Logic, Automata, and Algebra, Together Again

Dakotah Lambert

23 April 2024

software

slides

Intro: Formal languages

Assume finite alphabet A

Formal language partitions words (A^+) into two classes: accepted / rejected

Intro: Formal languages and classification

Assume finite alphabet A

FORMAL LANGUAGE partitions words (A^+) into two classes:
accepted / rejected

CLASSES: sets of languages with some shared property

Intro: Applications

- Language modelling
- Machine learning and learnability
- Robotic planning and control
- Software security
  - Can’t correctly parse with something too weak
  - But too much power creates and exposes vulnerabilities

    (Sassaman *et al.*, 2013)

Intro: Some types of classes

- Logical
  - “Definable with first-order logic on strings with adjacency”
  - Prove: give the formula
  - Disprove: Ehrenfeucht-Fraïssé games, …

Intro: Some types of classes

- Logical
  - "Definable with first-order logic on strings with adjacency"
  - Prove: give the formula
  - Disprove: Ehrenfeucht-Fraïssé games, ...
- Language-theoretic
  - "Words with the same count, up to some threshold t, of k-wide substrings are treated the same"
  - Prove: generalize on the infinite set of words
  - Disprove: provide (parameterized) words violating property

Intro: Some types of classes

- Logical
  - "Definable with first-order logic on strings with adjacency"
  - Prove: give the formula
  - Disprove: Ehrenfeucht-Fraïssé games, …
- Language-theoretic
  - "Words with the same count, up to some threshold t, of k-wide substrings are treated the same"
  - Prove: generalize on the infinite set of words
  - Disprove: provide (parameterized) words violating property
- Algebraic
  - "Semigroup is finite and satisfies both $x^\omega a y^\omega b x^\omega c y^\omega = x^\omega c y^\omega b x^\omega a y^\omega$ and $x x^\omega = x^\omega$"
  - Prove: verify equations for all instantiations
  - Disprove: find an instantiation breaking an equation

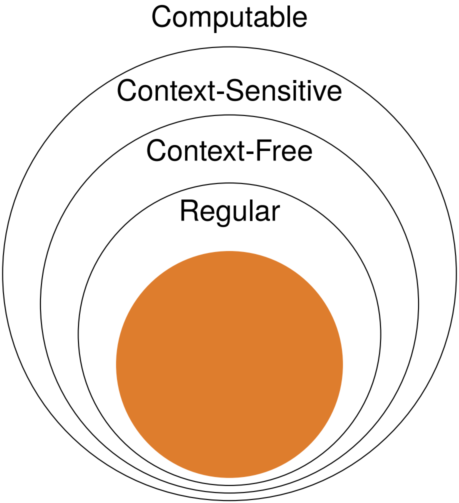
Intro: Why algebra?

Often want logical or language-theoretic descriptions.

So why algebra?

- Easy to prove and disprove membership
- Equations make subclass relations easy to show
- Unified approach for all applicable classes

Intro: The Chomsky hierarchy (Chomsky, 1959)



Semigroups

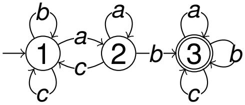
SEMIGROUP: a set S with multiplication

- xy always exists
- $(xy)z = x(yz)$ so xyz makes sense

Strings are a semigroup but infinite

Semigroups: Equivalence classes

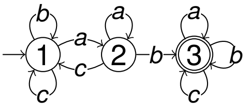
$$A = \{a, b, c\}$$



$$L_{\text{good}} = \{w \in A^+ : w \text{ contains } ab\}$$
$$L_{\text{bad}} = A^+ - L_{\text{good}}$$

Semigroups: Equivalence classes

$$A = \{a, b, c\}$$



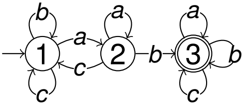
$$L_{\text{good}} = \{w \in A^{+} : w \text{ contains } ab\}$$
$$L_{\text{bad}} = A^{+} - L_{\text{good}}$$

not compatible with concatenation

$a \sim b$ but $a\,a \not\sim a\,b$

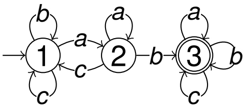
Semigroups: Equivalence classes

$$A = \{a, b, c\}$$



Semigroups: Equivalence classes

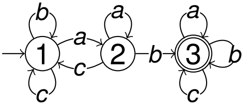
$$A = \{a, b, c\}$$



$$L_3 = \{w \in A^+ : w \text{ contains } ab\}$$
$$L_2 = \{w \in A^+ : w \text{ ends with } a\} - L_3$$
$$L_1 = A^+ - L_2 - L_3$$

