CS 301 - Lecture 5
Regular Grammars, Regular Languages, and Properties of Regular Languages
Fall 2008

Review
• Languages and Grammars
  – Alphabets, strings, languages
• Regular Languages
  – Deterministic Finite Automata
  – Nondeterministic Finite Automata
  – Equivalence of NFA and DFA
  – Regular Expressions
• Today:
  – Regular Grammars and Regular Languages
  – Properties of Regular Languages

Grammars
• Grammars express languages

Example: the English language
\[
\langle sentence \rangle \rightarrow \langle noun \_ phrase \rangle \ \langle predicate \rangle
\]
\[
\langle noun \_ phrase \rangle \rightarrow \langle article \rangle \ \langle noun \rangle
\]
\[
\langle predicate \rangle \rightarrow \langle verb \rangle
\]

Grammar Notation

Production Rules

\[
\langle noun \rangle \rightarrow \text{cat}
\]
\[
\langle noun \rangle \rightarrow \text{dog}
\]
Variable Terminal
Some Terminal Rules

\[ \langle \text{article} \rangle \rightarrow \text{a} \]
\[ \langle \text{article} \rangle \rightarrow \text{the} \]
\[ \langle \text{noun} \rangle \rightarrow \text{cat} \]
\[ \langle \text{noun} \rangle \rightarrow \text{dog} \]
\[ \langle \text{verb} \rangle \rightarrow \text{runs} \]
\[ \langle \text{verb} \rangle \rightarrow \text{walks} \]

A Resulting Sentence

\[ \langle \text{sentence} \rangle \Rightarrow \langle \text{noun\_phrase} \rangle \langle \text{predicate} \rangle \]
\[ \Rightarrow \langle \text{noun\_phrase} \rangle \langle \text{verb} \rangle \]
\[ \Rightarrow \langle \text{article} \rangle \langle \text{noun} \rangle \langle \text{verb} \rangle \]
\[ \Rightarrow \text{the} \langle \text{noun} \rangle \langle \text{verb} \rangle \]
\[ \Rightarrow \text{the} \text{ dog} \langle \text{verb} \rangle \]
\[ \Rightarrow \text{the} \text{ dog walks} \]

The Resulting Language

\[ L = \{ \text{"a cat runs"}, \]
\[ \text{"a cat walks"}, \]
\[ \text{"the cat runs"}, \]
\[ \text{"the cat walks"}, \]
\[ \text{"a dog runs"}, \]
\[ \text{"a dog walks"}, \]
\[ \text{"the dog runs"}, \]
\[ \text{"the dog walks"} \} \]

Definition of a Grammar

\[ G = (V, T, S, P) \]

\[ V \]: Set of variables

\[ T \]: Set of terminal symbols

\[ S \]: Start variable

\[ P \]: Set of Production rules
A Simple Grammar

- Grammar:
  
  \[ S \to aSb \]
  
  \[ S \to \lambda \]

- Derivation of sentence: \( ab \)
  
  \[ S \Rightarrow aSb \Rightarrow ab \]

Example Grammar Notation

\[ S \to aSb \]

\[ S \to \lambda \]

\[ G = (V, T, S, P) \]

\[ V = \{ S \} \]

\[ T = \{ a, b \} \]

\[ P = \{ S \to aSb, S \to \lambda \} \]

Deriving Strings in the Grammar

- Grammar:
  
  \[ S \to aSb \]
  
  \[ S \to \lambda \]

- Derivation of sentence: \( aabb \)
  
  \[ S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aabb \]

Sentential Form

- A sentence that contains variables and terminals

\[ S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbb \Rightarrow aabbb \]
General Notation for Derivations

* In general we write: \( w_1 \Rightarrow w_n \)

* If: \( w_1 \Rightarrow w_2 \Rightarrow w_3 \Rightarrow \cdots \Rightarrow w_n \)

* It is always the case that: \( w \Rightarrow w \)

Why Notation Is Useful

* We can now write:

\[
S \Rightarrow aaabbb
\]

* Instead of:

\[
S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow aaabbb
\]

Language of a Grammar

Grammar can produce some set of strings

Set of strings over an alphabet is a language

Language of a grammar is all strings produced by the grammar

\[
L(G) = \{ w : S \Rightarrow w \}
\]

String of terminals

Example Language

\[
S \rightarrow aSb
\]

\[
S \rightarrow \lambda
\]

Consider the set of all strings that can derived from this grammar.....

\[
S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow aaabbb
\]

\[
S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb
\]

\[
\Rightarrow aaaaSbbbb \Rightarrow aaaaabbb
\]

What language is being described?
The Resulting Language

\[ S \rightarrow aSb \]
\[ S \rightarrow \lambda \]

