The Mathematical Theory of Formal Languages

Matilde Marcolli and Doris Tsao

Ma191b Winter 2017
Geometry of Neuroscience

References for this lecture:

1. Ian Chiswell, *A course in formal languages, automata and groups*, Springer, 2009
2. György Révész, *Introduction to formal languages*, McGraw-Hill, 1983
3. Noam Chomsky, *Three models for the description of language*, IRE Transactions on Information Theory, (1956) N.2, 113–124.
4. Noam Chomsky, *On certain formal properties of grammars*, Information and Control, Vol.2 (1959) N.2, 137–167
5. A.V. Anisimov, *The group languages*, Kibernetika (Kiev) 1971, no. 4, 18–24
6. D.E. Muller, P.E. Schupp, *Groups, the theory of ends, and context-free languages*, J. Comput. System Sci. 26 (1983), no. 3, 295–310

A very general abstract setting to describe languages (natural or artificial: human languages, codes, programming languages, ...)

Alphabet: a (finite) set $\mathfrak{A}$; elements are *letters* or *symbols*

Words (or strings): $\mathfrak{A}^m$ = set of all sequences $a_1 \ldots a_m$ of length m of letters in $\mathfrak{A}$

Empty word: $\mathfrak{A}^0 = \{\epsilon\}$ (an additional symbol)

$$\mathfrak{A}^+ = \cup_{m\geq 1}\mathfrak{A}^m, \qquad \mathfrak{A}^\star = \cup_{m\geq 0}\mathfrak{A}^m$$

concatenation: $\alpha = a_1 \ldots a_m \in \mathfrak{A}^m$, $\beta = b_1 \ldots b_k \in \mathfrak{A}^k$

$$\alpha\beta = a_1 \ldots a_m b_1 \ldots b_k \in \mathfrak{A}^{m+k}$$

Length $\ell(\alpha) = m$ for $\alpha \in \mathfrak{A}^m$
Language: a subset of $\mathfrak{A}^\star$
Question: how is the subset constructed?

Rewriting system on $\mathfrak{A}$: a subset $\mathcal{R}$ of $\mathfrak{A}^\star \times \mathfrak{A}^\star$

$(\alpha, \beta) \in \mathcal{R}$ means that for any $u, v \in \mathfrak{A}^\star$ the word $u\alpha v$ *rewrites* to $u\beta v$

Notation: write $\alpha \to_{\mathcal{R}} \beta$ for $(\alpha, \beta) \in \mathcal{R}$

$\mathcal{R}$-derivation: for $u, v \in \mathfrak{A}^\star$ write $u \stackrel{\bullet}{\to}_{\mathcal{R}} v$ if $\exists$ sequence $u = u_1, \dots, u_n = v$ of elements in $\mathfrak{A}^\star$ such that $u_i \to_{\mathcal{R}} u_{i+1}$

Grammar: a quadruple $\mathcal{G} = (V_N, V_T, P, S)$

- V_N and V_T disjoint finite sets: *non-terminal* and *terminal* symbols
- $S \in V_N$ *start symbol*
- P finite rewriting system on $V_N \cup V_T$

$P =$ *production rules*

Language produced by a grammar $\mathcal{G}$:

$$\mathcal{L}_{\mathcal{G}} = \{ w \in V_T^\star \,|\, S \stackrel{\bullet}{\to}_P w \}$$

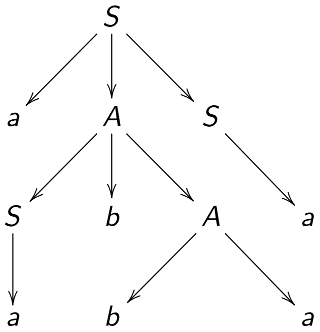
language with alphabet V_T

Production rules can be seen as *parsing trees*

Example: Grammar: $\mathcal{G} = \{\{S, A\}, \{a, b\}, P, S\}$ with productions P

$$S \to aAS, \quad S \to a, \quad A \to SbA, \quad A \to SS, \quad A \to ba$$

- this is a possible parse tree for the string *aabbaa* in $\mathcal{L}_{\mathcal{G}}$



