Regular Monoidal Languages*

Matthew Earnshaw and Paweł Sobociński

Tallinn University of Technology, Estonia

Abstract

We introduce regular languages of morphisms in free monoidal categories. These subsume the classical theory of regular languages of words and trees, but also open up a much wider class of string diagram languages. We use monoidal and cartesian restriction categories to investigate the properties of regular monoidal languages, and provide a sufficient condition for recognition by deterministic monoidal automata.

1 Introduction

Formal language theory extends beyond words to infinite words, rational sequences, trees, countable linear orders, graphs of bounded tree width, etc. Recently, the unity of the field has been better understood by seeing such structures as arising from monads on the category of sets [1].

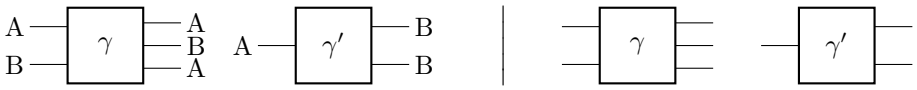
Here we make a step in extending language theory to higher-dimensional algebraic structures, moving from subsets of free monoids (classical languages) to subsets of morphisms in strict monoidal categories. By investigating morphisms in free monoidal categories from the perspective of language theory, this work contributes to research into the computational manipulation of string diagrams, and so their usage in industrial strength applications [2, 6, 9].

Section 2 introduces the definitions of monoidal language and regular monoidal grammars, with examples. Section 3 introduces monoidal automata, and a determinizable subclass. Section 4 introduces an invariant of languages that provides a sufficient condition for determinizability.

2 Regular monoidal grammars and their languages

Classically, languages over an alphabet Σ are subsets of the free monoid Σ^*. Monoidal languages are defined similarly, replacing free monoids with free strict monoidal categories (free *pros*).

A *monoidal graph* is given by edges (represented by boxes) bounded by source/target *words* of vertices (e.g. left below). When the set of vertices is a singleton, these words are natural number *arities*/*coarities* – we call such monoidal graphs *monoidal alphabets* (e.g. right below):



The components of a monoidal graph are *generators* for free pros, with objects words of vertices, and morphisms *string diagrams* built from the edges of the monoidal graph by serial and parallel composition (see [4, Appendix A]).

Definition 2.1. *A monoidal language L over a monoidal alphabet Γ is a subset $L \subseteq \mathscr{F}\Gamma(0,0)$ of morphisms with arity and coarity 0 in the free pro generated by Γ.*

*This is an extended abstract of the paper [4]. This research was supported by the ESF funded Estonian IT Academy research measure (project 2014-2020.4.05.19- 0001). The second author was additionally supported by the Estonian Research Council grant PRG1210.

Regular monoidal grammars are a finite specification of monoidal languages analogous to classical regular languages, generalizing Walters' definition of regular grammar [8]:

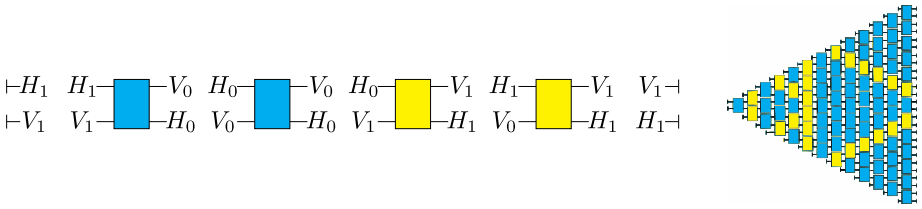
Definition 2.2. *A regular monoidal grammar is a morphism of finite monoidal graphs* $\Psi : M \to \Gamma$ *where* Γ *is a monoidal alphabet. Every regular monoidal grammar determines a pro morphism between free pros,* $\mathscr{F}\Psi$*, which we also refer to as a regular monoidal grammar.*

Intuitively, a regular monoidal grammar is a labelling of the edges of the monoidal graph M by edges in Γ. Given a string diagram $s \in \mathscr{F}\Gamma$, we can think of the set of string diagrams $\mathscr{F}\Psi^{-1}(s)$ the possible "parsings" of that diagram. Regular monoidal grammars have a simple graphical representation illustrated in the examples below: we draw the monoidal graph M but label its boxes by their image under Ψ.

