Regular Monoidal Languages

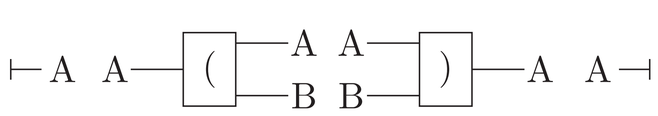
arXiv:2207.00526, to appear at MFCS 2022

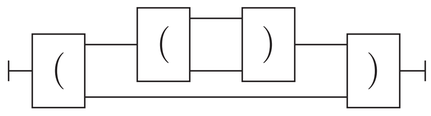
Matthew Earnshaw and Paweł Sobociński
Department of Software Science, Tallinn University of Technology, Estonia

Language theory studies finite means for generating and recognizing infinite sets. It extends to sets of various structures: words, trees, infinite words, graphs of bounded tree width, etc. In this paper we make a first step in extending language theory to higher-dimensional algebraic structures, moving from monoids (words) to monoidal categories: our languages are sets of morphisms in free pros. The algebra of monoidal and cartesian restriction categories is used to investigate the theoretical properties of such languages including closure properties and deterministic recognizability.

Regular Monoidal Grammars and Languages

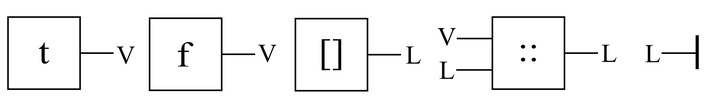
Classically, a language L over an alphabet Σ is a subset of the free monoid $L \subseteq \Sigma^*$. A *monoidal language* is defined similarly, replacing free monoids with free pros, and considering subsets of the hom-set of scalars, $L \subseteq \mathscr{F}\Gamma(0,0)$. The *regular monoidal languages* are defined by finite, labelled monoidal graphs (signatures): these are *regular monoidal grammars*. For example, consider the following regular monoidal grammar (left below), and an element in the language of this grammar (right below):



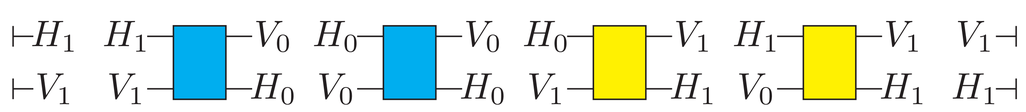


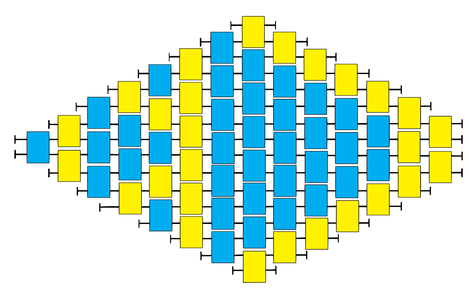
Intuitively, the language defined by a grammar consists of the untyped scalar string diagrams that can be built from the typed building blocks of the grammar. The grammar above defines a language of balanced parentheses and illustrates how regular monoidal grammars permit unbounded concurrency. As one scans from left to right, the size of the internal boundary of a string diagram keeps track of the number of open left parentheses.

Classical regular word and tree languages are regular monoidal languages over grammars of a particular shape. For example, consider the following regular monoidal grammar, having only generators of coarity 1, plus one "root" generator (right). The elements of this language are trees corresponding to terms of the inductive type of lists of boolean values:



Regular monoidal languages can also be found implicitly in DNA computing, where self-assembly of DNA tiles has been used to realize the computation of Sierpiński fractals. Such tiles are represented in the following regular monoidal grammar (left below), and the elements of the language are Sierpiński triangles of arbitrary iteration depth (example right below):





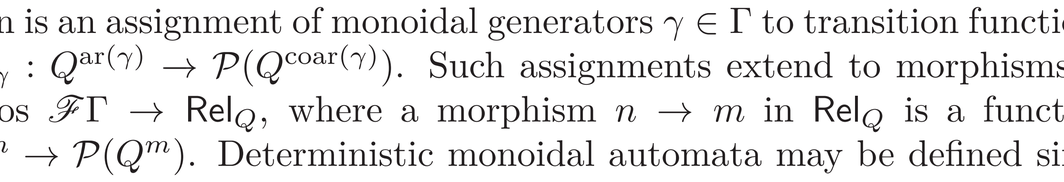
Regular monoidal languages enjoy several closure properties: under union, intersection, monoidal product and factors, homomorphic images, and homomorphic preimages. They are not however closed under complement, since every regular monoidal language contains the empty diagram.