Context free and context sensitive production rules

- context free: $A \to \alpha$ with $A \in V_N$ and $\alpha \in (V_N \cup V_T)^\star$
- context sensitive: $\beta A \gamma \to \beta \alpha \gamma$ with $A \in V_N$
$\alpha, \beta, \gamma \in (V_N \cup V_T)^\star$ and $\alpha \neq \epsilon$

context free is context sensitive with $\beta = \gamma = \epsilon$

"context free" languages: a first attempt (Chomsky, 1956) to model natural languages; not appropriate, but good for some programming languages (e.g. Fortran, Algol, HTML)

The Chomsky hierarchy

Types:

- Type 0: just a grammar $\mathcal{G}$ as defined above (*unrestricted grammars*)
- Type 1: *context-sensitive grammars*
- Type 2: *context-free grammars*
- Type 3: *regular grammars*, where all productions $A \to aB$ or $A \to a$ with $A, B \in V_N$ and $a \in V_T$

(right/left-regular if aB or Ba in r.h.s. of production rules)

Language of type n if produced by a grammar of type n

Examples

• Type 3 (regular): $\mathcal{G} = (\{S, A\}, \{0, 1\}, P, S)$ with productions P given by

$$S \to 0S, \quad S \to A, \quad A \to 1A, \quad A \to 1$$

then $\mathcal{L}_{\mathcal{G}} = \{0^m 1^n \mid m \geq 0, n \geq 1\}$

• Type 2 (context-free): $\mathcal{G} = (\{S\}, \{0, 1\}, P, S)$ with productions P given by

$$S \to 0S1, \quad S \to 01$$

then $\mathcal{L}_{\mathcal{G}} = \{0^n 1^n \mid n \geq 1\}$

• Type 1 (context-sensitive): $\mathcal{G} = (\{S, B, C\}\{a, b, c\}, P, S)$ with productions P

$$S \to aSBC, \quad S \to aBC, \quad CB \to BC,$$

$$aB \to ab, \quad bB \to bb, \quad bC \to bc, \quad cC \to cc$$

the $\mathcal{L}_{\mathcal{G}} = \{a^n b^n c^n \,|\, n \geq 1\}$

Main Idea: a generative grammar $\mathcal{G}$ determines *what kinds of recursive structures* are possible in the language $\mathcal{L}_{\mathcal{G}}$

Why is it useful to organize formal languages in this way?

Types and Machine Recognition

Recognized by:

- Type 0: Turing machine
- Type 1: linear bounded automaton
- Type 2: non-deterministic pushdown stack automaton
- Type 3: finite state automaton

What are these things?

Finite state automaton (FSA)

$M = (Q, F, \mathfrak{A}, \tau, q_0)$

- Q finite set: set of possible states
- F subset of Q: the final states
- $\mathfrak{A}$ finite set: alphabet
- $\tau \subset Q \times \mathfrak{A} \times Q$ set of transitions
- $q_0 \in Q$ initial state

computation in M: sequence $q_0 a_1 q_1 a_2 q_2 \dots a_n q_n$ where $q_{i-1} a_i q_i \in \tau$ for $1 \leq 1 \leq n$

- label of the computation: $a_1 \dots a_n$
- successful computation: $q_n \in F$
- M accepts a string $a_1 \dots a_n$ if there is a successful computation in M labeled by $a_1 \dots a_n$

Language recognized by M:

$$\mathcal{L}_M = \{ w \in \mathfrak{A}^\star \,|\, w \text{ accepted by } M \}$$

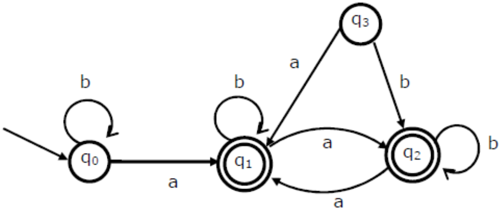
Graphical description of FSA

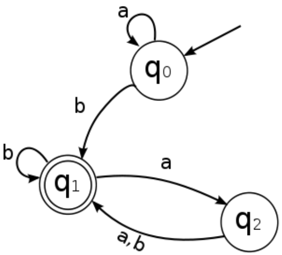
Transition diagram: oriented finite labelled graph Γ with vertices $V(\Gamma) = Q$ set of states and $E(\Gamma) = \tau$, with $e_{q,a,q'}$ an edge from v_q to $v_{q'}$ with label $a \in \mathfrak{A}$; label vertex q_0 with $-$ and all final states vertices with $+$