Definition 2.3. *Given a regular monoidal grammar* $\Psi : M \to \Gamma$*, the image under* $\mathscr{F}\Psi$ *of the endo-hom-set of the monoidal unit* ε *in* $\mathscr{F}M$ *is a monoidal language* $\mathscr{F}\Psi[\mathscr{F}M(\varepsilon,\varepsilon)] \subseteq \mathscr{F}\Gamma(0,0)$*. We call the class of monoidal languages determined by regular monoidal grammars the regular monoidal languages.*

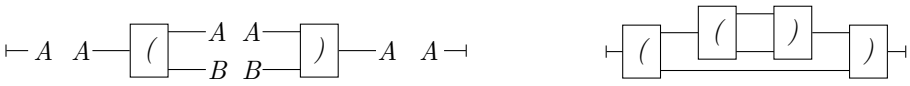
Diagrammatically, a regular monoidal language consists of the string diagrams with no dangling wires that can be built using the "typed" building blocks of a grammar, then forgetting the types:

Example 2.4 (Sierpiński triangles)**.** *Self-assembly of DNA tiles has been used to compute Sierpiński triangle fractals of arbitrary iteration depth [7]. This work implicitly uses a monoidal grammar (left below, where we use colours for the alphabet). An element in the language defined by this grammar is shown on the right.*



In the following examples, we consider monoidal languages up to their *connected* string diagrams:

Example 2.5 (Balanced parentheses)**.** *Recall that the Dyck language is a paradigmatic example of a non-regular word language. However, we can represent balanced parentheses using the regular monoidal grammar shown below left. An example of a morphism in the language defined by this grammar is shown on the right.*



This shows how regular monoidal grammars permit unbounded concurrency. As one scans from left to right, the (unbounded) size of the internal boundary of a string diagram tracks the number of open left parentheses. This also indicates an interesting connection between regular monoidal languages and context-free languages that we will examine in future work.

Example 2.6. *Regular grammars are monoidal grammars over alphabets having only generators* σ *of arity and coarity 1, along with a start generator* $\vdash$ *and end generator* $\dashv$*.*

Example 2.7. *Regular tree grammars are monoidal grammars over alphabets having only generators* *of arity 1 (and coarity* $\geqslant 0$*), along with a single "root" generator* $\vdash$ *.*

Remark 2.8. *Regular monoidal languages enjoy several closure properties: intersection, union, monoidal product/factors, images and preimages of alphabets, but not complementation.*

3 Monoidal automata and convex monoidal automata

Monoidal grammars can also be considered as depicting transition graphs of *monoidal automata*:

Definition 3.1. *A non-deterministic monoidal automaton* $\Delta = (Q, \Delta_\Gamma)$ *over a monoidal alphabet* Γ *is given by a finite set* Q*, together with a set of transition functions indexed by generators* $\Delta_\Gamma = \{Q^{ar(\gamma)} \xrightarrow{\Delta_\gamma} \mathscr{P}(Q^{coar(\gamma)})\}_{\gamma \in \Gamma}$*.*

The inductive extension of a monoidal automaton from generators to string diagrams determines a unique pro morphism $\Delta : \mathscr{F}\Gamma \to \mathsf{Rel}_Q$, where a morphism $n \to m$ in Rel_Q is a relation $R \subseteq Q^n \times Q^m$. Δ maps a string diagram with no dangling wires to one of the two relations from Q^0 to Q^0 (accepting or rejecting) and thus define the language of the automaton.

Remark 3.2. *The non-deterministic monoidal automata determined by the grammars of Examples 2.6 and 2.7 correspond to classical non-deterministic automata over words and trees.*

Deterministic monoidal automata can be defined in a similar way. Since top-down tree automata are not determinizable [5], monoidal automata are not determinizable in general, and it is an intriguing theoretical challenge to characterize the deterministically recognizable monoidal languages. Below we describe a class of determinizable monoidal automata, and give a powerset construction.

