Automata Theory :: (Regular) Grammars

Jörg Endrullis

Vrije Universiteit Amsterdam

Introduction to Grammars

A **grammar** defines a **language**.

Applications areas:

- natural language
- artificial intelligence
- syntax of programming languages

Example

$$
\begin{aligned}
\langle \text{sentence} \rangle &\rightarrow \langle \text{article} \rangle\ \langle \text{noun} \rangle\ \langle \text{verb} \rangle\ \langle \text{article} \rangle\ \langle \text{noun} \rangle \\
\langle \text{article} \rangle &\rightarrow \text{the} \\
\langle \text{article} \rangle &\rightarrow \text{a} \\
\langle \text{noun} \rangle &\rightarrow \text{farmer} \\
\langle \text{noun} \rangle &\rightarrow \text{cow} \\
\langle \text{verb} \rangle &\rightarrow \text{milks}
\end{aligned}
$$

With these **grammar rules** we can construct a $\langle$sentence$\rangle$.

Introduction to Grammars

$$
\begin{aligned}
\langle\text{sentence}\rangle &\rightarrow \langle\text{article}\rangle\ \langle\text{noun}\rangle\ \langle\text{verb}\rangle\ \langle\text{article}\rangle\ \langle\text{noun}\rangle \\
\langle\text{article}\rangle &\rightarrow \text{the} \\
\langle\text{article}\rangle &\rightarrow \text{a} \\
\langle\text{noun}\rangle &\rightarrow \text{farmer} \\
\langle\text{noun}\rangle &\rightarrow \text{cow} \\
\langle\text{verb}\rangle &\rightarrow \text{milks}
\end{aligned}
$$

The farmer milks a cow is a sentence in the language.

$$
\begin{aligned}
\langle\text{sentence}\rangle &\Rightarrow \langle\text{article}\rangle\langle\text{noun}\rangle\langle\text{verb}\rangle\langle\text{article}\rangle\langle\text{noun}\rangle \\
&\Rightarrow \text{the } \langle\text{noun}\rangle\langle\text{verb}\rangle\langle\text{article}\rangle\langle\text{noun}\rangle \\
&\Rightarrow \text{the farmer } \langle\text{verb}\rangle\langle\text{article}\rangle\langle\text{noun}\rangle \\
&\Rightarrow \text{the farmer milks } \langle\text{article}\rangle\langle\text{noun}\rangle \\
&\Rightarrow \text{the farmer milks a } \langle\text{noun}\rangle \\
&\Rightarrow \text{the farmer milks a cow}
\end{aligned}
$$

Grammars

A **grammar** $G = (V, T, S, P)$ consists of:

- finite set V of **non-terminals** (or **variables**)
- finite set T of **terminals**
- a **start symbol** $S \in V$
- finite set P of **production rules** $x \to y$ where
  - $x \in (V \cup T)^+$ containing at least one symbol from V
  - $y \in (V \cup T)^*$

In the previous example:

- variables: ⟨sentence⟩, ⟨article⟩, ⟨noun⟩, ⟨verb⟩
- terminals: the, a, farmer, cow, milks
- starting symbol: ⟨sentence⟩

A grammar is **context-free** if $x \in V$ for every rule $x \to y$.

B(ackus) N(aur) F(orm) is a Context-Free Grammar

The BNF (Backus Naur Form) is often used to define the syntax of programming languages. These are context-free grammars!

Example

$$
\begin{array}{rcl}
\langle \text{stm} \rangle & \rightarrow & \langle \text{var} \rangle \ \textbf{:=} \ \langle \text{expr} \rangle \\
\langle \text{stm} \rangle & \rightarrow & \langle \text{stm} \rangle \ \textbf{;} \ \langle \text{stm} \rangle \\
\langle \text{stm} \rangle & \rightarrow & \textbf{begin} \ \langle \text{stm} \rangle \ \textbf{end} \\
\langle \text{stm} \rangle & \rightarrow & \textbf{if} \ \langle \text{cond} \rangle \ \textbf{then} \ \langle \text{stm} \rangle \ \textbf{else} \ \langle \text{stm} \rangle \\
\langle \text{stm} \rangle & \rightarrow & \textbf{while} \ \langle \text{cond} \rangle \ \textbf{do} \ \langle \text{stm} \rangle \\
\langle \text{cond} \rangle & \rightarrow & \cdots \\
\langle \text{var} \rangle & \rightarrow & \cdots \\
\langle \text{expr} \rangle & \rightarrow & \cdots \\
\cdots & \rightarrow & \cdots
\end{array}
$$

In BNF, non-terminals (variables) are indicated by $\langle$ and $\rangle$.

