Chapter 3 Homework

3.1 (1)
3.2 (1b, 2bc)
3.4 (1b, 2)
3.5 (1, 2)
3.6 (1, 3)

Foundations of Computing

Due Next Wed

Exams 1 to 4

(no final)
(drop low exam)

25% (Attend + HW)
75% (Exams)

Foundations of Computing, Critchlow / Eck

chapters 3, 4 Languages, Strings, Grammars and Machines

3.1 Languages
4.1 - 4.3, 4.6 Grammars
3.2 Regular Expressions
3.4 - 3.6, 4.4 Automata

Ch5 Turing Machines and Computability

(1) Read ch's 3 and 4

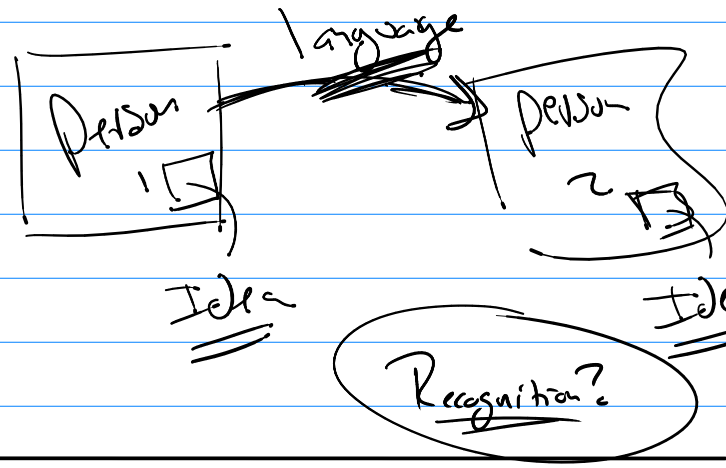
Languages and Grammars

Languages

1st **Natural** vs **Formal**

I ran today. — Idea #1

Is it hot? — Idea #2



Symbols:

- Σ = { set of symbols to use } (alphabet)
- String of symbols = word, sentence, idea
- string is a concatenation of symbols.
- empty string (no symbols) λ or ϵ (empty) string of no symbols

Language → sets of strings / sets of words / sets of sentences

English = { every possible string of ideas that english speakers recog. }

Natural!!

Symantic   syntax

Formal?

(ex) $\Sigma = \{0, 1\}$ length of string = # of symbols used.

length(ε) = 0

strings of length 1: 0, 1

2: 00, 01, 10, 11

⋮

Concatenation of symbols
= string
= idea

01, , cat

Language = { list the ideas }
(ideas → string)

ε	= idea 1
0	= idea 2
1	= idea 3
01	= ⋮
00	=
10	=
11	=
111	=

---

how many languages for a Σ?

[a] Language = subset of all possible strings

Kleene Closure, Σ^*

Σ^* = all possible strings of any length
0, 1, 2, 3, 4, ...

$\Sigma^* = \varepsilon \cup \Sigma \cup \Sigma\Sigma \cup \ldots$ = countably infinite

(Σ: 1 symbol; $\Sigma\Sigma$: 2 symbol)

Language is a subset of Σ^* ← countably infinite $\aleph_0$

counted by power set $P(S)$ = set of all subsets

$|P(S)| = 2^{|S|}$

$|\Sigma^*| = \aleph_0$ countable

all possible languages $|P(\Sigma^*)| = 2^{\aleph_0} = \aleph_1$

$= |\mathbb{R}| =$ uncountably infinite.

$|\mathbb{R}| = 2^{\aleph_0} = \aleph_1$ (countable)

Formal Languages → restrict why a string is in the language.

ex: english textbook, english dictionary

→ **Grammar**

list a set vs Set builder by induction

Σ - alphabet

Σ = Terminals ∪ Nonterminals

Terminals: fixed symbol ideas — cat = [cat]

Nonterminals: larger "variable" idea. → english sentence

Grammar = (Alphabet = Non-Term ∪ terminals, start symbol

Set of productions: tells how to construct (replace Non-term with term)) ex

$\Sigma = \{ S, a, b, A, B \}$ — S: Non, a, b: term, A, B: Non-term

$N = \{S, A, B\}$

$T = \{a, b\}$

Start = S

Productions:

1. $S \to AB$
2. $A \to a$
3. $A \to bB$
4. $B \to b$
5. $B \to aba$

Language of grammar = set of strings = $\{ S \overset{*}{\Rightarrow} x \}$

Derivation

$S \underset{1}{\Rightarrow} AB \underset{5}{\Rightarrow} Aaba$

$\underset{3}{\Rightarrow} bBaba$

$\underset{4}{\Rightarrow} bbaba$

$S \underset{*}{\Rightarrow} bbaba$

Phrase Structure Grammar / General Grammar

If we restrict productions we make different languages and create grammar types / language types.

Type 1 Regular Grammars / Regular Strings

1) Constructing the strings by induction

Basis: Σ is an alphabet. $a \in \Sigma$

ϕ, ϵ, $\{a\}$ are regular

Inductive Step s_1, s_2 are regular then

$s_1 | s_2$, $s_1 \cdot s_2$

$s_1 \cup s_2$, $s_1 s_2$, s_1^*

Languages = sets of strings so..

regular language → generated by regular expression

(2) Regular grammars   set of terminal, non-terminals, start symbol

Regular Productions: single nonterminal → single terminal
single nonterminal → single term | single non-term
or $S \to \varepsilon$

(ex) $S \to aA$, $S \to \lambda$
$A \to a$, $B \to b$
$A \to aB$, $B \to bA$

generates sets of regular strings

What ties these together?

Finite State Automatons

Next Lecture   State Machines