Always add on a and b on each side resulting in:
- a’s at the left
- b’s at the right
- equal number of a’s and b’s

Linear Grammars

- Grammars with at most one variable at the right side of a production

Examples:
\[ S \rightarrow aSb \]
\[ S \rightarrow \lambda \]

A Non-Linear Grammar

Grammar \( G \) :
\[ S \rightarrow SS \]
\[ S \rightarrow \lambda \]
\[ S \rightarrow aSb \]
\[ S \rightarrow bSa \]
\[ L(G) = \{ w : n_a(w) = n_b(w) \} \]

Number of \( a \)’s in string \( w \)

Another Linear Grammar

Grammar \( G \) :
\[ S \rightarrow A \]
\[ A \rightarrow aB | \lambda \]
\[ B \rightarrow Ab \]
\[ L(G) = \{ a^n b^n : n \geq 0 \} \]
Right-Linear Grammars

- All productions have form:
  \[ A \rightarrow xB \]
  or
  \[ A \rightarrow x \]
- Example:
  \[ S \rightarrow abS \]
  \[ S \rightarrow a \]  string of terminals

Left-Linear Grammars

- All productions have form:
  \[ A \rightarrow Bx \]
  or
  \[ A \rightarrow x \]
- Example:
  \[ S \rightarrow Aab \]
  \[ A \rightarrow Aab \mid B \]
  \[ B \rightarrow a \]  string of terminals

Regular Grammars

- A regular grammar is any right-linear or left-linear grammar
- Examples:
  \[ S \rightarrow abS \]
  \[ S \rightarrow Aab \]
  \[ S \rightarrow a \]
  \[ A \rightarrow Aab \mid B \]
  \[ B \rightarrow a \]

What languages are generated by these grammars?
Languages and Grammars

\[
\begin{align*}
S & \rightarrow abS \\
S & \rightarrow a \\
A & \rightarrow Aab | B \\
B & \rightarrow a
\end{align*}
\]

\[L(G_1) = (ab)^*a \quad L(G_2) = aab(ab)^*\]

Note both these languages are regular
we have regular expressions for these languages (above)
we can convert a regular expression into an NFA (how?)
we can convert an NFA into a DFA (how?)
we can convert a DFA into a regular expression (how?)
Do regular grammars also describe regular languages??

---

Regular Grammars Generate Regular Languages

Theorem

\[
\begin{align*}
\{ \text{Languages Generated by Regular Grammars} \} &= \{ \text{Regular Languages} \}
\end{align*}
\]

Theorem - Part 1

\[
\begin{align*}
\{ \text{Languages Generated by Regular Grammars} \} &\subseteq \{ \text{Regular Languages} \}
\end{align*}
\]

Any regular grammar generates a regular language
Proof – Part 1.

Languages Generated by Regular Grammars ⊆ Regular Languages

The language $L(G)$ generated by any regular grammar $G$ is regular.

The case of Right-Linear Grammars

- Let $G$ be a right-linear grammar
- We will prove: $L(G)$ is regular

- Proof idea: We will construct NFA using the grammar transitions

Example

Given right linear grammar:

$V_0 \rightarrow aV_1$
$V_1 \rightarrow abV_0|b$

Step 1: Create States for Each Variable

- Construct NFA $M$ such that every state is a grammar variable:

$V_0 \rightarrow aV_1$
$V_1 \rightarrow abV_0|b$
Step 2.1: Edges for Productions

- Productions of the form $V_i \rightarrow aV_j$ result in $\delta(V_i, a) = V_j$

$V_0 \rightarrow aV_1$
$V_1 \rightarrow abV_0 | b$

Step 2.2: Edges for Productions

- Productions of the form $V_i \rightarrow wV_j$ are only slightly harder…. Create row of states that derive $w$ and end in $V_j$

$V_0 \rightarrow aV_1$  
$V_1 \rightarrow abV_0 | b$

In General

- Given any right-linear grammar, the previous procedure produces an NFA
  - We sketched a proof by construction
  - Result is both a proof and an algorithm
  - Why doesn’t this work for a non-linear grammar?
- Since we have an NFA for the language, the right-linear grammar produces a regular language
Any regular language \( L \) is generated by some regular grammar \( G \).

Proof idea:
Let \( M \) be the NFA with \( L = L(M) \).

Construct from \( M \) a regular grammar \( G \) such that \( L(M) = L(G) \).