Grammar Derivations

If $x \to y$ is a production rule, then we have a **derivation step**

$$uxv \Rightarrow uyv$$

for every $u, v \in (V \cup T)^*$.

$G = (\{S\}, \{a, b\}, S, P)$, where P consists of

$$S \to aSb \qquad\qquad S \to \lambda$$

Example derivations:

$$S \Rightarrow \lambda \qquad\qquad S \Rightarrow^* \lambda$$

$$S \Rightarrow aSb \Rightarrow ab \qquad\qquad S \Rightarrow^* ab$$

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aabb \qquad\qquad S \Rightarrow^* aabb$$

A **derivation** $\Rightarrow^*$ is the reflexive, transitive closure of $\Rightarrow$.

Thus there is a derivation $u \Rightarrow^* v$ if v can be obtained from u by zero or more derivation steps.

Languages Generated by Grammars

The **language generated** by a grammar $G = (V, T, S, P)$ is

$$L(G) = \{ w \in T^* \mid S \Rightarrow^* w \}$$

The language consists of all words that

- contain only terminal letters (no variables), and
- can be derived from the start symbol

$G = (\{S\}, \{a, b\}, S, P)$, where P consists of

$$S \to aSb \qquad\qquad S \to \lambda$$

What is the language generated by G?

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

Recall that this language is not regular.

Notational Conventions for Grammars

Notational Conventions

When defining grammars, we use the following conventions:

- **upper case** letters for **variables** (non-terminals)
- **lower case** letters for **terminals**
- $V \rightarrow w_1 \mid \ldots \mid w_n$ is **shorthand for** n rules

$$\begin{array}{c} V \rightarrow w_1 \\ \vdots \\ V \rightarrow w_n \end{array}$$

Often, we only specify the production rules.

$$G_1: \quad S \rightarrow Ab \qquad\qquad G_2: \quad S \rightarrow aSb$$
$$\quad A \rightarrow aAb \mid \lambda \qquad\qquad S \rightarrow b$$

What languages are generated by these grammars?

$$L(G_1) = \{a^n b^{n+1} \mid n \geq 0\} \quad = \quad L(G_2) = \{a^n b^{n+1} \mid n \geq 0\}$$

Exercises (1)

Find a grammar G such that

$$L(G) = \{a, b\}^* \{c\}\{b, c\}^*$$

There are many possible solutions!

One possible solution is:

$$S \to XcY$$
$$X \to aX \mid bX \mid \lambda$$
$$Y \to bY \mid cY \mid \lambda$$

Exercises (2)

A word $w = a_1 a_2 \cdots a_n$ is called **palindrome** if $w = w^R$, that is

$$a_1 a_2 \cdots a_n = a_n \cdots a_2 a_1$$

For instance **hannah** is a palindrome.

Find a grammar G that generates all palindromes over the alphabet $\Sigma = \{a, b\}$. In other words

$$L(G) = \{\, w \in \Sigma^* \mid w \text{ is a palindrome}\}$$

Find a grammar G that generates all non-palindromes over the alphabet $\Sigma = \{a, b\}$. In other words

$$L(G) = \{\, w \in \Sigma^* \mid w \text{ is not a palindrome}\,\}$$

Regular Grammars

Right Linear Grammars

A grammar $G = (V, T, S, P)$ is **right linear** if all production rules are of the form

$$A \rightarrow uB \qquad \text{or} \qquad A \rightarrow u$$

with $A, B \in V$ and $u \in T^*$.

Moreover G is **strictly right linear** if $|u| \leq 1$ (i.e. $u \in (T \cup \{\lambda\})$).

Construct a right linear grammar G such that

$$L(G) = \{a, b\}^* \{aa\} \{b\}^*$$

Construct a right linear grammar G such that

$$L(G) = \{ab\} \left(\{a\}^* \{cb\}\right)^* \{b\}$$

(Strictly) Right Linear Grammars

Theorem

Let G be a right linear grammar G. There exists a **strictly** right linear grammar H such that $L(G) = L(H)$.

Construction

Let $G = (V, T, S, P)$ be a right linear grammar.

