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

Sungjin Im

University of California, Merced

04-02-2014

Grammars

Definition

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

- V is a finite set of variables/non-terminals
- Σ 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
- Σ 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 $(\alpha \to \beta) \in R$.

Grammars

Definition

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

- V is a finite set of variables/non-terminals
- Σ 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
$L(G) = \{w \in \Sigma^* \mid S \overset{*}{\Rightarrow}_G w\}$

Example

Example

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

$$\begin{array}{lll} S \rightarrow \$Ca\# \mid a \mid \epsilon & Ca \rightarrow aaC & \$D \rightarrow \$C \\ C\# \rightarrow D\# \mid E & aD \rightarrow Da & aE \rightarrow Ea \\ \$E \rightarrow \epsilon & & \end{array}$$

The following are derivations in this grammar

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

$$\begin{aligned} S &\Rightarrow \$Ca\# \Rightarrow \$aaC\# \Rightarrow \$aaD\# \Rightarrow \$aDa\# \Rightarrow \$Daa\# \Rightarrow \$Caa\# \\ &\Rightarrow \$aaCa\# \Rightarrow \$aaaaC\# \Rightarrow \$aaaaE \Rightarrow \$aaaEa \Rightarrow \$aaEaa \\ &\Rightarrow \$aEaaa \Rightarrow \$Eaaaa \Rightarrow aaaa \end{aligned}$$

Example

Example

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

$$\begin{array}{lll} S \rightarrow \$Ca\# \mid a \mid \epsilon & Ca \rightarrow aaC & \$D \rightarrow \$C \\ C\# \rightarrow D\# \mid E & aD \rightarrow Da & aE \rightarrow Ea \\ \$E \rightarrow \epsilon & & \end{array}$$

The following are derivations in this grammar

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

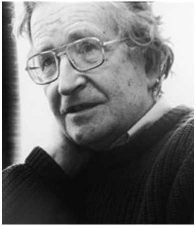
$$\begin{aligned} S &\Rightarrow \$Ca\# \Rightarrow \$aaC\# \Rightarrow \$aaD\# \Rightarrow \$aDa\# \Rightarrow \$Daa\# \Rightarrow \$Caa\# \\ &\Rightarrow \$aaCa\# \Rightarrow \$aaaaC\# \Rightarrow \$aaaaE \Rightarrow \$aaaEa \Rightarrow \$aaEaa \\ &\Rightarrow \$aEaaa \Rightarrow \$Eaaaa \Rightarrow aaaa \end{aligned}$$

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

Grammars for each task

- What is the expressive power of these grammars?

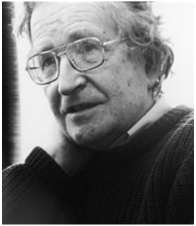
Grammars for each task



Noam Chomsky

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

Grammars for each task



Noam Chomsky

- 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

Type 0 Grammars

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

$$\begin{array}{lll} S \to \$Ca\# \mid a \mid \epsilon & Ca \to aaC & \$D \to \$C \\ C\# \to D\# \mid E & aD \to Da & aE \to Ea \\ \$E \to \epsilon & & \end{array}$$

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.

Type 1 Grammars

The rules in a type 1 grammar are of the form

$$\alpha \to \beta$$

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

$$\alpha \to \beta$$

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

$$\alpha \to \beta$$

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

$$\begin{array}{lll} S \to aSBC \mid aBC & CB \to HB & HB \to HC \\ HC \to BC & aB \to ab & bB \to bb \\ bC \to bc & cC \to cc & \end{array}$$

Type 1 Grammars

The rules in a type 1 grammar are of the form

$$\alpha \to \beta$$

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

$$\begin{array}{lll} S \to aSBC \mid aBC & CB \to HB & HB \to HC \\ HC \to BC & aB \to ab & bB \to bb \\ bC \to bc & cC \to cc & \end{array}$$

$L(G) = \{a^n b^n c^n \mid n \geq 0\}$

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 $\beta \in (\Sigma \cup V)^*$

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

$$\alpha_1 A \alpha_2 \to \alpha_1 \beta \alpha_2$$

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

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

$$\alpha_1 A \alpha_2 \to \alpha_1 \beta \alpha_2$$

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

$$A \to \beta$$

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

Type 2 Grammars

The rules in a type 2 grammar are of the form

$$A \to \beta$$

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

$$A \to \beta$$

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

$$S \to \epsilon \mid 0S1$$

Type 2 Grammars

The rules in a type 2 grammar are of the form

$$A \to \beta$$

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

$$S \to \epsilon \mid 0S1$$

$L(G) = \{0^n 1^n \mid n \geq 0\}$

Type 3 Grammars

The rules in a type 3 grammar are of the form

$$A \to aB \qquad \text{or} \qquad A \to a$$

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

$$A \to aB \qquad \text{or} \qquad A \to a$$

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

Example

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

$$S \to 1S \mid 0A \qquad A \to \epsilon \mid 1A \mid 0S$$

Type 3 Grammars

The rules in a type 3 grammar are of the form

$$A \to aB \qquad \text{or} \qquad A \to a$$

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

Example

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

$$S \to 1S \mid 0A \qquad A \to \epsilon \mid 1A \mid 0S$$

$L(G) = \{w \in \{0,1\}^* \mid w \text{ has an odd number of 0s}\}$

Type 3 Grammars and Regularity

Proposition

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

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

$\cdot\cdot\longrightarrow$

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 $q_F \notin V$

$\cdot\cdot\longrightarrow$

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 $q_F \notin V$
- $q_0 = S$

$\cdot\cdot\longrightarrow$

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 $q_F \notin V$
- $q_0 = S$
- $F = \{q_F\}$

$\cdot\cdot\longrightarrow$

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

$\cdot\cdot\!\longrightarrow$

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

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

Proof (contd).

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

$\square$

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

- $V = Q$

$\square$

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

- $V = Q$
- $S = q_0$

□

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 $q \in F$.

$\square$

Proof (contd).

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

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

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

Grammars and their Languages

Grammar	Rules	Languages
Type 3	$A \rightarrow aB$ or $A \rightarrow a$	Regular
Type 2	$A \rightarrow \alpha$	Context Free
Type 1	$\alpha \rightarrow \beta$ with $\lvert\alpha\rvert \leq \lvert\beta\rvert$	Context Sensitive
Type 0	$\alpha \rightarrow \beta$	Recursively Enumerable

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

Chomsky Hierarchy

Theorem

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

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.

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

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 ($L = \{0^n 1^n \mid n \geq 0\}$),

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

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 ($L = \{0^n 1^n \mid n \geq 0\}$), a language that has a Type 1 grammar but no Type 2 grammar ($L = \{a^n b^n c^n \mid n \geq 0\}$),

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 ($L = \{0^n 1^n \mid n \geq 0\}$), 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. $\square$

Overview of Languages