ABSTRACT ALGEBRA AND THE SUBREGULAR HIERARCHY

Dakotah Lambert

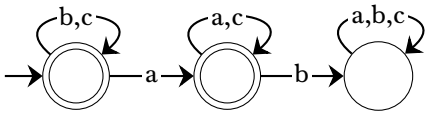
Algebra is the study of structure. Using algebraic techniques, we can study the structure of phonotactic patterns and phonological transformations.

1 Semigroups

A semigroup is an algebraic structure: a set S and an associative binary relation. At core, a semigroup describes how things act. Fix a set of basic concatenable units Σ (an ALPHABET, though it need not consist of anything representing letters), then Σ^* is the set of all possible concatenations of finitely many things. Call those 'strings', no matter how unstringlike they look. The set of all strings is the FREE SEMIGROUP.

In phonology, one way to decide whether phones represent different phonemes is to seek a CONTEXT $u\_\_v$ that distinguishes them, yielding a different meaning. For example, length is contrastive in Finnish, as evidenced by /*tuli*/ 'fire' and /*tu:li*/ 'wind'. Strings are distinguished in a language iff there is a context that distinguishes them by *membership*.

Consider the language of all and only those words which do not contain an 'a..b' subsequence. That is, if there is an 'a' somewhere, there is not a 'b' anywhere later in the word. As a finite-state acceptor:



Notice that strings 'a' and 'ba' lead to the same state. But they are distinguished algebraically, because the 'a__' context distinguishes them: 'a<u>a</u>' is accepted but 'a<u>ba</u>' is not.

The free semigroup has infinitely many elements, but if a language is regular, then there are only finitely many equivalence classes. Remember: semigroups are about *actions*. The equivalence classes, the elements of the syntactic semigroup, can be derived from the canonical finite-state machine by looking at the state-to-state functions of words. Begin with the letters – here, 'a' maps states 1, 2 and 3 to 2, 2 and 3, respectively. Write that as $f_a = \langle 2, 2, 3\rangle$. Then 'b' corresponds to $f_b = \langle 1, 3, 3\rangle$ and 'c' corresponds to $f_c = \langle 1, 2, 3\rangle$. Concatenation xy corresponds to reversed composition $f_y \circ f_x$. So $f_{ba} = f_a \circ f_b = \langle 2, 3, 3\rangle \neq f_a$. By the way, the same procedure works for canonical deterministic transducers: ignore the output information and look at the state-to-state functions of the input words.

2 Classification

To classify languages or processes by their algebraic properties is to determine whether their semigroup multiplication satisfies certain constraints. For example, consider the k-DEFINITE languages, where whenever two words have the same suffix of length k, either both are accepted or both are rejected.[1]

In a definite language, if $a = a'x_1 \dots x_k$ and $x = x_1 \dots x_k$ share the suffix $x_1 \dots x_k$, then either a and x are both accepted, or both are rejected. Then av and xv also share the same suffix of length k (although not necessarily the $x_1 \dots x_k$ suffix). Prepending more material won't change the suffix, so $u\underline{ax}v$ and $u\underline{x}v$ are either both accepted or both rejected, no matter what u and v are. In other words, $ax \equiv x$. So in a k-definite language, the semigroup satisfies the equation $ax_1 \cdots x_k = x_1 \cdots x_k$.

[1] For shorter words, the k-suffix is the whole word.