- computations in $M \Leftrightarrow$ paths in Γ starting at v_{q_0}

- an oriented labelled finite graph with at most one edge with a given label between given vertices, and only one vertex labelled $-$ is the transition diagram of some FDA

Examples





Examples of finite state automata with marked final states deterministic FSA

for all $q \in Q$ and $a \in \mathfrak{A}$, there is a unique $q' \in Q$ with $(q, a, q') \in \tau$

$\Rightarrow$ function $\delta : Q \times \mathfrak{A} \to Q$ with $\delta(q, a) = q'$, *transition function*

determines $\delta : Q \times \mathfrak{A}^\star \to Q$ by $\delta(q, \epsilon) = q$ and
$\delta(q, wa) = \delta(\delta(q, w), a)$ for all $w \in \mathfrak{A}^\star$ and $a \in \mathfrak{A}$

if $q_0 a_1 q_1 \ldots a_n q_n$ computation in M then $q_n = \delta(q_0, a_1 \ldots a_n)$

non-deterministic: multivalued transition functions also allowed

Languages recognized by (non-deterministic) FSA are Type 3

- for $\mathcal{G} = (V_N, V_T, P, S)$ type 3 grammar construct an FSA

$$M = (V_N \cup \{X\}, F, V_T, \tau, S)$$

with X a new letter, $F = \{S, X\}$ if $S \to_P \epsilon$, $F = \{X\}$ if not;

$$\tau = \{(B, a, C) \,|\, B \to_P aC\} \cup \{(B, a, X) \,|\, B \to_P a,\ a \neq \epsilon\}$$

then $\mathcal{L}_{\mathcal{G}} = \mathcal{L}_M$

- if M is a FSA take $\mathcal{G} = (Q, \mathfrak{A}, P, q_0)$ with P given by

$$P = \{B \to aC \,|\, (B, a, C) \in \tau\} \cup \{B \to a \,|\, (B, a, C) \in \tau, C \in F\}$$

then $\mathcal{L}_M = \mathcal{L}_{\mathcal{G}}$

pushdown stack automaton (PDA)

$M = (Q, F, \mathfrak{A}, \Gamma, \tau, q_0, z_0)$

- Q finite set of possible states
- F subset of Q: the final states
- $\mathfrak{A}$ finite set: alphabet
- Γ finite set: *stack alphabet*
- $\tau \subset Q \times (\mathfrak{A} \cup \{\epsilon\}) \times \Gamma \times Q \times \Gamma^\star$ finite subset: set of transitions
- $q_0 \in Q$ initial state
- $z_0 \in \Gamma$ start symbol

• it is a FSA $(Q, F, \mathfrak{A}, \tau, q_0)$ together with a stack $\Gamma^\star$

• the transitions are determined by the first symbol in the stack, the current state, and a letter in $\mathfrak{A} \cup \{\epsilon\}$

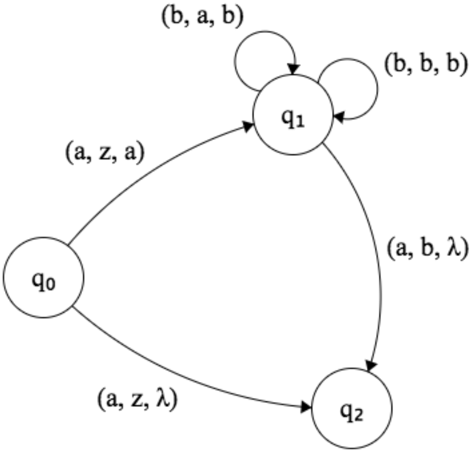
• the transition adds a new (finite) sequence of symbols at the beginning of the stack $\Gamma^\star$

• a configuration of M is an element of $Q \times \mathfrak{A}^\star \times \Gamma^\star$

• given $(q, a, z, q', \alpha) \in \tau \subset Q \times (\mathfrak{A} \cup \{\epsilon\}) \times \Gamma \times Q \times \Gamma^\star$ the corresponding transition is from a configuration $(q, aw, z\beta)$ to a configuration $(q', w, \alpha\beta)$

• computation in M: a chain of transitions $c \to c'$ between configurations $c = c_1, \ldots, c_n = c'$ where each $c_i \to c_{i+1}$ a transition as above

Example



a transition labelled (a, b, c) between vertex q_i and q_j means read letter a on string, read letter b on top of memory stack, remove b and place c at the top of the stack: move from configuration $(q_i, aw, b\alpha)$ to configuration $(q_j, w, c\alpha)$

