CSE 135: Introduction to Theory of Computation
(A taste of) Chomsky Hierarchy

Sungjin Im

University of California, Merced

03-19-2015
Grammars

Definition
A grammar is $G = (V, \Sigma, R, S)$, where

- $V$ is a finite set of variables/non-terminals
- $\Sigma$ is a finite set of terminals
- $S \in V$ is the start symbol
- $R \subseteq (\Sigma \cup V)^* \times (\Sigma \cup V)^*$ is a finite set of rules/productions
Grammars

Definition
A grammar is $G = (V, \Sigma, R, S)$, where

- $V$ is a finite set of variables/non-terminals
- $\Sigma$ is a finite set of terminals
- $S \in V$ is the start symbol
- $R \subseteq (\Sigma \cup V)^* \times (\Sigma \cup V)^*$ is a finite set of rules/productions

We say $\gamma_1 \alpha \gamma_2 \Rightarrow_G \gamma_1 \beta \gamma_2$ iff $(\alpha \rightarrow \beta) \in R$. 
Grammars

Definition
A grammar is \( G = (V, \Sigma, R, S) \), where
- \( V \) is a finite set of variables/non-terminals
- \( \Sigma \) is a finite set of terminals
- \( S \in V \) is the start symbol
- \( R \subseteq (\Sigma \cup V)^* \times (\Sigma \cup V)^* \) is a finite set of rules/productions

We say \( \gamma_1 \alpha \gamma_2 \Rightarrow_G \gamma_1 \beta \gamma_2 \) iff \( (\alpha \rightarrow \beta) \in R \). And
\[ L(G) = \{ w \in \Sigma^* \mid S \Rightarrow_G^* w \} \]
Example

Consider the grammar \( G \) with \( \Sigma = \{a\} \) with

\[
\begin{align*}
S & \rightarrow \$Ca\# \mid a \mid \epsilon \\
Ca & \rightarrow aaC \\
C \# & \rightarrow D \# \mid E \\
D \# & \rightarrow $ \\
E & \rightarrow \epsilon \\
\end{align*}
\]

\( aD \rightarrow Da \quad aE \rightarrow Ea \)

The following are derivations in this grammar

\[
\begin{align*}
S & \Rightarrow \$Ca\# \Rightarrow \$aaC\# \Rightarrow \$aaE \Rightarrow \$aEa \Rightarrow \$Eaa \Rightarrow aa \\
S & \Rightarrow \$Ca\# \Rightarrow \$aaC\# \Rightarrow \$aaD\# \Rightarrow \$Da\# \Rightarrow \$Daa\# \Rightarrow \$Caa\#
\end{align*}
\]

\[
\begin{align*}
\Rightarrow \$aaCa\# \Rightarrow \$aaaaC\# \Rightarrow \$aaaaE \Rightarrow \$aaEa \Rightarrow \$aaEaa \\
\Rightarrow \$aEaaa \Rightarrow \$Eaaaa \Rightarrow aaaa
\end{align*}
\]

\[
\text{L}(G) = \{a^i \mid i \text{ is a power of 2}\}
\]
Example

Consider the grammar $G$ with $\Sigma = \{a\}$ with

$$
S \rightarrow Ca\# | a | \epsilon \\
Ca \rightarrow aaC \\
D \rightarrow \epsilon
$$

The following are derivations in this grammar

$$
S \Rightarrow Ca\# \Rightarrow aaC\# \Rightarrow aaE \Rightarrow aEa \Rightarrow Eaa \Rightarrow aa
$$

$$
S \Rightarrow Ca\# \Rightarrow aaC\# \Rightarrow aaD\# \Rightarrow aDa\# \Rightarrow Da\# \Rightarrow Caa\# \\
\Rightarrow aaCa\# \Rightarrow aaaaC\# \Rightarrow aaaaE \Rightarrow aaaEa \Rightarrow aaEaa \\
\Rightarrow aEaaa \Rightarrow Eaaaa \Rightarrow aaaa
$$

$L(G) = \{a^i | i \text{ is a power of 2}\}$
Grammars for each task

- What is the expressive power of these grammars?
Grammars for each task

- What is the expressive power of these grammars?
- Restricting the types of rules, allows one to describe different aspects of natural languages

Noam Chomsky
Grammars for each task

What is the expressive power of these grammars?