We can abstract away from the length parameter k. Let e be an IDEMPOTENT, an element where $ee = e$. Then $ae = aee = \cdots = ae^k$. The empty string is not included in the semigroup, so certainly ae^k and e^k share the same suffix of length at least k. In a definite language then, $ae = ae^k = e^k = e$. (It takes a bit more work to show that everything that satisfies this is definite.) The idempotents are exactly the limits of the form $y^\omega = \lim_{n\to\infty} y^{n!}$, so we write this as follows.

$$\mathbf{D} = [\![xy^\omega = y^\omega]\!]$$

The definite languages correspond to a propositional logic in which literals are length-k suffixes (or shorter words). Such a language can always be written as a set of forbidden suffixes (a conjunction of negative literals) or as a set of permissible suffixes (a disjunction of positive literals). But this class, and every other class characterized by equations, is closed under the Boolean operations.[2]

One can convert any finite semigroup S into a finite automaton for a language with the same structural properties. Make a state for every element. If there is a neutral element 1 where $1x = x = x1$, this is the start state; otherwise introduce such an element as a new distinguished start state. Pick a set G of generators, where $T = G \cup \{xy : x \in T, y \in T\}$ is equal to S. Draw an edge from state $q \in S$ to state $r \in S$ labeled $g \in G$ whenever $qg = r$. Select any set of states as accepting, and the result is a language over G^* that is in the class. That is:

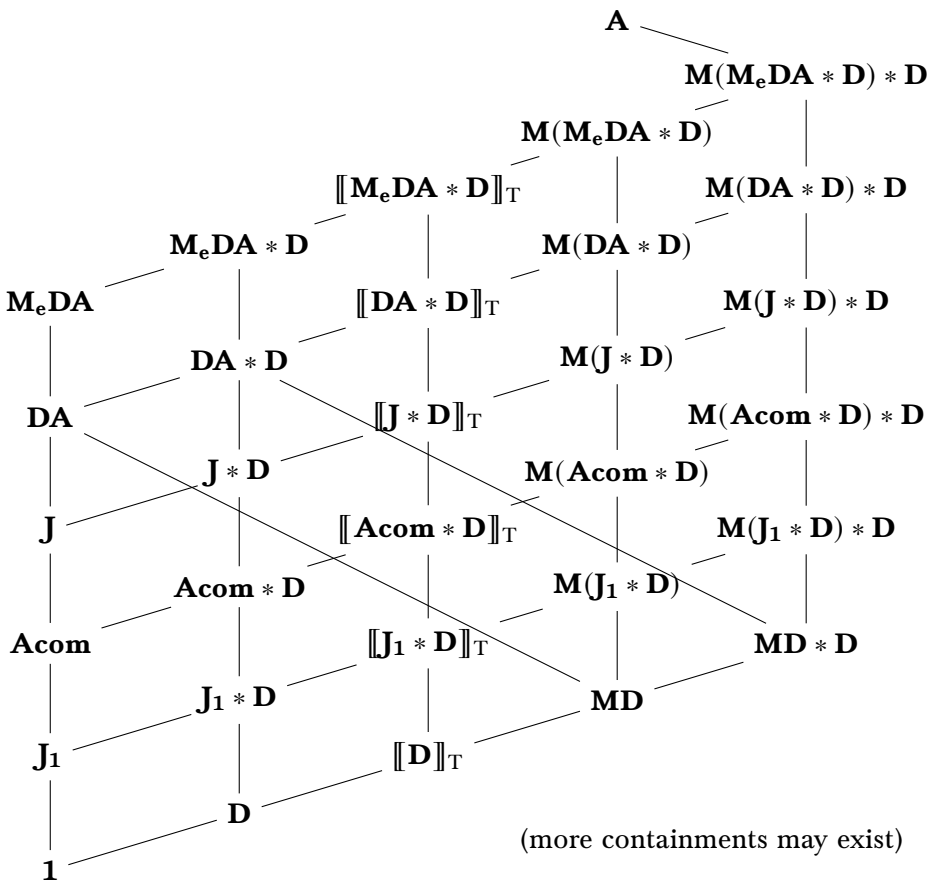
> *Every state or set of states in a finite-state acceptor or transducer corresponds to the set of input-strings that satisfy some formula in the associated logic(s).*

A sampler of other classes (Decision procedures all implemented in software [7].)[3]