Assume that we have a production rule γ of the form

$$A \to u(B)$$

with $|u| > 1$. Then $u = aw$ for some $a \in T$ and $w \in T^+$.

Let X be a fresh variable ($X \notin (V \cup T)$).

We add X to V and split the rule γ into:

$$A \to aX \qquad\qquad X \to w(B)$$

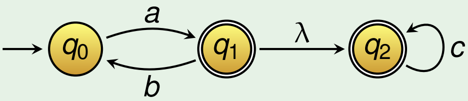
Then $A \to aX \to aw(B) = u(B)$. It follows $L(G) = L(H)$.

Repeat splitting until $|u| \leq 1$ for all rules.

Right Linear Grammars and Regular Languages

From NFAs to Right Linear Grammars

Consider the following NFA M



Construct a right linear grammar G **such that:**

$$L(M) = L(G)$$

From NFAs to Right Linear Grammars

For every NFA M there exists a right linear grammar G with

$$L(G) = L(M)$$

Construction

Let $M = (Q, \Sigma, \delta, \{q_0\}, F)$ be an NFA with a single starting state.

Define $G = (V, T, S, P)$ with $V = Q$ and $T = \Sigma$ and $S = q_0$.

The set P consists of the following production rules

$q \to \alpha q'$	for every $q' \in \delta(q, \alpha)$ where $\alpha \in \Sigma \cup \{\lambda\}$
$q \to \lambda$	for every $q \in F$

Then: $A \Rightarrow^* uB$ in $G \iff A \xrightarrow{u} B$ in M.

It follows that, $L(G) = L(M)$.

From Right Linear Grammars to NFAs

For every right linear grammar G there exists an NFA M with

$$L(M) = L(G)$$

Construction ($\Leftarrow$)

Let $G = (V, T, S, P)$ be a right linear grammar.

Make G to strictly right linear. Then all rules are of the form

$$A \to u \quad \text{or} \quad A \to uB$$

for $A, B \in V$, $u \in (T \cup \{\lambda\})$. Let NFA $M = (Q, \Sigma, \delta, \{S\}, F)$ with

$$\Sigma = T \qquad Q = V \cup \{\Omega\} \qquad F = \{\Omega\}$$

where $\Omega \notin V$. The transitions δ are given by

$$A \xrightarrow{u} B \qquad \text{for every } A \to uB \text{ in } G$$
$$A \xrightarrow{u} \Omega \qquad \text{for every } A \to u \text{ in } G$$

Then $S \Rightarrow^* w \in T^* \iff M$ accepts w. Hence $L(G) = L(M)$.

Exercise

Construct an NFA that accepts the language generated by

$$S \rightarrow aT \qquad\qquad T \rightarrow abcS \mid b$$

Note that $T \rightarrow abS \mid b$ is short for two rules:

$$T \rightarrow abcS$$
$$T \rightarrow b$$

Right Linear Grammars $\iff$ Regular Languages

Theorem

Language L is **regular**
$\iff$ there is a **right linear grammar** G with $L(G) = L$

Proof.

The proof consists of two directions:

- $(\Rightarrow)$ Translating NFAs to right linear grammars.
- $(\Leftarrow)$ Translate right linear grammars to NFAs.

We have already seen both constructions. $\square$

Left Linear Grammars

Left Linear Grammars

A grammar $G = (V, T, S, P)$ is **left linear** if all production rules are of the form

$$A \rightarrow Bu \qquad \text{or} \qquad A \rightarrow u$$

with $A, B \in V$ and $u \in T^*$.

(Difference with right linear grammars highlighted in red.)

Theorem

Language L is **regular**
$\iff$ there is a **left linear grammar** G with $L(G) = L$

Proof.

$$\begin{aligned} L \text{ is regular} &\iff L^R \text{ is regular} \\ &\iff \text{right linear grammar for } L^R \\ &\iff \text{left linear grammar for } L \end{aligned}$$

(For the last step, reverse both sides of all production rules.) □

Mixing Right and Left Linear Rules

Mixing right **and** left linear rules, the generated language is **not** always regular.

Example

Let G be the grammar

$$S \rightarrow aA$$
$$A \rightarrow Sb$$
$$S \rightarrow \lambda$$

Every rule of G is either right or left linear.

However, the language $L(G) = \{a^n b^n \mid n \geq 0\}$ is **not** regular.