Algebraic Classification of Reduplicative Processes

ACL 2026 SAN DIEGO JULY 2–7

Matthew Hayden

Stony Brook University

Overview

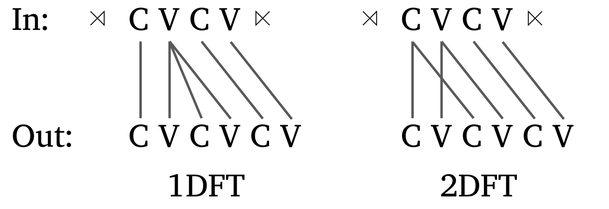
- Morphological processes are known to be computationally simple, modeled by 1-way finite state transducers (1DFTs), **except** total reduplication[9]
- I draw on algebraic methods to show that reduplication is subregular even though it can't be computed by a 1DFT
- Classifying the RedTyp[5] database reveals reduplication falls into subregular classes **𝔻𝔸** (100%), **𝕃𝕁₁** (98%), and **𝕃1** (87%).

2DFTs for Reduplication

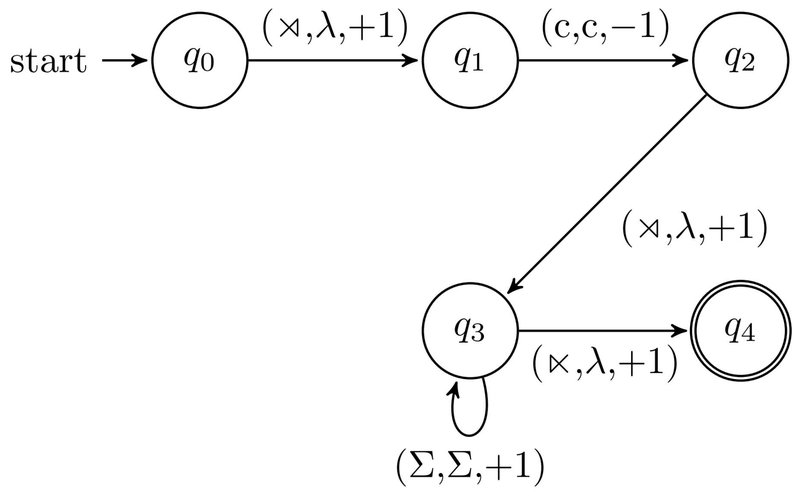
- Reduplication comes in two flavors
  - Total: [buku] ↦ [buku buku]

    book *books* (Indonesian)[4]
  - Partial: [liw] ↦ [lliw]

    angry *scold* (Agta)[8]
- Despite both kinds being common, neither can be computed effectively under the correct **origin semantics**[2]



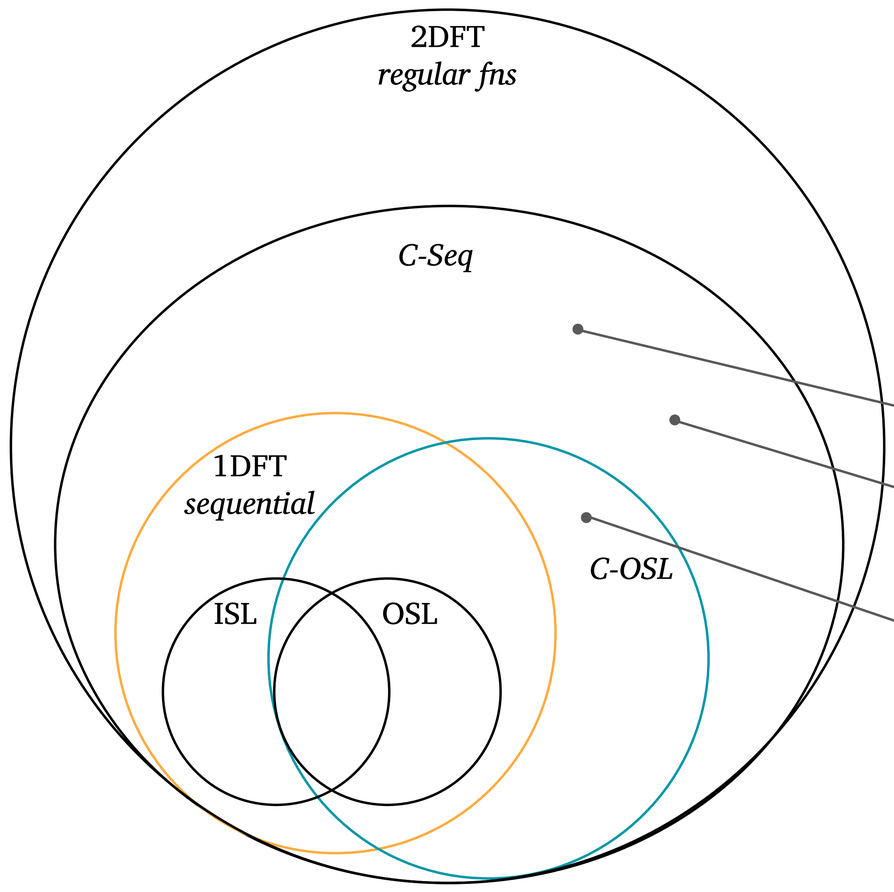
- 2-way finite-state transducers (2DFTs) gain extra power by moving forward and backward on the input string, allowing them to compute both types[6]



