowledgements L. T. D. Nguyễn would like to thank Clovis Eberhart and Cécilia Pradic for their ongoing collaboration on "implicit automata for data words" (cf. [12, §1.4.4]), which inspired some ideas in this paper. R. Stefański would like to thank Samson Abramsky for introducing him to the beautiful world of game semantics and suggesting to study it in the context of this paper.

Orbit-finite sets. The class of orbit-finite sets contains all finite sets and some infinite sets, but still shares some properties with the class of finite sets. The idea, which dates back to Fraenkel–Mostowski models of set theory, is to begin with an infinite set $\mathbb{A}$ of *atoms* or *urelements*. We think of the atoms as being names, such as Eve or John, and atoms can only be compared with respect to equality. Intuitively speaking, an orbit-finite set is a set that can be constructed using the atoms, such as $\mathbb{A}^2$ or $\mathbb{A}^*$, subject to the constraint that there are finitely many elements up to renaming atoms. For example, $\mathbb{A}^2$ is orbit-finite because it has two elements up to renaming atoms, namely (John, John) and (John, Eve), while $\mathbb{A}^*$ is not orbit-finite, because the length of a sequence is invariant under renaming atoms, and there are infinitely many possible lengths. For a survey on orbit-finite sets, see [3].

The notion of orbit-finiteness can be seen as an attempt to find an appropriate notion of finiteness for the nominal sets of Gabbay and Pitts [14] (which have also been used to handle syntax with binders). This attempt emerged from the study of computational models such as monoids [4] and automata [5] over infinite alphabets. Let us illustrate orbit-finiteness using an automaton example.

▶ **Example 1** (An orbit-finite automaton). Let $L \subseteq \mathbb{A}^*$ be the language of words in which the letter from the first position does not appear again. This language contains John · Mark · Mark · Eve, because John does not reappear, but it does not contain John · Mark · John. To recognize this language, we can use a deterministic automaton, which uses its state to remember the first letter. In this automaton, the input alphabet is $\Sigma = \mathbb{A}$ and the state space is $Q = 1 + 1 + \mathbb{A}$. In this state space, there are two special states, namely the initial state and a rejecting error state, and furthermore there is one state for each atom $a \in \mathbb{A}$, which represents a situation where the first letter was a but it has not been seen again yet. This state space is infinite; but it is orbit-finite, since each of the three components in Q represents a single orbit.◀

Orbit-finite sets have many advantages, which ensure that they are a good setting for automata theory, and discrete mathematics in general. For example, an orbit-finite set can be represented in a finite way [3], which ensures that it becomes meaningful to talk about algorithms that input orbit-finite sets, such as an emptiness check for an automaton. Also, orbit-finite sets are closed under taking disjoint unions and products, which ensures that natural automata constructions, such as the union of two nondeterministic automata or the product of two deterministic automata can be performed.

However, orbit-finite sets do not have all the closure properties of finite sets. Notably missing is the powerset operation, and more generally taking function spaces. For example, the powerset of $\mathbb{A}$ is not orbit-finite, since already the finite subsets give infinitely many orbits (two finite subsets of different sizes will be in different orbits). The lack of powersets means that one cannot do the subset construction from automata theory, and in particular deterministic and nondeterministic automata are not equivalent. This non-equivalence was known from the early days of automata for infinite alphabets [10], and in fact, some decision problems, such as equivalence, are decidable for deterministic automata but undecidable for nondeterministic automata [11]. Another construction that fails is converting a deterministic automaton into a monoid [4, p. 221]; this is because function spaces on orbit-finite sets are no longer orbit-finite, as explained in the following example.

▶ **Example 2** (Failure of the monoid construction)**.** Let us show that the automaton from Example 1 cannot be converted into a monoid. The standard construction would be to define the monoid as the subset $M \subseteq Q \to Q$ of all state transformations, namely the subset generated by individual input letters. Unfortunately, this construction does not work. This is because in order for two input words to give the same state transformation, they need to have the same set of letters that appear in them. In particular, the corresponding set of set transformations is not orbit-finite, for the same reason as why the finite powerset is not orbit-finite. Not only does the standard construction not work, but also this language is not recognized by any orbit-finite monoid.◀

The single-use restriction. An attempt to address this problem was provided in [17, 6], based on *single-use* functions. The idea, which originates in linear types and linear logic, is to restrict the functions so that they use each argument at most once. For example, consider the following two functions that input atoms and output Booleans:

$$a \in \mathbb{A} \mapsto \begin{cases} \text{true} & \text{if } a = \text{John} \\ \text{false} & \text{otherwise} \end{cases} \qquad a \in \mathbb{A} \mapsto \begin{cases} \text{true} & \text{if } a = \text{John or } a = \text{Eve} \\ \text{false} & \text{otherwise} \end{cases}$$

Intuitively, the first function is single-use, since it compares the input atom to John only, while the second function is not single-use, since it requires two comparisons, with John and Eve. Here is another example, which shows that the problems with the monoid construction from Example 2 could be blamed on a violation of the single-use condition.