NFA to Grammar Example
• Since \( L \) is regular there is an NFA

Step 1: Convert Edges to Productions
\[
\begin{align*}
q_0 &\rightarrow aq_1 \\
q_1 &\rightarrow bq_1 \\
q_1 &\rightarrow aq_2 \\
q_2 &\rightarrow bq_3
\end{align*}
\]
In General

• Given any NFA, the previous procedure produces a right linear grammar
  – We sketched a proof by construction
  – Result is both a proof and an algorithm

• Every regular language has an NFA
  – Can convert that NFA into a right linear grammar
  – Thus every regular language has a right linear grammar

• Combined with Part 1, we have shown right linear grammars are yet another way to describe regular languages

But What About Left-Linear Grammars

• What happens if we reverse a left linear grammar as follows:
  \[ V_i \rightarrow V_j w \]  Reverses to  \[ V_i \rightarrow w^{R} V_j \]
  \[ V_i \rightarrow w \]  Reverses to  \[ V_i \rightarrow w^{R} \]
  – The result is a right linear grammar.
  – If the left linear grammar produced L, then what does the resulting right linear grammar produce?
But What About Left-Linear Grammars

- The previous slide reversed the language!

\[ V_i \rightarrow V_j w \] Reverses to \[ V_i \rightarrow w^R V_j \]

\[ V_i \rightarrow w \] Reverses to \[ V_i \rightarrow w^R \]

- If the left linear grammar produced language \( L \), then the resulting right linear grammar produces \( L^R \).

Claim we just proved left linear grammars produce regular languages? Why?

Left-Linear Grammars Produce Regular Languages

- Start with a Left Linear grammar that produces \( L \) want to show \( L \) is regular

- Can produce a right linear grammar that produces \( L^R \)

- All right linear grammars produce regular languages so \( L^R \) is a regular language

- The reverse of a regular language is regular so \( (L^R)^R = L \) is a regular language!

For regular languages \( L_1 \) and \( L_2 \), we will prove that:

- Union: \( L_1 \cup L_2 \)
- Concatenation: \( L_1 L_2 \)
- Star: \( L_1^* \)
- Reversal: \( L_1^R \)
- Complement: \( \overline{L_1} \)
- Intersection: \( L_1 \cap L_2 \)

We say: Regular languages are closed under:

- Union: \( L_1 \cup L_2 \)
- Concatenation: \( L_1 L_2 \)
- Star: \( L_1^* \)
- Reversal: \( L_1^R \)
- Complement: \( \overline{L_1} \)
- Intersection: \( L_1 \cap L_2 \)
Regular language $L_1$  
Regular language $L_2$

$L(M_1) = L_1$  
$L(M_2) = L_2$

Single final state  
Single final state

Example:

$M_1$

$L_1 = \{a^n b\}_{n \geq 0}$

$M_2$

$L_2 = \{ba\}$

Union

Example:

NFA for $L_1 \cup L_2 = \{a^n b\} \cup \{ba\}$

$L_1 = \{a^n b\}$

$L_2 = \{ba\}$
**Concatenation**

NFA for $L_1L_2$

Example

$L_1L_2 = \{a^n b\} \{ba\} = \{a^n bba\}$

**Star Operation**

NFA for $L_1^*$

Example

NFA for $L_1^* = \{a^n b\}^* \{w = w_1 w_2 \cdots w_k | w_i \in L_1\}$
Reverse

NFA for $L_1^R$

1. Reverse all transitions
2. Make initial state final state and vice versa

Example

$L_1 = \{a^n b\}$

$L_1^R = \{b a^n\}$

Complement

1. Take the DFA that accepts $L_1$
2. Make final states non-final, and vice-versa

Example

$L_1 = \{a^n b\}$

$L_1 = \{a b\}^* - \{a^n b\}$
**Intersection**

DeMorgan’s Law:

\[ L_1 \cap L_2 = \overline{L_1 \cup L_2} \]

\( L_1, L_2 \) regular

\[ \overline{L_1}, \overline{L_2} \] regular

\[ \overline{L_1 \cup L_2} \] regular

\[ \overline{L_1} \cup \overline{L_2} \] regular

\[ L_1 \cap L_2 \] regular

---

**What’s Next**

- **Read**
  - Linz Chapter 1.2, 2.1, 2.2, 2.3, (skip 2.4), 3, and Chapter 4
  - JFLAP Startup, Chapter 1, 2.1, (skip 2.2), 3, 4
- **Next Lecture Topics from Chapter 4.2 and 4.3**
  - Properties of regular languages
  - The pumping lemma (for regular languages)
- **Quiz 1 in Recitation on Wednesday 9/17**
  - Covers Linz 1.1, 1.2, 2.1, 2.2, 2.3, and JFLAP 1, 2.1
  - Closed book, but you may bring one sheet of 8.5 x 11 inch paper with any notes you like.
  - Quiz will take the full hour on Wednesday
- **Homework**
  - Homework Due Thursday