COMP3630 / COMP6363

week 4: **The Chomsky Hierarchy**

Not in the book, but relevant!

slides created by: Mostly by Pascal Bercher, some by Dirk Pattinson

convenor & lecturer: Pascal Bercher

The Australian National University

Semester 1, 2025

Content of this Chapter

- One new language class: Context-sensitive languages
- Classification of languages: The Chomsky Hierarchy

Languages So Far

So far, we covered the following language classes:

- Any languages:
  - By definition: Just any set set of strings over an alphabet.
  - Grammar: Didn't discuss yet! Does there exist any?
- Context-free languages:
  - By definition: If there exists a context-free grammar.
    (We didn't say it that clearly at the time. See updated slides!)
  - By theorems: If there is a PDA.
  - Grammar: See definition of context-free grammar.
- Regular languages:
  - By definition: If there exists a regular expression.
  - By theorems: If there is a DFA, NFA, or ϵ-NFA
  - Grammar: See tutorials! A syntactic restriction on CFGs.

Are there any other important languages?

Context-Sensitive Grammars

Introduction to Context-Sensitive Grammars

A grammar $G = (V, T, \mathcal{P}, S)$ is **context-sensitive** if every production rule

$$\alpha \to \beta$$

in $\mathcal{P}$ satisfies:

- $|\alpha| \leq |\beta|$,
- α contains at least one non-terminal (i.e., $\alpha \cap V \neq \emptyset$).

The exception is the rule $S \to \epsilon$ (if $\epsilon \in L(G)$), provided S does not occur on any right-hand side.

The point behind the length restriction is to make the CFLs decidable.
For any word, we know exactly when to stop applying rules and hence checking!

Example

Later, we will learn that $L = \{0^n 1^n 2^n \mid n \geq 0\}$ is not context-free.

Here is a context-sensitive grammar for it:

$$
\begin{array}{lll}
S \rightarrow 0\,S\,B\,C \mid 0\,1\,2 & 2B \rightarrow B2 & 0B \rightarrow 0\,1 \\
1B \rightarrow 1\,1 & 1C \rightarrow 1\,2 & 2C \rightarrow 2\,2
\end{array}
$$

E.g., this is a derivation for 001122:

$$
\begin{array}{cccl}
S & \Rightarrow_S & 0\,S\,B\,C & (\text{applying } S \rightarrow 0\,S\,B\,C) \\
 & \Rightarrow_S & 0\,(0\,1\,2)\,B\,C & (\text{applying } S \rightarrow 0\,1\,2) \\
 & = & 0\,0\,1\,2\,B\,C & \\
 & \Rightarrow_{2B} & 0\,0\,1\,B\,2\,C & (\text{applying } 2B \rightarrow B2) \\
 & \Rightarrow_{1B} & 0\,0\,1\,1\,2\,C & (\text{applying } 1B \rightarrow 1\,1) \\
 & \Rightarrow_{2C} & 0\,0\,1\,1\,2\,2 & (\text{applying } 2C \rightarrow 2\,2)
\end{array}
$$

The Chomsky Hierarchy

The Chomsky Hierarchy (by Noam Chomsky, a Linguist!)

Each grammar is of a type: (There are lots of intermediate types, too.)

Unrestricted: (type 0) no constraints, i.e., all productions $\alpha \to \beta$ (where $\alpha \cap V \neq \emptyset$)
(This does not mean that it can describe all possible languages!)

Context-sensitive: (type 1) the length of the left hand side of each production must not exceed the length of the right*, $|\alpha| \leq |\beta|$ and $\alpha \cap V \neq \emptyset$.

- There are other equivalent definitions which don't restrict the length
- *If $\epsilon \in L$ should be allowed, we are allowed $S \to \epsilon$, but then we don't allow S to occur on any right-hand side, just like in the CNF.

Context-free: (type 2) the left of each production must be a single non-terminal.
(Q. You can argue that not every type 2 grammar is if type 1, why?)

Regular: (type 3) As for type 2, but the right of each production is further constrained (see week 3 tutorials and later).

This also gives us a way to classify languages. (Next slide.)

Classification of Languages

Definition. A language is type n if it can be generated by a type n grammar.

Immediate Fact.

- Every language of type $n+1$ is also of type n.
- E.g., every context-free language (type 2) is also context-sensitive (type 1).

Establishing that a language is of type n

- give a grammar of type n that generates the language
  (or an automaton with equal expressive power)
- usually the easier task (and often fun! like designing an automaton/RegEx)

Disproving that a language is of type n

- must show that no type n-grammar generates the language
- usually a difficult problem. (E.g., by using the Pumping Lemma.)