Monoidal Automata

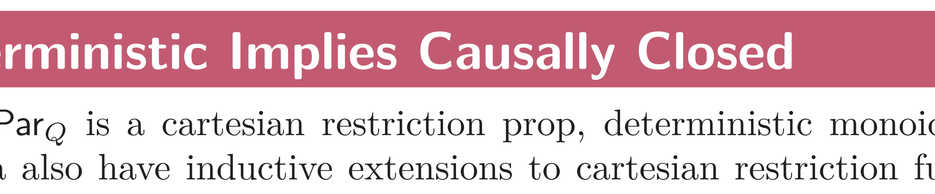
Regular monoidal grammars can also be seen as transition graphs of *monoidal automata*. Formally, a (non-deterministic) monoidal automaton is an assignment of monoidal generators $\gamma \in \Gamma$ to transition functions $\Delta_\gamma : Q^{\mathrm{ar}(\gamma)} \to \mathcal{P}(Q^{\mathrm{coar}(\gamma)})$. Such assignments extend to morphisms of pros $\mathscr{F}\Gamma \to \mathsf{Rel}_Q$, where a morphism $n \to m$ in Rel_Q is a function $Q^n \to \mathcal{P}(Q^m)$. Deterministic monoidal automata may be defined similarly, replacing the powerset monad with the maybe monad. The inductive extensions are then morphisms of pros $\mathscr{F}\Gamma \to \mathsf{Par}_Q$.

Deterministic Implies Causally Closed

Since Par_Q is a cartesian restriction prop, deterministic monoidal automata also have inductive extensions to cartesian restriction functors $\mathscr{F}_\downarrow\Gamma \to \mathsf{Par}_Q$, where $\mathscr{F}_\downarrow$ denotes the free cartesian restriction prop. This "automaton" accepts an extended language including tree-like subdiagrams that we call causal histories. For example, the diagrams below are causal histories of the rightmost one (formally, $A \otimes h = A$ for all h):



The equational theory of cartesian restriction categories implies that some (monoidal products of) causal histories are also string diagrams in $\mathscr{F}\Gamma$, for example:



By splitting a language into its causal histories and recombining these, we obtain its *causal closure*. We show that all languages recognized by deterministic monoidal automata are equal to their causal closure.

Convex Automata and Determinization

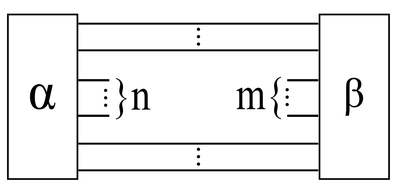
Since root-to-leaf tree languages cannot be deterministically recognized, neither can all monoidal languages. However, monoidal automata factoring through the sub-pro of *convex* relations admit a generalization of powerset construction (a determinization algorithm). A relation Δ is convex if there is a morphism Δ^* making the following square commute:

$$\begin{array}{ccc} (\mathcal{P}Q)^n & \xrightarrow{\Delta^*} & (\mathcal{P}Q)^m \\ {\scriptstyle\nabla}\downarrow & & \downarrow{\scriptstyle\nabla} \\ \mathcal{P}(Q^n) & \xrightarrow{\Delta^\#} & \mathcal{P}(Q^m) \end{array}$$

where $\Delta^\#$ is the Kleisli lift of Δ, and ∇ is the monoidal multiplication of the powerset monad.

The Syntactic Pro

The *syntactic monoid* is an invariant of classical languages, isomorphic to the transition monoid of the minimal automaton. It identifies words up to contextual equivalence. We extend this to monoidal languages using string diagram *contexts*. A context is a diagram with a hole:



Given a context C of capacity (n,m), we obtain a scalar string diagram $C[\alpha]$ by plugging in a string diagram $\alpha : n \to m$. Contexts define an equivalence relation on the morphisms of a pro: morphisms $\alpha, \beta : n \to m$ are equivalent if $C[\alpha] \in L \iff C[\beta] \in L$, for all contexts C of capacity (n,m). The corresponding quotient pro is the *syntactic pro*.

We show that if L is a regular monoidal language then its syntactic pro has finite homsets, and that if the syntactic pro has the structure of a cartesian restriction prop, then the language is deterministically recognizable.