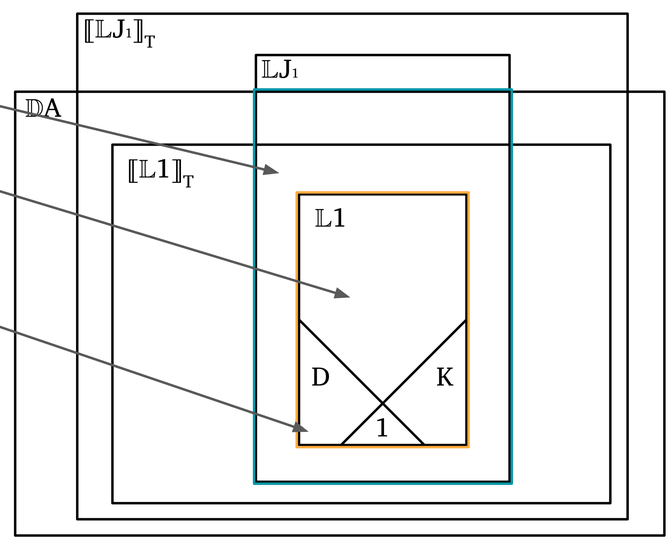
Ex. 2DFT for Initial-C Reduplication in Agta

RedTyp Database

- RedTyp[5] contains 2DFTs for 138 reduplicative processes, from 91 languages, totalling 57 distinct 2DFTs
- Designed for typological breadth
- Vast majority known to belong to the class of concatenated output strictly-local maps (C-OSL), though some require even more power[6]
- However, C-OSL is not well-behaved mathematically, containing functions with arbitrarily complex algebras[7]
- Algebraic classification reveals a tighter, properly subregular characterization of reduplication



Paper → Code ←

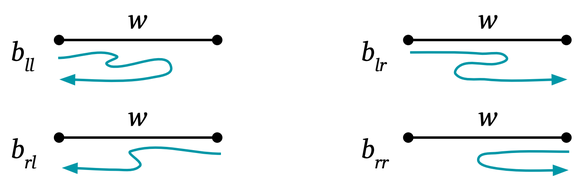


Algebraic Classification

- 2DFTs admit special finite algebras called **transition semigroups**[3] that encode the possible state transitions
- Every input string over the alphabet is associated with an element, collapsing the space into a finite object
- Reasoning over the algebra allows us to measure the functional complexity
- Membership in distinct subregular classes corresponds to satisfaction of distinct algebraic identities

String Behaviors

- A string w can be processed four ways



- These are called the string's **behaviors**
- Formally, these are state pairs (p,q) where it starts processing w in state p and exits w in state q
- Elements in the transition semigroup correspond to the possible behaviors