- computation stops when reach final state or empty stack
- PDA M accepts $w \in \mathfrak{A}^\star$ by final state if $\exists \gamma \in \Gamma^\star$ and $q \in F$ such that $(q_0, w, z_0) \to (q, \epsilon, \gamma)$ is a computation in M
- Language recognized by M *by final state*

$$\mathcal{L}_M = \{ w \in \mathfrak{A}^\star \,|\, w \text{ accepted by } M \text{ by final state } \}$$

- $w \in \mathfrak{A}^\star$ accepted by M *by empty stack*: if $(q_0, w, z_0) \to (q, \epsilon, \epsilon)$ is a computation on M with $q \in Q$
- Language recognized by M *by empty stack*

$$\mathcal{N}_M = \{ w \in \mathfrak{A}^\star \,|\, w \text{ accepted by } M \text{ by empty stack } \}$$

deterministic PDA

1. at most one transition $(q, a, z, q', \alpha) \in \tau$ with given (q, a, z) source
2. if there is a transition from (q, ϵ, z) then there is no transition from (q, a, z) with $a \neq \epsilon$

first condition as before; second condition avoids choice between a next move that does not read the tape and one that does

Fact: recognition by final state and by empty stack equivalent for non-deterministic PDA

$$\mathcal{L} = \mathcal{L}_M \Leftrightarrow \mathcal{L} = \mathcal{N}_{M'}$$

not equivalent for deterministic: in deterministic case languages $\mathcal{L} = \mathcal{N}_M$ have additional property:
prefix-free: if $w \in \mathcal{L}$ then no prefix of w is in $\mathcal{L}$

Languages recognized by (non-deterministic) PDA are Type 2 (context-free)

- If $\mathcal{L}$ is context free then $\mathcal{L} = \mathcal{N}_M$ for some PDA M

$\mathcal{L} = \mathcal{L}_{\mathcal{G}}$ with $\mathcal{G} = (V_N, V_T, P, S)$ context-free, take $M = (\{q\}, \emptyset, V_T, V_N, \tau, q, S)$ with τ given by the (q, a, A, q, γ) for productions $A \to a\gamma$ in P

then for $\alpha \in V_N^\star$ and $w \in V_T^\star$ have

$$S \stackrel{\bullet}{\to}_P w\alpha \quad \Leftrightarrow \quad (q, w, S) \to_M (q, \epsilon, \alpha)$$

if also $\epsilon \in \mathcal{L}$ add new state q' and new transition $(q, \epsilon, Sq', \epsilon)$, where S start symbol of a PDA that recognizes $\mathcal{L} \smallsetminus \{\epsilon\}$

- if $\mathcal{L} = \mathcal{N}_M$ for PDA M then $\mathcal{L} = \mathcal{L}_{\mathcal{G}}$ with $\mathcal{G}$ context-free

for $M = (Q, F, \mathfrak{A}, \Gamma, \tau, q_0, z_0)$ define $\mathcal{G} = (V_N, \mathfrak{A}, P, S)$ where

$$V_N = \{(q, z, p) \,|\, q, p \in Q, z \in \Gamma\} \cup \{S\}$$

with production rules P given by

1. $S \to (q_0, z_0, q)$ for all $q \in Q$
2. $(q, z, p) \to a(q_1, y_1, q_2)(q_2, y_2, q_3) \cdots (q_m, y_m, q_{m+1})$ with $q_1 = q$, $q_{m+1} = p$ and $(q, a, z, q_1, y_1 \ldots y_m)$ transition of M

$$(q, w, z) \to_M (p, \epsilon, \epsilon) \quad \Leftrightarrow \quad (q, z, p) \overset{\bullet}{\to}_P w$$

Similar arguments show Type 0 = recognized by Turing machine; Type 1 (context sensitive) = recognized by "linear bounded automata" (Turing machines but only part of tape can be used)

Representing natural languages?

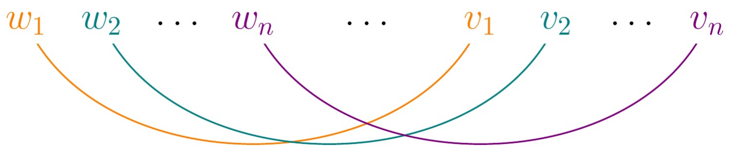
• Question: How good are context-free grammars at representing natural languages?