▶ **Example 3.** Consider the transition function of the automaton in Example 1, which inputs a state in $1 + 1 + \mathbb{A}$ together with an input letter from $\mathbb{A}$, and returns a new state. This function is not single use. Indeed, if the state is in $\mathbb{A}$, then the transition function compares it for equality with the input letter; but if the comparison returns true, another copy of the old state must be kept as the new state for future comparisons. ◀

If one restricts attention to functions that are single-use, much of the usual robustness of automata theory is recovered, with deterministic automata being equivalent to monoids, and both being equivalent to two-way deterministic automata [6].

Despite the success of the single-use restriction in solving automata problems, one would ideally prefer a more principled approach, in which instead of defining single-use automata, we would define a more general object, namely single-use sets and functions. Then the definitions of automata and monoids, as well theorems speaking about them, should arise automatically as a result of suitable closure properties of the sets and functions.

This approach was pursued in [17], in which a *category* of orbit-finite sets with single-use functions was proposed. In this corresponding category, one can represent the set of all single-use functions between two orbit-finite sets X and Y as a new set, call it $X \Rightarrow Y$, which is also orbit-finite. However, as we will see later in this paper, this proposal is not entirely satisfactory, since it fails to account for standard operations that one would like to perform on function spaces, most importantly partial application (currying). In the language of category theory, the proposal from [17] failed to be a monoidal closed category.

Contributions. The main contribution of this work is to propose a notion of single-use sets and functions, which extends the proposal from [17], but which is rich enough to be closed under taking function spaces. More formally, we propose a category of single-use functions between *linear types* that denote orbit-finite sets with additional metadata, and we prove that:

▶ **Theorem 4.** *Let V and W be objects (i.e. linear types). There exists an object, denoted by $V \Rightarrow W$, and a morphism (i.e. a single-use function)* $\mathrm{eval} : (V \Rightarrow W) \otimes V \to W$ *with the following property. For every morphism $f : X \otimes V \to W$ there is a (*not necessarily unique*) morphism $h : X \to (V \Rightarrow W)$ such that the following diagram commutes:*

$$\begin{tikzcd}
X \otimes V \arrow[r, "h \otimes \mathrm{id}"] \arrow[dr, "f"'] & (V \Rightarrow W) \otimes V \arrow[d, "\mathrm{eval}"] \\
& W
\end{tikzcd}$$

In other words, our symmetric monoidal category is *weakly closed*. To guarantee uniqueness in the universal property, and thus get an actual symmetric monoidal closed category, one can perform a quotient by partial equivalence relations.

The main idea is to extend the type system from [17] with the additive conjunction '&' of linear logic. Thanks to the distinction between $X \otimes Y$ (in [17] denoted as $X \times Y$) and $X \& Y$, the function space can be built so that the appropriate operations on functions, namely application and currying, can be implemented in a single-use way. Note that this function space is a derived notion, not a type constructor in our syntax; indeed, our linear types are all "first-order", so that they denote orbit-finite sets in a straightforward way:

$$[\![1]\!] = 1 \quad [\![\mathbb{A}]\!] = \mathbb{A} \quad [\![X + Y]\!] = [\![X]\!] + [\![Y]\!] \quad [\![X \otimes Y]\!] = [\![X \& Y]\!] = [\![X]\!] \times [\![Y]\!].$$

Our construction of weak function spaces uses *game semantics* to represent single-use functions as first-order data. More precisely, it is closely based on the "simple games" exposed in e.g. [1, 9]. However, as far as we know, it is an original idea to have an infinite but orbit-finite base type $\mathbb{A}$, and to observe that all constructions in game semantics are consistent with orbit-finiteness. We believe that the resulting category deserves further study, and that it is an interesting and non-trivial example of a category representing "finite" objects.

Related work: categories and λ-calculus. There have been several works using category theory to generalize classical operations on automata, such as the coalgebraic "generalized powerset construction” [16]. The closest to the philosophy that this paper might be the work of Colcombet and Petrişan [7]: it introduces a setting where automata over different categories may be studied and compared (see e.g. [2] for applications). Within this setting, Pradic and the second author have investigated some properties of automata over symmetric monoidal closed categories [12, Sections 1.2.3 and 4.7–4.8].

