MACM-300

Intro to Formal Languages and Automata

Anoop Sarkar

http://www.cs.sfu.ca/~anoop

The Chomsky Hierarchy

- **unrestricted** or **type-0** grammars, generate the *recursively enumerable* languages, automata equals *Turing machines*
- **context-sensitive** grammars, generate the *context-sensitive* languages, automata equals *Linear Bounded Automata*
- **context-free** grammars, generate the *context-free* languages, automata equals *Pushdown Automata*
- **regular** grammars, generate the *regular* languages, automata equals *Finite-State Automata*

The Chomsky Hierarchy: $G = (V, T, P, S)$ where, $\alpha, \beta, \gamma \in (N \cup T)^*$

- **unrestricted** or **type-0** grammars: $\alpha \rightarrow \beta$, such that $\alpha \neq \epsilon$
- **context-sensitive** grammars: $\alpha A \beta \rightarrow \alpha \gamma \beta$, such that $\gamma \neq \epsilon$
- **context-free** grammars: $A \rightarrow \gamma$
- **regular** grammars: $A \rightarrow a\ B$ or $A \rightarrow a$

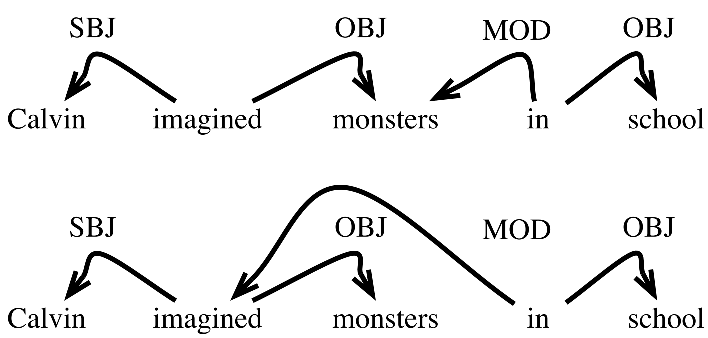
Regular grammars: **right-linear CFG**: $L(G) = \{a^* b^* \mid n \geq 0\}$

$$
\begin{array}{lcl}
A & \rightarrow & a\ A \\
A & \rightarrow & \epsilon \\
A & \rightarrow & b\ B \\
B & \rightarrow & b\ B \\
B & \rightarrow & \epsilon
\end{array}
$$

Context-free grammars: $L(G) = \{a^n b^n \mid n \geq 0\}$

$$
\begin{aligned}
S &\rightarrow a\,S\,b \\
S &\rightarrow \epsilon
\end{aligned}
$$

Dependency Grammar



Dependency Grammar: (Tesnière, 1959), (Panini, 500-400 BC)

1	Calvin	2	SBJ
2	imagined	–	TOP
3	monsters	2	OBJ
4	in	{2,3}	MOD
5	school	4	OBJ

- If the dependencies are *nested* then DGs are equivalent (formally) to CFGs

  1. TOP(imagined) → SBJ(Calvin) imagined OBJ(monsters) MOD(in)

  2. MOD(in) → in OBJ(school)

- However, each rule is lexicalized (has a terminal symbol)

Categorial Grammar: (Adjukiewicz, 1935)

Calvin	hates	mangoes
NP	(S\NP)/NP	NP
	S\NP	
	S	

- Also equivalent to CFGs

- Similar to DGs, each rule in CG is lexicalized

Context-sensitive grammars: $L(G) = \{a^n b^n \mid n \geq 1\}$

$$\begin{array}{rcl} S & \rightarrow & S\ B\ C \\ S & \rightarrow & a\ C \\ a\ B & \rightarrow & a\ a \\ C\ B & \rightarrow & B\ C \\ C & \rightarrow & b \end{array}$$