Restricting the types of rules, allows one to describe different aspects of natural languages

These grammars form a hierarchy

Noam Chomsky
Definition
Type 0 grammars are those where the rules are of the form

\[ \alpha \to \beta \]

where \( \alpha, \beta \in (\Sigma \cup V)^* \)

Example
Consider the grammar \( G \) with \( \Sigma = \{a\} \) with

\[
\begin{align*}
S & \to \$Ca\# \mid a \mid \epsilon \\
Ca & \to aaC \\
C\# & \to D\# \mid E \\
E & \to \epsilon \\
D & \to D\# \\
D & \to aD \\
E & \to aE \\
$D & \to $C \\
$aD & \to Da \\
$aE & \to Ea \\
\end{align*}
\]
Expressive Power of Type 0 Grammars

Theorem

$L$ is recursively enumerable iff there is a type 0 grammar $G$ such that $L = L(G)$. 
Expressive Power of Type 0 Grammars

Theorem
$L$ is recursively enumerable iff there is a type 0 grammar $G$ such that $L = L(G)$.

Thus, type 0 grammars are as powerful as Turing machines.
The rules in a type 1 grammar are of the form

\[ \alpha \rightarrow \beta \]

where \( \alpha, \beta \in (\Sigma \cup V)^* \) and \( |\alpha| \leq |\beta| \).
Type 1 Grammars

The rules in a type 1 grammar are of the form

\[ \alpha \rightarrow \beta \]

where \( \alpha, \beta \in (\Sigma \cup V)^* \) and \( |\alpha| \leq |\beta| \).
In every derivation, the length of the string never decreases.
Type 1 Grammars

The rules in a type 1 grammar are of the form

$$\alpha \rightarrow \beta$$

where $\alpha, \beta \in (\Sigma \cup V)^*$ and $|\alpha| \leq |\beta|$.

In every derivation, the length of the string never decreases.

**Example**
Consider the grammar $G$ with $\Sigma = \{a, b, c\}$, $V = \{S, B, C, H\}$ and

- $S \rightarrow aSBC \mid aBC$
- $HC \rightarrow BC$
- $bC \rightarrow bc$
- $CB \rightarrow HB$
- $aB \rightarrow ab$
- $cC \rightarrow cc$
- $HB \rightarrow HC$
- $bB \rightarrow bb$
Type 1 Grammars

The rules in a type 1 grammar are of the form

\[ \alpha \rightarrow \beta \]

where \( \alpha, \beta \in (\Sigma \cup V)^* \) and \( |\alpha| \leq |\beta| \).

In every derivation, the length of the string never decreases.

Example
Consider the grammar \( G \) with \( \Sigma = \{a, b, c\} \), \( V = \{S, B, C, H\} \) and

\[
\begin{align*}
S &\rightarrow aSBC \mid aBC \\
HC &\rightarrow BC \\
bC &\rightarrow bc
\end{align*}
\]

\[
\begin{align*}
CB &\rightarrow HB \\
HB &\rightarrow HC \\
aB &\rightarrow ab \\
bB &\rightarrow bb \\
cC &\rightarrow cc
\end{align*}
\]

\( L(G) = \{a^n b^n c^n \mid n \geq 0\} \)
Normal Form for Type 1 grammars

For every Type 1 grammar $G$, there is a grammar (in normal form) $G'$ such that $L(G) = L(G')$ and all the rules of $G'$ are of the form

$$\alpha_1 A \alpha_2 \rightarrow \alpha_1 \beta \alpha_2$$

where $A \in V$ and $\beta \in (\Sigma \cup V)^*$
Context Sensitivity

Normal Form for Type 1 grammars
For every Type 1 grammar $G$, there is a grammar (in normal form) $G'$ such that $L(G) = L(G')$ and all the rules of $G'$ are of the form

$$\alpha_1 A \alpha_2 \rightarrow \alpha_1 \beta \alpha_2$$

where $A \in V$ and $\beta \in (\Sigma \cup V)^*$
So, rules of $G'$ replace a variable $A$ by $\beta$ in the context $\alpha_1 \Box \alpha_2$. 

Thus, the class of language described by Type 1 grammars are called context-sensitive languages.
Context Sensitivity

Normal Form for Type 1 grammars

For every Type 1 grammar $G$, there is a grammar (in normal form) $G'$ such that $L(G) = L(G')$ and all the rules of $G'$ are of the form

$$\alpha_1 A \alpha_2 \rightarrow \alpha_1 \beta \alpha_2$$

where $A \in V$ and $\beta \in (\Sigma \cup V)^*$

So, rules of $G'$ replace a variable $A$ by $\beta$ in the context $\alpha_1 \square \alpha_2$.