Definition 3.3. *A relation* $\Delta : Q^n \to \mathscr{P}(Q^m)$ *is convex if there is a morphism* $\Delta^* : \mathscr{P}(Q)^n \to \mathscr{P}(Q)^m$ *such that* $\nabla_m \circ \Delta^* = \Delta^\# \circ \nabla_n$*, where* $\Delta^\# : \mathscr{P}(Q^n) \to \mathscr{P}(Q^m)$ *is the Kleisli lift of* Δ*, and* $\nabla_n : \mathscr{P}(Q)^n \to \mathscr{P}(Q^n)$ *is the monoidal multiplication of the monoidal monad* $\mathscr{P}$*.*

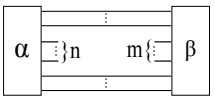
Convex relations determine a sub-pro $\mathsf{CRel}_Q \hookrightarrow \mathsf{Rel}_Q$, and an automaton $\Delta : \mathscr{F}\Gamma \to \mathsf{Rel}_Q$ is *convex* if it factors through CRel_Q. Convex automata can be determinized by a form of powerset construction. In particular, bottom-up regular tree automata and word automata seen as monoidal automata (Remark 3.2) are convex.

We have also shown that deterministically recognizable monoidal languages have the property of *causal closure*. Roughly this means they contain all the string diagrams that can be written as monoidal products of tree-like sub-diagrams, subject to the equational theory of cartesian restriction categories [3].

4 Syntactic pro gives sufficient condition for determinism

By analogy with the syntactic monoid of a word language, the syntactic pro associated to a monoidal language identifies string diagrams up to contextual equivalence. When this pro has *cartesian restriction category* structure [3], the language is determinizable (Theorem 4.4).

Definition 4.1. *An* (n, m)*-context is a* $0 \to 0$ *string diagram with a hole of size (n,m), where* $n, m \geqslant 0$ *(depicted right). Given an* (n, m)*-context, we can fill the hole with a morphism* $\gamma : n \to m$*. Write* $C[\gamma]$ *for the resulting string diagram.*



Definition 4.2. *The syntactic congruence $\equiv_L$ of a monoidal language L over Γ is defined by $\alpha \equiv_L \beta$ whenever $C[\alpha] \in L \iff C[\beta] \in L$, for all (n,m)-contexts C, where $\alpha, \beta \in \mathscr{F}\Gamma(n,m)$.*

Definition 4.3. *The syntactic pro of a monoidal language L is the quotient pro $\mathscr{F}\Gamma/\equiv_L$.*

Theorem 4.4. *If the syntactic pro of a regular monoidal language is a cartesian restriction prop (a pro with symmetric monoidal structure), then the language is recognizable by a deterministic monoidal automaton.*

We also have one direction of a Myhill-Nerode type theorem:

Lemma 4.5. *If a monoidal language L is regular, its syntactic pro $\mathscr{F}\Gamma/\equiv_L$ has finite hom-sets.*

References

[1] Mikołaj Bojańczyk, Bartek Klin, and Julian Salamanca. Monadic monadic second order logic. 2022.

[2] Filippo Bonchi, Fabio Gadducci, Aleks Kissinger, Pawel Sobocinski, and Fabio Zanasi. String diagram rewrite theory I: Rewriting with Frobenius structure. *J. ACM*, 69(2), mar 2022.

[3] Ivan Di Liberti, Fosco Loregian, Chad Nester, and Paweł Sobociński. Functorial semantics for partial theories. *Proc. ACM Program. Lang.*, 5(POPL), January 2021.

[4] Matthew Earnshaw and Paweł Sobociński. Regular monoidal languages. *arXiv*, 2022.

[5] Ferenc Gécseg and Magnus Steinby. Tree automata, 2015.

[6] Aleks Kissinger and Vladimir Zamdzhiev. Quantomatic: A proof assistant for diagrammatic reasoning. In *Automated Deduction - CADE-25*, pages 326–336. Springer, 2015.

[7] Paul W. K Rothemund, Nick Papadakis, and Erik Winfree. Algorithmic self-assembly of DNA Sierpiński triangles. *PLOS Biology*, 2(12), 12 2004.

[8] R.F.C. Walters. A note on context-free languages. *Journal of Pure and Applied Algebra*, 62(2):199–203, 1989.

[9] Vladimir Zamdzhiev. *Rewriting Context-free Families of String Diagrams.* PhD thesis, University of Oxford, 2016.