Context-sensitive grammars: $L(G) = \{a^n b^n \mid n \geq 1\}$

$$\begin{array}{r} S_1 \\ S_2\ B_1\ C_1 \\ S_3\ B_2\ C_2\ B_1\ C_1 \\ a_3\ C_3\ B_2\ C_2\ B_1\ C_1 \\ a_3\ B_2\ C_3\ C_2\ B_1\ C_1 \\ a_3\ a_2\ C_3\ C_2\ B_1\ C_1 \\ a_3\ a_2\ C_3\ B_1\ C_2\ C_1 \\ a_3\ a_2\ B_1\ C_3\ C_2\ C_1 \\ a_3\ a_2\ a_1\ C_3\ C_2\ C_1 \\ a_3\ a_2\ a_1\ b_3\ b_2\ b_1 \end{array}$$

Context-sensitive grammars: $L(G) = \{a^{2^i} \mid i \geq 1\}$

$$\begin{array}{rcl} S & \rightarrow & A\ C\ a\ B \\ C\ a & \rightarrow & a\ a\ C \\ C\ B & \rightarrow & D\ B \\ \mathbf{C\ B} & \rightarrow & \mathbf{E} \\ a\ D & \rightarrow & D\ a \\ A\ D & \rightarrow & A\ C \\ a\ E & \rightarrow & E\ a \\ \mathbf{A\ E} & \rightarrow & \epsilon \end{array}$$

Context-sensitive grammars: $L(G) = \{a^{2^i} \mid i \geq 1\}$

$$\begin{array}{r} S \\ A\ C\ a\ B \\ A\ a\ a\ C\ B \\ A\ a\ a\ E \\ A\ a\ E\ a \\ A\ E\ a\ a \\ a\ a \end{array}$$

Context-sensitive grammars: $L(G) = \{a^{2^i} \mid i \geq 1\}$

- A and B serve as left and right end-markers for sentential forms (derivation of each string)
- C is a marker that moves through the string of a's between A and B, doubling their number using $C\ a \rightarrow a\ a\ C$
- When C hits right end-marker B, it becomes a D or E by $C\ B \rightarrow D\ B$ or $C\ B \rightarrow E$
- If a D is chosen, that D migrates left using $a\ D \rightarrow D\ a$ until left end-marker A is reached
- At that point D becomes C using $A\ D \rightarrow A\ C$ and the process starts over
- Finally, E migrates left until it hits left end-marker A using $a\ E \rightarrow E\ a$

Strong vs. Weak Generative Capacity

- **Weak generative capacity** of a grammar is the set of strings or the language, e.g. 0^n1^n for $n \geq 0$
- **Strong generative capacity** is the set of structures (usually the set of trees) provided by the grammar

- **context-sensitive** grammars: 0^i, i is not a prime number and $i > 0$
- **indexed** grammars: $0^n1^n2^n \ldots m^n$, for any fixed m and $n \geq 0$
- **tree-adjoining** grammars (TAG), **linear-indexed** grammars (LIG), **combinatory categorial** grammars (CCG): $0^n1^n2^n3^n$, for $n \geq 0$
- **context-free** grammars: 0^n1^n for $n \geq 0$
- **deterministic context-free** grammars: $S' \rightarrow S\ c$, $S \rightarrow S\ A \mid A$, $A \rightarrow a\ S\ b \mid ab$: the language of "balanced parentheses"
- **regular** grammars: $(0|1)^*00(0|1)^*$

Language	*Automaton*	*Grammar*	*Recognition*	*Dependency*
Recursively Enumerable Languages	Turing Machine	Unrestricted Baa → A	Undecidable	Arbitrary
Context-Sensitive Languages	Linear-Bounded	Context-Sensitive At → aA	NP-Complete	Crossing
Context-Free Languages	Pushdown (stack)	Context-Free S → gSc	Polynomial	Nested
Regular Languages	Finite-State Machine	Regular A → cA	Linear	Strictly Local

Recognition Complexity

- Given grammar G and input x, provide algorithm for: Is $x \in L(G)$?
- **unrestricted**: undecidable
- **context-sensitive**: NSPACE[n] – linear non-deterministic space
- **indexed** grammars: NP-Complete
- **tree-adjoining** grammars (TAG), **linear-indexed** grammars (LIG), **combinatory categorial** grammars (CCG), **head** grammars: $O(n^6)$
- **context-free**: $O(n^3)$
- **deterministic context-free**: $O(n)$
- **regular** grammars: $O(n)$