Thus, the class of language described by Type 1 grammars are called context-sensitive languages.
Type 2 Grammars

The rules in a type 2 grammar are of the form

$$A \rightarrow \beta$$

where $A \in V$ and $\beta \in (\Sigma \cup V)^*$. 
Type 2 Grammars

The rules in a type 2 grammar are of the form

$$A \rightarrow \beta$$

where $$A \in V$$ and $$\beta \in (\Sigma \cup V)^*$$.

Type 2 grammars describe context-free languages.
Type 2 Grammars

The rules in a type 2 grammar are of the form

\[ A \rightarrow \beta \]

where \( A \in V \) and \( \beta \in (\Sigma \cup V)^* \).

Type 2 grammars describe context-free languages

Example

Consider \( G \) over \( \Sigma = \{0, 1\} \) with rules

\[ S \rightarrow \epsilon \mid 0S1 \]
Type 2 Grammars

The rules in a type 2 grammar are of the form

\[ A \rightarrow \beta \]

where \( A \in V \) and \( \beta \in (\Sigma \cup V)^* \).

Type 2 grammars describe context-free languages.

Example
Consider \( G \) over \( \Sigma = \{0, 1\} \) with rules

\[ S \rightarrow \epsilon \mid 0S1 \]

\( L(G) = \{0^n1^n \mid n \geq 0\} \)
Type 3 Grammars

The rules in a type 3 grammar are of the form

\[ A \rightarrow aB \quad \text{or} \quad A \rightarrow a \]

where \( A, B \in V \) and \( a \in \Sigma \cup \{\epsilon\} \).
Type 3 Grammars

The rules in a type 3 grammar are of the form

\[ A \to aB \quad \text{or} \quad A \to a \]

where \( A, B \in V \) and \( a \in \Sigma \cup \{\epsilon\} \).

Example
Consider the grammar over \( \Sigma = \{0, 1\} \) with rules

\[ S \to 1S \mid 0A \quad A \to \epsilon \mid 1A \mid 0S \]
Type 3 Grammars

The rules in a type 3 grammar are of the form

\[ A \to aB \quad \text{or} \quad A \to a \]

where \( A, B \in V \) and \( a \in \Sigma \cup \{\epsilon\} \).

**Example**

Consider the grammar over \( \Sigma = \{0, 1\} \) with rules

\[ S \to 1S \mid 0A \quad A \to \epsilon \mid 1A \mid 0S \]

\( L(G) = \{w \in \{0, 1\}^* \mid w \text{ has an odd number of 0s}\} \)
Type 3 Grammars and Regularity

Proposition

$L$ is regular iff there is a Type 3 grammar $G$ such that $L = L(G)$. 
Proposition

$L$ is regular iff there is a Type 3 grammar $G$ such that $L = L(G)$.

Proof.
Let $G = (V, \Sigma, R, S)$ be a type 3 grammar. Consider the NFA $M = (Q, \Sigma, \delta, q_0, F)$ where

\[ Q = V \cup \{ q_F \}, \quad q_F \not\in V \]
\[ q_0 = S \]
\[ F = \{ q_F \} \]
\[ \delta(A, a) = \{ B | A \rightarrow aB \in R \} \cup \{ q_F | A \rightarrow a \in R \} \quad \text{for} \quad A \in V \]
\[ \delta(q_F, a) = \emptyset \quad \text{for all} \quad a \]

$L(M) = L(G)$ as $\forall A \in V, \forall w \in \Sigma^*, A \Rightarrow G w$ iff $q_F \in \hat{\Delta}(A, w)$. 

\[ \rightarrow \]
Proposition

$L$ is regular iff there is a Type 3 grammar $G$ such that $L = L(G)$.

Proof.

Let $G = (V, \Sigma, R, S)$ be a type 3 grammar. Consider the NFA $M = (Q, \Sigma, \delta, q_0, F)$ where

$Q = V \cup \{q_F\}$, where $q_F \notin V$
Type 3 Grammars and Regularity

Proposition

$L$ is regular iff there is a Type 3 grammar $G$ such that $L = L(G)$.

Proof.

