Formal Languages & Finite Automata

Reference

Collection of interconnected topics.

- IA Discrete Math
  - Regular Languages and finite automata
  - Pumping Lemma for Regular Languages
- IB Compiler Construction
  - CFG, PDA
- IB Formal Model of Language
  - Pumping Lemma for CFG
  - Chomsky hierarchy
- IB Computation Theory
  - TM
- II Quantum Computing
  - Quantum Automata

Notation

Kleene star $\star$

- the set of all strings that can be written as the concatenation of zero or more strings from A.
- finite set

Optional bracket $[]$

- or write it in case split

Definition

String $\omega \in T^\star$, over alphabet Σ

- of length $n \in \mathbb{N}$
- Empty string ϵ, the unique string of length 0

Language $L = \{\omega | \omega \in T^\star\}$

Grammar $G = \langle N, T, P, S \rangle$

- N is a finite set of Non-terminal/variables. (Q in Automata)
- T is a finite set of terminals/symbols. (Σ in Automata)
- Require that N and T disjoint, i.e. $N \cap T = \emptyset$
- P is Production rule, a relation defined

- Regular $P \in N \times T[N]$
- CFG $P \in N \times (T \cup N)^\star$
- CSG $P \in (N \cup T)^\star N (N \cup T)^\star \times (N \cup T)^\star (T \cup N)^\star (N \cup T)^\star$
- Recursive enumerable $P \in (N \cup T)^\star \times (N \cup T)^\star$

- $S \in N$ is the Start symbol

Relationship of Grammar and Language

Grammar	Language
Finite (Kleene star)	∞
Structured	Flat lists of w
many grammars	map to the same language

Finite Automata

$M_{Automata} = (Q, \Sigma, \delta, s, F)$

- Q is a finite set of states (N in Grammar)
- Σ is the finite set alphabet (T in Grammar)
  - Q and Σ disjoints, i.e. $\Sigma \cap Q = \emptyset$
- δ is the transition relation
  - corresponds to Production rule (Grammar)
- $s \in Q$ is the start state, also known as q_0
- $F \subset Q$ is the set of accepting states
- $L(M)$: the language accepted by the automata

[Non]Deterministic Finite Automata

$M_{NFA} = (Q, \Sigma, \delta, s, F)$

- $\delta \subset Q \times (\Sigma \cup \{\epsilon\}) \rightarrow Q$ transition relation
  - (q, a, q') means $q \xrightarrow{a} q'$

$M_{DFA} = (Q, \Sigma, \delta, s, F)$

- $\delta \subset Q \times \Sigma \rightarrow Q$ transition relation
  - (q, a, q') means $q \xrightarrow{a} q'$
- $L(M) = \{\omega \in \Sigma^* \mid \exists q \in Q, s \xRightarrow{\omega} q\}$

PushDown Automata

$M_{PDA} = (Q, \Sigma, \Gamma, \delta, s, Z, F)$

- $\delta \subset Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \rightarrow Q \times \Gamma^\star$ transition relation
  - $\delta(q, a, X) = (q', \beta)$
    - $q \xrightarrow{a} q'$ and replace top of stack X with β

- Γ is a finite set which is called the **stack** alphabet
- $Z \in \Gamma$ is the initial stack symbol
- $L(M) = \{\omega \in \Sigma^* \mid \exists q \in Q, ID = \langle s, \omega, Z\rangle \to^+ ID' = \langle q, \epsilon, \epsilon\rangle\}$
  - The initial content ω is said to be accepted by M if it eventually halts in a state from F
  - with Instantaneous Description (ID) above.

Turing Machine

$M_{TM} = (Q, \Sigma, \delta, s, F)$

- $\Sigma = \{\sqcup, \triangleright\} \cup \Sigma'$ is the finite set of **tape** alphabet symbols
  - $\sqcup$ the blank symbol
    - the only symbol allowed to occur on the tape infinitely often at any step during the computation
  - $\triangleright$ left endmarker
    - the left end marker $\triangleright$ is never overwritten and M always move Right.
  - Σ' the finite set of input symbols, to be checked
    - allow to appear in the initial tape contents.
- $\delta \subset Q \times \Sigma \to Q \times \Sigma \times \{L, R, S\}$ transition relation
  - $\delta(q, a) = (q', b, u)$
    - $q \xrightarrow{a} q'$ and overwrite the current symbol from a to b, update the next head movement.
  - $\delta(q, \triangleright) = (q', \triangleright, R)$
    - the left endmarker $\triangleright$ is never overwritten and header always moves Right.
- $L(M) = \{\omega \in \Sigma' \mid \exists q \in \{acc, rej\}, c_0 = \langle s, \triangleright, u\rangle \to^+ c' = \langle q, \omega, u'\rangle\}$
  - The initial content ω is said to be accepted by M if it eventually halts in a state from F
  - with configuration above.

Pumping Lemma

Proof For Regular Language L,

$\exists k \in \mathbb{Z}^+$, $\forall \omega \in L$ with $\mid \omega \mid \geq k$ (the number of states), can be written as a concatenation of three strings $\omega = u_1\mathbf{v}u_2$, satisfying

- $\mid \mathbf{v} \mid \geq 1$ $(v \neq \epsilon)$
- $\mid u_1\mathbf{v} \mid \leq k$

$\forall n \in \mathbb{N}$, $u_1\mathbf{v^n}u_2 \in L\square$.