The latter emerged as part of their research on “implicit automata” [13, 12, 15], which is about relating the expressive power of automata and typed λ-calculi. In [12, Chapter 4], a monoidal closed category of single-use assignments on string-valued registers is built and used to relate a register-based string transducer model to a λ-calculus with linear types. Indeed, symmetric monoidal closed categories are famous for providing denotational semantics for the linear λ-calculus. Similarly, our results here could serve to characterize the languages of words with atoms studied by the first and third author in [6] via some typed λ-calculus.

Conversely, our Theorem 4 might also be provable by representing single-use functions as λ-terms (or programs in some richly structured syntactic formalism) instead of strategies over games. Indeed, it is a classical fact that a simple type is inhabited by finitely many linear λ-terms up to β-conversion (when there are no primitive constants), and variations on this fact have been used in the literature to relate automata and λ-calculus [13, 8].

References

1 Samson Abramsky. Semantics of interaction. In Peter Dybjer and Andrew Pitts, editors, *Proceedings of the 1996 CLiCS Summer School, Isaac Newton Institute*. Cambridge University Press, 1997. arXiv:1312.0121.

2 Quentin Aristote. Active learning of deterministic transducers with outputs in arbitrary monoids. In Aniello Murano and Alexandra Silva, editors, *32nd EACSL Annual Conference on Computer Science Logic, CSL 2024, February 19-23, 2024, Naples, Italy*, volume 288 of *LIPIcs*, pages 11:1–11:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.CSL.2024.11.

3 Mikołaj Bojańczyk. Slightly infinite sets. URL: https://www.mimuw.edu.pl/~bojan/paper/atom-book.

4 Mikołaj Bojańczyk. Nominal Monoids. *Theory of Computing Systems*, 53(2):194–222, 2013. doi:10.1007/S00224-013-9464-1.

5 Mikołaj Bojańczyk, Bartek Klin, and Slawomir Lasota. Automata theory in nominal sets. *Logical Methods in Computer Science*, 10(3), 2014. doi:10.2168/LMCS-10(3:4)2014.

6 Mikołaj Bojańczyk and Rafał Stefański. Single-Use Automata and Transducers for Infinite Alphabets. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *International Colloquium on Automata, Languages, and Programming (ICALP 2020)*, volume 168 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 113:1–113:14, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2020.113.

7 Thomas Colcombet and Daniela Petrişan. Automata Minimization: a Functorial Approach. *Logical Methods in Computer Science*, 16(1), March 2020. doi:10.23638/LMCS-16(1:32)2020.

8 Paul Gallot, Aurélien Lemay, and Sylvain Salvati. Linear high-order deterministic tree transducers with regular look-ahead. In Javier Esparza and Daniel Král’, editors, *45th International Symposium on Mathematical Foundations of Computer Science, MFCS 2020, August 24-28, 2020, Prague, Czech Republic*, volume 170 of *LIPIcs*, pages 38:1–38:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.MFCS.2020.38.

9 Martin Hyland. Game semantics. In Andrew M. Pitts and P. Dybjer, editors, *Semantics and Logics of Computation*, Publications of the Newton Institute, page 131–184. Cambridge University Press, 1997. doi:10.1017/CBO9780511526619.005.

10 Michael Kaminski and Nissim Francez. Finite-Memory Automata. *Theoretical Computer Science*, 134(2):329–363, 1994. doi:10.1016/0304-3975(94)90242-9.

11 Frank Neven, Thomas Schwentick, and Victor Vianu. Finite state machines for strings over infinite alphabets. *ACM Transactions on Computational Logic*, 5(3):403–435, 2004. doi:10.1145/1013560.1013562.

12 Lê Thành Dũng Nguyễn. *Implicit automata in linear logic and categorical transducer theory.* PhD thesis, Université Paris XIII (Sorbonne Paris Nord), December 2021. URL: https://theses.hal.science/tel-04132636.

13 Lê Thành Dũng Nguyễn and Cécilia Pradic. Implicit automata in typed λ-calculi I: aperiodicity in a non-commutative logic. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 168 of *LIPIcs*, pages 135:1–135:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ICALP.2020.135.

14 Andrew M. Pitts. *Nominal Sets: Names and Symmetry in Computer Science*, volume 57 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2013. doi:10.1017/CBO9781139084673.

15 Cécilia Pradic and Ian Price. Implicit automata in λ-calculi iii: affine planar string-to-string functions, 2024. arXiv:2404.03985.

16 Alexandra Silva, Filippo Bonchi, Marcello M. Bonsangue, and Jan J. M. M. Rutten. Generalizing determinization from automata to coalgebras. *Logical Methods in Computer Science*, 9(1), 2013. doi:10.2168/LMCS-9(1:9)2013.

17 Rafał Stefański. *The single-use restriction for register automata and transducers over infinite alphabets.* PhD thesis, University of Warsaw, 2023.