Semigroups: Equivalence classes

$$A = \{a, b, c\}$$



$$L_3 = \{w \in A^+ : w \text{ contains } ab\}$$
$$L_2 = \{w \in A^+ : w \text{ ends with } a\} - L_3$$
$$L_1 = A^+ - L_2 - L_3$$

still not compatible with concatenation

$a \sim ba$ but $a\,a \not\sim a\,ba$

Semigroups: Equivalence classes

Nerode: $u \sim v$ means, for all y,
uy is accepted if and only if vy is

Semigroups: Equivalence classes

Nerode: $u \sim v$ means, for all y,
uy is accepted if and only if vy is

Myhill: $u \sim v$ means, for all x, y,
xuy is accepted if and only if xvy is

Semigroups: Equivalence classes

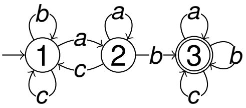
Nerode: $u \sim v$ means, for all y,
uy is accepted if and only if vy is

Myhill: $u \sim v$ means, for all x, y,
xuy is accepted if and only if xvy is

Myhill compatible with concatenation:
$u \sim v$ and $x \sim y$ implies $ux \sim vy$
(ux is good $\leftrightarrow$ vx is good $\leftrightarrow$ vy is good)

Semigroups: Equivalence classes

$$A = \{a, b, c\}$$

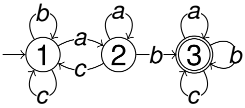


Core of Myhill equivalence:
State categorizes prefix.
How do words act on these prefixes?

Example: $f_a = (1 \mapsto 2, 2 \mapsto 2, 3 \mapsto 3)$

Semigroups: Equivalence classes

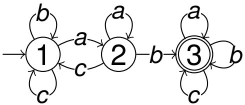
$$A = \{a, b, c\}$$



$$f_a = (1 \mapsto 2, 2 \mapsto 2, 3 \mapsto 3)$$

Semigroups: Equivalence classes

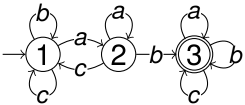
$$A = \{a, b, c\}$$



$$f_a = (1 \mapsto 2, 2 \mapsto 2, 3 \mapsto 3)$$
$$f_b = (1 \mapsto 1, 2 \mapsto 3, 3 \mapsto 3)$$

Semigroups: Equivalence classes

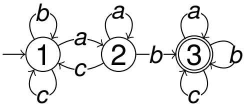
$$A = \{a, b, c\}$$



$$f_a = (1 \mapsto 2, 2 \mapsto 2, 3 \mapsto 3)$$
$$f_b = (1 \mapsto 1, 2 \mapsto 3, 3 \mapsto 3)$$
$$f_c = (1 \mapsto 1, 2 \mapsto 1, 3 \mapsto 3)$$

Semigroups: Equivalence classes

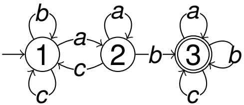
$$A = \{a, b, c\}$$



$$\begin{aligned}
f_a &= (1 \mapsto 2, 2 \mapsto 2, 3 \mapsto 3) \\
f_b &= (1 \mapsto 1, 2 \mapsto 3, 3 \mapsto 3) \\
f_c &= (1 \mapsto 1, 2 \mapsto 1, 3 \mapsto 3) \\
f_{ab} &= (1 \mapsto 3, 2 \mapsto 3, 3 \mapsto 3) \\
f_{ba} &= (1 \mapsto 2, 2 \mapsto 3, 3 \mapsto 3)
\end{aligned}$$

Semigroups: Equivalence classes

$$A = \{a, b, c\}$$

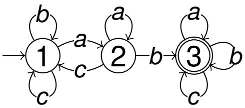


$$\begin{aligned}
f_a &= (1 \mapsto 2, 2 \mapsto 2, 3 \mapsto 3) \\
f_b &= (1 \mapsto 1, 2 \mapsto 3, 3 \mapsto 3) \\
f_c &= (1 \mapsto 1, 2 \mapsto 1, 3 \mapsto 3) \\
f_{ab} &= (1 \mapsto 3, 2 \mapsto 3, 3 \mapsto 3) \\
f_{ba} &= (1 \mapsto 2, 2 \mapsto 3, 3 \mapsto 3)
\end{aligned}$$

others: $f_{aa} = f_{ca} = f_a, \quad f_{ac} = f_{cb} = f_{cc} = f_c, \quad f_{bb} = f_{bc} = f_b$