Disproof L is not regular, (*negation* of the above)

$\forall k \geq 1$, $\exists \omega \in L$ with $\mid \omega \mid \geq k$, such that no matter how ω is split into three strings $\omega = u_1\mathbf{v}u_2$, satisfying

- $\mid \mathbf{v} \mid \geq 1$ ($v \neq \epsilon$)
- $\mid u_1\mathbf{v} \mid \leq k$

$\exists n \in \mathbb{N}$, $u_1\mathbf{v}^{\mathbf{n}}u_2 \notin L\square$.

[Need different cases on valid string split positions]

Proof For Language L of CFG,

$\exists k \in \mathbb{Z}^+$, $\forall \omega \in L$ with $\mid \omega \mid \geq k$ (the number of states), can be written as a concatenation of three strings $\omega = u_1\mathbf{v_1}u_2\mathbf{v_2}u_3$, satisfying

- $\mid \mathbf{v_1}\mathbf{v_2} \mid \geq 1$ ($v_1 \neq \epsilon$ and $v_2 \neq \epsilon$)
- $\mid \mathbf{v_1}u_2\mathbf{v_2} \mid \leq k$
- $\forall n \in \mathbb{N}$, $u_1\mathbf{v_1^n}u_2\mathbf{v_2^n}u_3 \in L$.

Disproof L is not CFG, (*negation* of the above)

$\forall k \in \mathbb{Z}^+$, $\exists \omega \in L$ with $\mid \omega \mid \geq k$, such that no matter how ω is split into strings $\omega = u_1\mathbf{v_1}u_2\mathbf{v_2}u_3$, satisfying

- $\mid \mathbf{v_1}\mathbf{v_2} \mid \geq 1$ ($v_1 \neq \epsilon$ and $v_2 \neq \epsilon$)
- $\mid \mathbf{v_1}u_2\mathbf{v_2} \mid \leq k$
- $\exists n \in \mathbb{N}$, $u_1\mathbf{v_1^n}u_2\mathbf{v_2^n}u_3 \notin L\square$.

[Need different cases on valid string split positions]

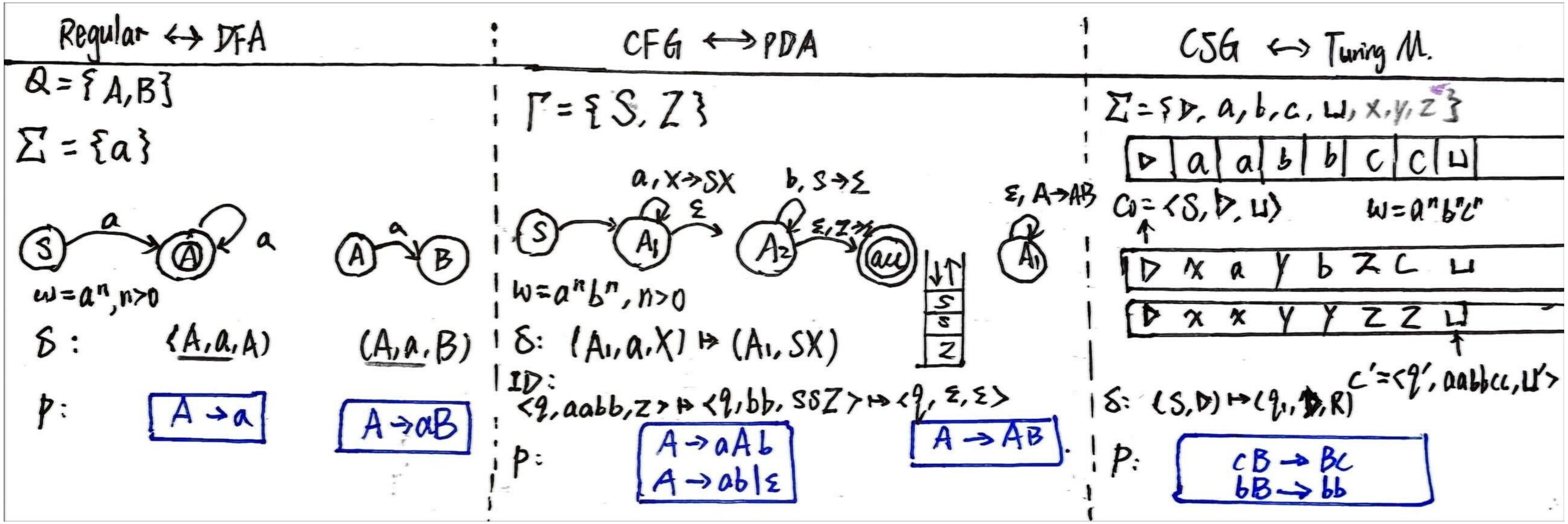
Chomsky hierarchy

	Gramma $\langle N, T, P, S \rangle$	Automata $(Q, \Sigma, \delta, s, F)$	Production rules P	Language e.g.	Usage
T-3	Regular	DFA	$A \to a$ $A \to aB$ (right)	$L = \{a^n \mid n > 0\}$	Lexer
T-2	Context-free	∞-stack PDA (Γ, Z)	$A \to \alpha$	$L = \{a^n b^n \mid n > 0\}$	General Parser
T-1	Context-sensitive	Linear-bounded Turing Machine	$\alpha A \beta \to \alpha \gamma \beta$	$L = \{a^n b^n c^n \mid n > 0\}$	Specific Parser
T-0	Recursively enumerable	Turning Machine	$\gamma \to \alpha$	$L = \{\omega\}$ TM recognizable	semi-decidable

The lower Automata/Grammar is strictly stronger than all the upper.

Notation:

- $A \in N$, denotes single Non-terminal
- $a \in T$, denotes single terminal
- $\alpha, \beta, \gamma \in (N \cup T)^\star$, denotes string of finite terminals and/or Non-terminals



(Adapted from wiki)