Let $G = (V, \Sigma, R, S)$ be a type 3 grammar. Consider the NFA $M = (Q, \Sigma, \delta, q_0, F)$ where

- $Q = V \cup \{q_F\}$, where $q_F \not\in V$
- $q_0 = S$
Proposition

$L$ is regular iff there is a Type 3 grammar $G$ such that $L = L(G)$.

Proof. 
Let $G = (V, \Sigma, R, S)$ be a type 3 grammar. Consider the NFA $M = (Q, \Sigma, \delta, q_0, F)$ where

- $Q = V \cup \{q_F\}$, where $q_F \notin V$
- $q_0 = S$
- $F = \{q_F\}$
Type 3 Grammars and Regularity

Proposition

$L$ is regular iff there is a Type 3 grammar $G$ such that $L = L(G)$.

Proof.

Let $G = (V, Σ, R, S)$ be a type 3 grammar. Consider the NFA $M = (Q, Σ, δ, q₀, F)$ where

- $Q = V \cup \{q₅\}$, where $q₅ \notin V$
- $q₀ = S$
- $F = \{q₅\}$
- $δ(A, a) = \{B \mid \text{if } A \rightarrow aB ∈ R\} \cup \{q₅ \mid \text{if } A \rightarrow a ∈ R\}$ for $A ∈ V$.
  And $δ(q₅, a) = \emptyset$ for all $a$. 

...→
Proposition

$L$ is regular iff there is a Type 3 grammar $G$ such that $L = L(G)$.

Proof.

Let $G = (V, \Sigma, R, S)$ be a type 3 grammar. Consider the NFA $M = (Q, \Sigma, \delta, q_0, F)$ where

- $Q = V \cup \{q_F\}$, where $q_F \notin V$
- $q_0 = S$
- $F = \{q_F\}$
- $\delta(A, a) = \{B \mid A \rightarrow aB \in R\} \cup \{q_F \mid A \rightarrow a \in R\}$ for $A \in V$. And $\delta(q_F, a) = \emptyset$ for all $a$.

$L(M) = L(G)$ as $\forall A \in V, \forall w \in \Sigma^*, A \xrightarrow{*} G w$ iff $q_F \in \hat{\Delta}(A, w)$.  

$\cdots \rightarrow$
Proof (contd).

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a NFA recognizing $L$. Consider $G = (V, \Sigma, R, S)$ where
Proof (contd).

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a NFA recognizing $L$. Consider $G = (V, \Sigma, R, S)$ where

- $V = Q$
Proof (contd).

Let \( M = (Q, \Sigma, \delta, q_0, F) \) be a NFA recognizing \( L \). Consider \( G = (V, \Sigma, R, S) \) where

- \( V = Q \)
- \( S = q_0 \)

Thus, \( L(M) = L(G) \). □
Type 3 Grammars and Regularity
NFA to Grammars

Proof (contd).
Let $M = (Q, \Sigma, \delta, q_0, F)$ be a NFA recognizing $L$. Consider
$G = (V, \Sigma, R, S)$ where

- $V = Q$
- $S = q_0$
- $q_1 \rightarrow aq_2 \in R$ iff $q_2 \in \delta(q_1, a)$ and $q \rightarrow \epsilon \in R$ iff $q \in F$.

\[ \square \]
Type 3 Grammars and Regularity

NFA to Grammars

Proof (contd).

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a NFA recognizing $L$. Consider $G = (V, \Sigma, R, S)$ where

1. $V = Q$
2. $S = q_0$
3. $q_1 \rightarrow aq_2 \in R$ iff $q_2 \in \delta(q_1, a)$ and $q \rightarrow \epsilon \in R$ iff $q \in F$.

We can show, for any $q, q' \in Q$ and $w \in \Sigma^*$, $q' \in \hat{\Delta}(q, w)$ iff $q \Rightarrow^*_G wq'$. Thus, $L(M) = L(G)$. □
Grammars and their Languages

<table>
<thead>
<tr>
<th>Grammar</th>
<th>Rules</th>
<th>Languages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type 3</td>
<td>$A \rightarrow aB$ or $A \rightarrow a$</td>
<td>Regular</td>
</tr>
<tr>
<td></td>
<td>$A \rightarrow \alpha$</td>
<td>Context Free</td>
</tr>
<tr>
<td>Type 2</td>
<td>$\alpha \rightarrow \beta$ with $</td>
<td>\alpha</td>
</tr>
<tr>
<td>Type 1</td>
<td>$\alpha \rightarrow \beta$</td>
<td>Recursively Enumerable</td>
</tr>
<tr>
<td>Type 0</td>
<td>$\alpha \rightarrow \beta$</td>
<td></td>
</tr>
</tbody>
</table>

In the above table, $\alpha, \beta \in (\Sigma \cup V)^*$, $A, B \in V$ and $a \in \Sigma \cup \{\epsilon\}$. 
Theorem

Type 0, Type 1, Type 2, and Type 3 grammars define a strict hierarchy of formal languages.
Theorem

Type 0, Type 1, Type 2, and Type 3 grammars define a strict hierarchy of formal languages.