Semigroups: Equivalence classes

$A = \{a, b, c\}$

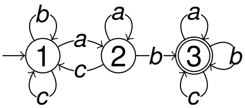


$$f_a = (1 \mapsto 2, 2 \mapsto 2, 3 \mapsto 3)$$
$$f_b = (1 \mapsto 1, 2 \mapsto 3, 3 \mapsto 3)$$
$$f_c = (1 \mapsto 1, 2 \mapsto 1, 3 \mapsto 3)$$
$$f_{ab} = (1 \mapsto 3, 2 \mapsto 3, 3 \mapsto 3)$$
$$f_{ba} = (1 \mapsto 2, 2 \mapsto 3, 3 \mapsto 3)$$

	a	*b*	*c*	*ab*	*ba*
a		*ab*			
b	*ba*				
c					
ab					
ba					

Semigroups: Equivalence classes

$A = \{a, b, c\}$

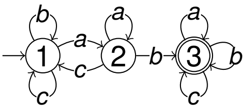


$$f_a = (1 \mapsto 2, 2 \mapsto 2, 3 \mapsto 3)$$
$$f_b = (1 \mapsto 1, 2 \mapsto 3, 3 \mapsto 3)$$
$$f_c = (1 \mapsto 1, 2 \mapsto 1, 3 \mapsto 3)$$
$$f_{ab} = (1 \mapsto 3, 2 \mapsto 3, 3 \mapsto 3)$$
$$f_{ba} = (1 \mapsto 2, 2 \mapsto 3, 3 \mapsto 3)$$

	a	b	c	ab	ba
a	a	ab	c		
b	ba	b	b		
c	a	c	c		
ab					
ba					

Semigroups: Equivalence classes

$A = \{a, b, c\}$



$$f_a = (1 \mapsto 2, 2 \mapsto 2, 3 \mapsto 3)$$
$$f_b = (1 \mapsto 1, 2 \mapsto 3, 3 \mapsto 3)$$
$$f_c = (1 \mapsto 1, 2 \mapsto 1, 3 \mapsto 3)$$
$$f_{ab} = (1 \mapsto 3, 2 \mapsto 3, 3 \mapsto 3)$$
$$f_{ba} = (1 \mapsto 2, 2 \mapsto 3, 3 \mapsto 3)$$

	a	*b*	*c*	*ab*	*ba*
a	*a*	*ab*	*c*	*ab*	*ab*
b	*ba*	*b*	*b*	*ab*	*ba*
c	*a*	*c*	*c*	*ab*	*a*
ab	*ab*	*ab*	*ab*	*ab*	*ab*
ba	*ba*	*ab*	*b*	*ab*	*ab*

Bridging logic and algebra: Definiteness

> “*We shall first restrict ourselves to events which refer to a fixed period of time, consisting of the* χ ($\geqslant 1$) *moments* $p - \chi + 1, \ldots, p$ *ending with the present.*” *(Kleene, 1951)*

k-**DEFINITE** languages:
last *k*-many symbols determine acceptability

Bridging logic and algebra: 1-definiteness

If u and v have the same last letter,
then xuy and xvy have the same last letter.

If the last letter determines acceptability:

$$u \sim v$$

Bridging logic and algebra: 1-definiteness

If u and v have the same last letter,
then xuy and xvy have the same last letter.

If the last letter determines acceptability:
$u \sim v$

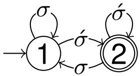
Semigroup elements are (equivalences of) nonempty words

So $sx \sim x$ for all s and x

Bridging logic and algebra: 1-definiteness

A linguistic pattern: **STRESS-FINAL**

$$A = \{\sigma, \acute{\sigma}\}$$



	σ	$\acute{\sigma}$
σ	σ	$\acute{\sigma}$
$\acute{\sigma}$	σ	$\acute{\sigma}$

$sx \sim x$ for all s and x

Bridging logic and algebra: 2-definiteness

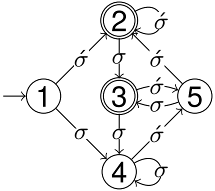
If last two letters determine acceptability
and both u and v have the same last two letters,
$u \sim v$

$sx_1x_2 \sim x_1x_2$ for all s, x_1 and x_2

Bridging logic and algebra: 2-definiteness

A linguistic pattern: **STRESS-PENULT**

$A = \{\sigma, \acute{\sigma}\}$



	σ	$\acute{\sigma}$	$\sigma\sigma$	$\sigma\acute{\sigma}$	$\acute{\sigma}\sigma$	$\acute{\sigma}\acute{\sigma}$
σ	σ	$\acute{\sigma}$	$\sigma\sigma$	$\sigma\acute{\sigma}$	$\acute{\sigma}\sigma$	$\acute{\sigma}\acute{\sigma}$
$\acute{\sigma}$	σ	$\acute{\sigma}$	$\sigma\sigma$	$\sigma\acute{\sigma}$	$\acute{\sigma}\sigma$	$\acute{\sigma}\acute{\sigma}$
$\vdots$			$\ddots$			