Concatenation of Behaviors

- While 1DFT semigroups use ordinary concatenation, 2DFTs semigroups use a more sophisticated operation[1]:

$$\mathrm{B}(u) \cdot \mathrm{B}(v) =$$
$$\langle\ b_{ll}(u) \cup b_{lr}(u)[b_{ll}(v)b_{rr}(u)]^* b_{ll}(v)b_{rl}(u),$$
$$b_{lr}(u)[b_{ll}(v)b_{rr}(u)]^* b_{lr}(v),$$
$$b_{rl}(v)[b_{rr}(u)b_{ll}(v)]^* b_{rl}(u),$$
$$b_{rr}(v) \cup b_{rl}(v)[b_{rr}(u)b_{ll}(v)]^* b_{rr}(u)b_{lr}(v)\ \rangle$$

Then $\mathrm{B}(u) \cdot \mathrm{B}(v) = \mathrm{B}(uv)$

- The behaviors of the alphabet symbols form the basis of the semigroup, so they generate all elements
- Ex. for the Agta transducer, we have:

$\mathrm{B}(c)$: $\langle\{(q_1, q_2)\}, \{(q_3, q_3)\}, \{(q_1, q_2)\}, \{(q_3, q_3)\}\rangle$

$\mathrm{B}(v)$: $\langle\emptyset, \{(q_1, q_2), (q_3, q_3)\}, \emptyset, \{(q_1, q_2), (q_3, q_3)\}\rangle$

$\mathrm{B}(c) \cdot \mathrm{B}(v) = \langle\{(q_1, q_2)\}, \{(q_3, q_3)\}, \emptyset, \{(q_3, q_3)\}\rangle$

- The structure summarized as a table:

•	C	V	CV
C	CV	CV	CV
V	V	V	V
CV	CV	CV	CV

Algebraic Identities

- We say a is **idempotent** if $aa = a$. a^ω denotes a's unique idempotent power. Let S be a semigroup. Then S is…

➔ in **1** iff $x = y$
➔ in **D** iff $xa^\omega = a^\omega$
➔ in **K** iff $a^\omega x = a^\omega$
➔ in **𝕃1** iff $a^\omega x a^\omega = a^\omega$
➔ in **𝕃𝕁₁** iff $a^\omega x a^\omega y a^\omega = a^\omega y a^\omega x a^\omega$
➔ in **𝔻𝔸** iff $(xyz)^\omega y (xyz)^\omega = (xyz)^\omega$

holds for all $x, y, z, a \in S$.[7]

- Ex. the Agta semigroup satisfies **K**

Automatic Classification

- My Python script automatically computes and tests semigroups
- Application to RedTyp shows reduplication is no more complex than other attested morphological processes

Select references **[1] Jean-Camille Birget**. 1989. Concatenation of inputs in a two-way automaton. Theoretical Computer Science, 63(2):141–156. **[2] Mikołaj Bojańczyk**. 2014. Transducers with origin information. In Automata, Languages, and Programming: 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part II 41, pages 26–37. Springer. **[3] Olivier Carton and Luc Dartois**. 2015. Aperiodic two-way transducers and FO-transductions. In 24th EACSL Annual Conference on Computer Science Logic, volume 41, pages 160–174. **[4] Abigail C. Cohn**. 1989. Stress in Indonesian and bracketing paradoxes. Natural Language & Linguistic Theory, 7(2):167–216. **[5] Hossep Dolatian and Jeffrey Heinz**. 2019. RedTyp: A database of reduplication with computational models. In Proceedings of the Society for Computation in Linguistics (SCiL) 2019, pages 8–18. **[6] Hossep Dolatian and Jeffrey Heinz**. 2020. Computing and classifying reduplication with 2-way finite-state transducers. Journal of Language Modelling, 8(1):179–250. **[7] Dakotah Lambert**. 2022. Unifying Classification Schemes for Languages and Processes with Attention to Locality and Relativizations Thereof. Ph.D. thesis, State University of New York at Stony Brook. **[8] Edith A. Moravcsik**. 1978. Reduplicative constructions. In Joseph H. Greenberg, Charles A. Ferguson, and Edith A. Moravcsik, editors, Universals of Human Language. Volume 3: Word Structure, pages 297–334. Stanford University Press, Stanford, CA. **[9] Brian Roark and Richard Sproat**. 2007. Computational approaches to morphology and syntax. OUP Oxford.