Proof.

Clearly a Type 3 grammar is a special Type 2 grammar, a Type 2 grammar is a special Type 1 grammar, and a Type 1 grammar is special Type 0 grammar.

Moreover, there is a language that has a Type 2 grammar but no Type 3 grammar ($L = \{0^n1^n | n \geq 0\}$), a language that has a Type 1 grammar but no Type 2 grammar ($L = \{a^n b^n c^n | n \geq 0\}$), and a language with a Type 0 grammar but no Type 1 grammar. □
Theorem

Type 0, Type 1, Type 2, and Type 3 grammars define a strict hierarchy of formal languages.

Proof.

Clearly a Type 3 grammar is a special Type 2 grammar, a Type 2 grammar is a special Type 1 grammar, and a Type 1 grammar is special Type 0 grammar.

Moreover, there is a language that has a Type 2 grammar but no Type 3 grammar.
Theorem
Type 0, Type 1, Type 2, and Type 3 grammars define a strict hierarchy of formal languages.

Proof.
Clearly a Type 3 grammar is a special Type 2 grammar, a Type 2 grammar is a special Type 1 grammar, and a Type 1 grammar is special Type 0 grammar.
Moreover, there is a language that has a Type 2 grammar but no Type 3 grammar ($L = \{0^n1^n \mid n \geq 0\}$),
Theorem
Type 0, Type 1, Type 2, and Type 3 grammars define a strict hierarchy of formal languages.

Proof.
Clearly a Type 3 grammar is a special Type 2 grammar, a Type 2 grammar is a special Type 1 grammar, and a Type 1 grammar is special Type 0 grammar.
Moreover, there is a language that has a Type 2 grammar but no Type 3 grammar ($L = \{0^n1^n | n \geq 0\}$), a language that has a Type 1 grammar but no Type 2 grammar
Chomsky Hierarchy

Theorem

Type 0, Type 1, Type 2, and Type 3 grammars define a strict hierarchy of formal languages.

Proof.

Clearly a Type 3 grammar is a special Type 2 grammar, a Type 2 grammar is a special Type 1 grammar, and a Type 1 grammar is special Type 0 grammar.

Moreover, there is a language that has a Type 2 grammar but no Type 3 grammar \((L = \{0^n1^n \mid n \geq 0\})\), a language that has a Type 1 grammar but no Type 2 grammar \((L = \{a^n b^n c^n \mid n \geq 0\})\),
Theorem

Type 0, Type 1, Type 2, and Type 3 grammars define a strict hierarchy of formal languages.

Proof.

Clearly a Type 3 grammar is a special Type 2 grammar, a Type 2 grammar is a special Type 1 grammar, and a Type 1 grammar is special Type 0 grammar.

Moreover, there is a language that has a Type 2 grammar but no Type 3 grammar ($L = \{0^n1^n \mid n \geq 0\}$), a language that has a Type 1 grammar but no Type 2 grammar ($L = \{a^n b^n c^n \mid n \geq 0\}$), and a language with a Type 0 grammar but no Type 1 grammar. □
Overview of Languages

- Regular Languages = Type 3
- Context-Free Languages (CFL) = Type 2
- Context-Sensitive Languages (CSL) = Type 1

- Decidable Languages
  - Type 0: Recursively Enumerable Languages
  - Type 1: Context-Sensitive Languages (CSL)
  - Type 2: Context-Free Languages (CFL)
  - Type 3: Regular Languages

- A_TM
  - Decidable Languages:
    - Type 0: Recursively Enumerable Languages
      - A_TM
    - Type 1: Context-Sensitive Languages (CSL)
      - L_{anbncn}
    - Type 2: Context-Free Languages (CFL)
      - L_{0n1n}
    - Type 3: Regular Languages