$sx_1x_2 \sim x_1x_2$ for all s and x

Bridging logic and algebra: Definiteness

k-definite: $sx_1 \ldots x_k \sim x_1 \ldots x_k$

Bridging logic and algebra: Definiteness

k-definite: $sx_1 \ldots x_k \sim x_1 \ldots x_k$

But what if we don't know k?
Do we have to try all of them?

Bridging logic and algebra: Definiteness

If there is some k: $sx_1 \ldots x_k \sim x_1 \ldots x_k$

Specifically for $s = x_1 \ldots x_k$:
$(x_1 \ldots x_k)(x_1 \ldots x_k) \sim x_1 \ldots x_k$

And if $x \sim xx$, then $x \sim xx \sim \cdots \sim x^k$

Bridging logic and algebra: Definiteness

Consider: $x, xx, \ldots$

Bridging logic and algebra: Definiteness

Consider: $x, xx, \ldots, x^n, \ldots, x^{n+m} = x^n$

Bridging logic and algebra: Definiteness

Consider: $x, xx, \ldots, x^n, \ldots, x^{n+m} = x^n$

$$x^{mn} = x^{mn}x^{mn}$$

$$x^{\omega} = \lim_{i\to\infty} x^{i!} = x^{mn}$$

Bridging logic and algebra: Definiteness

Consider: $x, xx, \ldots, x^n, \ldots, x^{n+m} = x^n$

$$x^{mn} = x^{mn}x^{mn}$$

$$x^\omega = \lim_{i\to\infty} x^{i!} = x^{mn}$$

Definite: $sx^\omega \sim x^\omega$ for all s and x

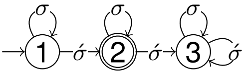
(see, among other sources, Straubing, 1985)

Ignoring letters: Culminativity

words do not contain $\acute{\sigma} \ldots \acute{\sigma}$

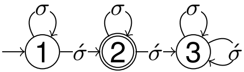
Ignoring letters: Culminativity

words do not contain $\acute{\sigma} \ldots \acute{\sigma}$



Ignoring letters: Culminativity

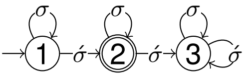
words do not contain $\acute{\sigma} \ldots \acute{\sigma}$



	σ	$\acute{\sigma}$	$\acute{\sigma}\acute{\sigma}$
σ	σ	$\acute{\sigma}$	$\acute{\sigma}\acute{\sigma}$
$\acute{\sigma}$	$\acute{\sigma}$	$\acute{\sigma}\acute{\sigma}$	$\acute{\sigma}\acute{\sigma}$
$\acute{\sigma}\acute{\sigma}$	$\acute{\sigma}\acute{\sigma}$	$\acute{\sigma}\acute{\sigma}$	$\acute{\sigma}\acute{\sigma}$

Ignoring letters: Culminativity

words do not contain $\acute{\sigma} \ldots \acute{\sigma}$



	σ	$\acute{\sigma}$	$\acute{\sigma}\acute{\sigma}$
σ	σ	$\acute{\sigma}$	$\acute{\sigma}\acute{\sigma}$
$\acute{\sigma}$	$\acute{\sigma}$	$\acute{\sigma}\acute{\sigma}$	$\acute{\sigma}\acute{\sigma}$
$\acute{\sigma}\acute{\sigma}$	$\acute{\sigma}\acute{\sigma}$	$\acute{\sigma}\acute{\sigma}$	$\acute{\sigma}\acute{\sigma}$

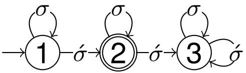
$\sigma^{\omega} = \sigma$ but $\acute{\sigma}\sigma^{\omega} \not\approx \sigma^{\omega}$

Ignoring letters: Culminativity

Tier projection:
delete letters you don't care about

Ignoring letters: Culminativity

Tier projection:
delete letters you don't care about



Ignoring letters: Culminativity

TIER PROJECTION:
delete letters you don't care about

Ignoring letters: Culminativity

Tier projection:
delete letters you don't care about

	$\acute{\sigma}$	$\acute{\sigma}\acute{\sigma}$
$\acute{\sigma}$	$\acute{\sigma}\acute{\sigma}$	$\acute{\sigma}\acute{\sigma}$
$\acute{\sigma}\acute{\sigma}$	$\acute{\sigma}\acute{\sigma}$	$\acute{\sigma}\acute{\sigma}$