- Locally 1-testable = piecewise 1-testable = $\mathbf{J_1} = [\![xy = yx; x^2 = x]\!]$ (Prop[$\emptyset$]) [2]
- Locally testable = $\mathbf{J_1} * \mathbf{D} = [\![a^\omega xa^\omega ya^\omega = a^\omega ya^\omega xa^\omega; (a^\omega xa^\omega)^2 = a^\omega xa^\omega]\!]$ (Prop[$\lhd$]) [2]
- Piecewise testable = $\mathbf{J} = [\![y(xy)^\omega = (xy)^\omega = (xy)^\omega x]\!]$ (Prop[$<$]) [10]
- Dot-depth one = $\mathbf{J} * \mathbf{D} = [\![(x^\omega az^\omega b)^\omega x^\omega az^\omega dx^\omega (cz^\omega dx^\omega)^\omega = (x^\omega az^\omega b)^\omega x^\omega (cz^\omega dx^\omega)^\omega]\!]$ (Prop[$<, \lhd$]) [5]
- $\mathbf{DA} = [\![(xyz)^\omega y(xyz)^\omega = (xyz)^\omega]\!]$ (FO^2[$<$]) [12]
- $\mathbf{DA} * \mathbf{D} = [\![(a^\omega xa^\omega ya^\omega za^\omega)^\omega a^\omega ya^\omega (a^\omega xa^\omega ya^\omega za^\omega)^\omega = (a^\omega xa^\omega ya^\omega za^\omega)^\omega]\!]$ (FO^2[$<, \lhd$]) [12]
- $\mathbf{M_e DA}$[4] (FO^2[$<$, bet]) [6]
- $\mathbf{M_e DA} * \mathbf{DA}$ (FO^2[$<$, betfac]) [6]
- Locally threshold 1-testable = $\mathbf{Acom} = [\![xy = yx; xx^\omega = x^\omega]\!]$
- Locally threshold testable = $\mathbf{Acom} * \mathbf{D} = [\![x^\omega az^\omega bx^\omega cz^\omega = x^\omega cz^\omega bx^\omega az^\omega; xx^\omega = x^\omega]\!]$ (FO[$\lhd$]) [1]
- Aperiodic = star-free = $\mathbf{A} = [\![xx^\omega = x^\omega]\!]$ (FO[$<$]) [9]

Tiers: Let $L \in \mathbf{V}$ be a language over $\Gamma \subseteq \Sigma$. Define π to be the homomorphism that deletes letters not in Γ. Then $\pi^{-1}(L) = \{w \in \Sigma^* : \pi(w) \in L\}$ is in tier-based $\mathbf{V}$, here written $[\![\mathbf{V}]\!]_{\mathrm{T}}$. Test by removing 1, if present, then closing under multiplication, then verifying the equations of $\mathbf{V}$.
Multitier: close $[\![\mathbf{V}]\!]_{\mathrm{T}}$ under the Boolean operations again to get an algebraically more natural structure, here written $\mathbf{MV}$.

[2] Ordered semigroups let you avoid closure under complement [8], but they are beyond today's scope.
[3] The Language Toolkit / `plebby` at `https://hackage.haskell.org/package/language-toolkit`
[4] $\Sigma_e = \{x : e \in \{x\} \cup xS \cup Sx \cup SxS\}$; $M_e = \Sigma_e^*$; $\mathbf{M_e V}$ means $eM_e e \in \mathbf{V}$ for all idempotents e.



The $\mathbf{V} * \mathbf{D}$ classes are inverse images under definite (total ISL) functions of languages that are in $\mathbf{V}$. In other words, wherever $\mathbf{V}$ looks at letters, $\mathbf{V} * \mathbf{D}$ looks at k-wide factors [11]. From an algebraic perspective, precedence is fundamental, while locality is derived.

3 Continuity

The preceding discussion focused on structural properties of languages, which extend naturally to subsequential functions via the automata-based characterizations. Another way to think about function classes is by what properties are preserved under inverse images.