- Originally conjectured to be the right class of formal languages to contain natural languages
- Not always good, but often good (better than earlier criticism indicated)
- Some explicit examples not context-free (cross-serial subordinate clause in Swiss-German)

1. G.K. Pullum, G. Gazdar *Natural languages and context-free languages*, Linguistics and Philosophy, Vol.4 (1982) N.4, 471–504
2. S. Shieber, *Evidence against the context-freeness of natural language*, Linguistics and Philosophy, Vol.8 (1985) N.3, 333–343

Are natural languages context-free?

• Try to show they are not by finding cross-serial dependencies of arbitrarily large size



• Example: the language $\mathcal{L} = \{xx^R \,|\, x \in \{a, b\}^*\}$ has cross serial dependencies of arbitrary length (the i-th and $(n+i)$-th term have to be the same (x^R = reversal of x)

• if cross serial dependencies of arbitrary length not context-free

The Swiss German Example

Swiss German cross-serial order in dependent clauses

$$wa^n b^m x c^n d^m y$$

Jan säit das mer (d'chind)^n (em Hans)^m es huus haend wele (laa)^n (häfte)^m aastrüche
non-context-free language

• S. Shieber, *Evidence against the context-freeness of natural language*, Linguistics and Philosophy, Vol.8 (1985) N.3, 333–343

- Context-free class too small
- Context-sensitive class too large
- Intermediate candidates:
  1. Tree Adjoining Grammars
  2. Merge Grammars

Other Problem: Clearly there are many more formal languages that do not correspond to natural (human) languages (even within the appropriate class that contains natural languages)

Example: Programming Languages: Fortran is context-free; C is context-sensitite; C^{++} is Type 0, ...

Examples: Formal Languages constructed from finitely presented discrete groups

Formal Language of a finitely presented group

- Group G, with presentation $G = \langle X \,|\, R \rangle$ (finitely presented)
  - X (finite) set of generators $x_1, \ldots, x_N$
  - R (finite) set of relations: $r \in R$ words in the generators and their inverses
- for $G = \langle X \,|\, R \rangle$ call $\hat{X} = \{x, x^{-1} \,|\, x \in X\}$ symmetric set of generators
- Language associated to a finitely presented group $G = \langle X \,|\, R \rangle$

$$\mathcal{L}_G = \{ w \in \hat{X}^\star \,|\, w = 1 \in G \}$$

set of words in the generators representing trivial element of G

- Question: What kind of formal language is it?

• Algebraic properties of the group G correspond to properties of the formal language $\mathcal{L}_G$:

1. $\mathcal{L}_G$ is a regular language (Type 3) iff G is finite (Anisimov)
2. $\mathcal{L}_G$ is context-free (Type 2) iff G has a free subgroup of finite index (Muller–Schupp)

Example: Take $G = \mathrm{SL}_2(\mathbb{Z})$, infinite so $\mathcal{L}_G$ not regular; generators

$$S = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \text{ and } T = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$$

with relations S^2 and $(ST)^3$

$$\begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix} \text{ and } \begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix}$$

generate a free subgroup F_2 of index 12 in $\mathrm{SL}_2(\mathbb{Z})$ (of index 2 in $\Gamma(2)$ that has index 6 in $\mathrm{SL}_2(\mathbb{Z})$) so $\mathcal{L}_{\mathrm{SL}_2(\mathbb{Z})}$ is context-free

The "Boundaries of Babel" Problem

- Given a class of formal languages good enough to contain natural languages
- How to characterize the "region" within this class of formal languages that is populated by actual human (natural) languages?
- What is the *geometry* of the space of natural languages inside the space of formal languages?

• Andrea Moro, *The Boundaries of Babel. The Brain and the Enigma of Impossible Languages*, Second Edition, MIT Press, 2015

Want: a characterization and parameterization of the syntax of human languages

Broad Question: Is it possible to develop something like the mathematical theory of formal languages for Vision instead of Language?

- Best attempt so far: Pattern Theory
  - Ulf Grenander, *Elements of Pattern Theory*, Johns Hopkins University Press, 1996
  - Ulf Grenander, Michael I. Miller, *Pattern Theory: From Representation to Inference*, Oxford University Press, 2007
  - David Mumford, Agnès Desolneux, *Pattern Theory: The Stochastic Analysis of Real-World Signals*, CRC Press, 2010.