Ignoring letters: Culminativity

TIER PROJECTION:
delete letters you don't care about

	$\acute{\sigma}$	$\acute{\sigma}\acute{\sigma}$
$\acute{\sigma}$	$\acute{\sigma}\acute{\sigma}$	$\acute{\sigma}\acute{\sigma}$
$\acute{\sigma}\acute{\sigma}$	$\acute{\sigma}\acute{\sigma}$	$\acute{\sigma}\acute{\sigma}$

$$sx^{\omega} \sim x^{\omega}$$

Definite after projection

"TIER" DEFINITE

Igoring letters: Multiple tiers

Culminativity: ignore σ
Stress-final: nothing can be ignored

Words that satisfy both: **MULTITIER DEFINITE**

Igoring letters: Multiple tiers

Culminativity: ignore σ
Stress-final: nothing can be ignored

Words that satisfy both: **MULTITIER DEFINITE**

Multitier definite: $s(xsy)^{\omega} \sim (xsy)^{\omega}$

Related classes

Multitier definite: attend to (projected) suffixes

Multitier reverse definite: attend to (projected) prefixes

Multitier generalized definite: attend to both

and many more

Outcomes

- StressTyp2 stress pattern database (Goedemans *et al.*, 2015)
  - 107 distinct patterns
  - All but two definable by (projected) factors
  - The remaining two are contested
    and would be local with slight modification
- Harmony and tone: need further catalogue,
  but equally simple.

Extensions

- Functions (processes)
  - Transducers are handled the same way
  - Input strictly local (Chandlee, 2014) are
    - Definite (Lambert and Heinz, 2023)
    - Quantifier-free logic on strings with adjacency (Chandlee and Lindell, to appear)
  - Attested nonlocal processes are also simple (Lambert, 2022)
- Syntactic structures
  - Under Myhill's congruence, regular $\leftrightarrow$ finite (Rabin and Scott, 1959)
    study the infinite semigroups?
  - Another congruence:
    operator-precedence language $\leftrightarrow$ finite (Henziger *et al.*, 2023)
    study these finite semigroups?

Thank You

Questions?

software

slides

```
$ cabal install language-toolkit
```

Jane Chandlee (2014), Strictly Local Phonological Processes, Ph.D. thesis, University of Delaware, URL https://chandlee.sites.haverford.edu/wp-content/uploads/2015/05/Chandlee_dissertation_2014.pdf.

Jane Chandlee and Steven Lindell (to appear), Logical Perspectives on Strictly Local Transformations.

Noam Chomsky (1959), On Certain Formal Properties of Grammars, Information and Control, 2(2):137–167, doi:10.1016/S0019-9958(59)90362-6.

R. W. N. Goedemans, Jeffrey Heinz, and Harry van der Hulst (2015), StressTyp2, URL http://st2.ullet.net/.

Thomas A. Henziger, Pavol Kebis, Nicolas Mazzocchi, and N. Ege Saraç (2023), Regular Methods for Operator-Precedence Languages, in Proceedings of the 50th International Colloquium on Automata, Languages, and Programming (ICALP 2023), volume 261 of Leibniz International Proceedings in Informatics (LIPIcs), pp. 129:1–20, doi:10.4230/LIPIcs.ICALP.2023.129.

Stephen Cole Kleene (1951), Representation of Events in Nerve Nets and Finite Automata, Technical Report RM-704, U.S. Air Force Project RAND.

Dakotah Lambert (2022), Unifying Classification Schemes for Languages and Processes with Attention to Locality and Relativizations Thereof, Ph.D. thesis, Stony Brook University.

Dakotah Lambert and Jeffrey Heinz (2023), An Algebraic Characterization of Total Input Strictly Local Functions, in Proceedings of the Society for Computation in Linguistics, volume 6, pp. 25–34, Amherst, Massachusetts, doi:10.7275/Q54B-MG07.

Michael Oser Rabin and Dana Scott (1959), Finite Automata and Their Decision Problems, IBM Journal of Research and Development, 3(2):114–125, doi:10.1147/rd.32.0114.

Len Sassaman, Meredith L. Patterson, Sergey Bratus, and Michael E. Locasto (2013), Security Applications of Formal Language Theory, IEEE Systems Journal, 7(3):489–500, doi:10.1109/JSYST.2012.2222000.

Howard Straubing (1985), Finite Semigroup Varieties of the Form $V * D$, Journal of Pure and Applied Algebra, 36:53–94, doi:10.1016/0022-4049(85)90062-3.