In analysis, a function is CONTINUOUS iff the inverse images of open sets are open. We say that a word-to-word function is $\mathbf{V}$-continuous iff the inverse images of languages in $\mathbf{V}$ are in $\mathbf{V}$ [3]. For every class $\mathbf{V}$ in the first or fourth columns of the above diagram (the "monoidal varieties"), except possibly $\mathbf{1}$ and $\mathbf{J_1}$, the class of $\mathbf{V}$-continuous functions is the largest class C where the following hold: [3]

- If $L \subseteq \Sigma^*$ is in $\mathbf{V}$, then the characteristic function χ_L is in C

$$\chi_L(w) = \begin{cases} 1 & \text{if } w \in L \\ 0 & \text{otherwise} \end{cases}$$

- C is closed under composition.

Something special about aperiodicity is that any subsequential function that is structurally $\mathbf{A}$ is also $\mathbf{A}$-continuous; however, there are $\mathbf{A}$-continuous functions that are not structurally $\mathbf{A}$ [3]. Also, $\mathbf{A}$ corresponds to Courcellian first-order transductions (order-preserving, if subsequential) [4].

References

[1] Danièle Beauquier and Jean-Éric Pin. Languages and scanners. *Theoretical Computer Science*, 84(1):3–21, July 1991.

[2] Janusz Antoni Brzozowski and Imre Simon. Characterizations of locally testable events. *Discrete Mathematics*, 4(3):243–271, March 1973.

[3] Michaël Cadilhac, Olivier Carton, and Charles Paperman. Continuity of functional transducers: A profinite study of rational functions. *Logical Methods in Computer Science*, 16(1):24:1–24:29, February 2020.

[4] Emmanuel Filiot, Olivier Gauwin, and Nathan Lhote. First-order definability of rational transductions: An algebraic approach. In *LICS '16: Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 387–396. Association for Computing Machinery, July 2016.

[5] Robert Knast. A semigroup characterization of dot-depth one languages. *RAIRO – Informatique théorique*, 17(4):321–330, 1983.

[6] Andreas Krebs, Kamal Lodaya, Paritosh K. Pandya, and Howard Straubing. Two-variable logics with some betweenness relations: Expressiveness, satisfiability, and membership. *Logical Methods in Computer Science*, 16(3):1–41, September 2020.

[7] Dakotah Lambert. System description: A theorem-prover for subregular systems: The Language Toolkit and its interpreter, plebby. In Jeremy Gibbons and Dale Miller, editors, *Functional and Logic Programming: 17th Annual Symposium, FLOPS 2024*, volume 14659 of *Lecture Notes in Computer Science*, pages 311–328, Kumamoto, Japan, May 2024. Springer, Singapore.

[8] Jean-Éric Pin. Syntactic semigroups. In Grzegorz Rozenberg and Arto Salomaa, editors, *Handbook of Formal Languages: Volume 1 Word, Language, Grammar*, pages 679–746. Springer-Verlag, Berlin, 1997.

[9] Marcel-Paul Schützenberger. On finite monoids having only trivial subgroups. *Information and Control*, 8(2):190–194, April 1965.

[10] Imre Simon. Piecewise testable events. In Helmut Brakhage, editor, *Automata Theory and Formal Languages*, volume 33 of *Lecture Notes in Computer Science*, pages 214–222. Springer-Verlag, Berlin, 1975.

[11] Howard Straubing. Finite semigroup varieties of the form $V * D$. *Journal of Pure and Applied Algebra*, 36:53–94, 1985.

[12] Denis Thérien and Thomas Wilke. Over words, two variables are as powerful as powerful as one quantifier alternation: $\mathrm{FO}^2 = \Sigma_2 \cap \Pi_2$. In *STOC '98: Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, pages 234–240, New York, New York, 1998. Association for Computing Machinery.