university of groningen

Languages and Machines

L2: Context-Free Grammars

Jorge A. Pérez

Bernoulli Institute for Mathematics, Computer Science, and AI
University of Groningen, Groningen, the Netherlands

April 17, 2025

Previously: Regular Sets / Languages

- Recursively defined over an alphabet Σ from
  - $\emptyset$
  - $\{\epsilon\}$
  - $\{a\}$ for all $a \in \Sigma$

  by applying union, concatenation, and Kleene star.
- Regular expressions: a notation to denote *regular languages*
- Example:
  The regular expression

  $$\mathtt{a}^*(\mathtt{c} \mid \mathtt{d})\,\mathtt{b}^*$$

  denotes the regular set

  $$\{a\}^*(\{c\} \cup \{d\})\{b\}^*$$
- The regular expression of a set is not unique

Example

Language $L = \{a, b\}^*\{bb\}\{a, b\}^*$ is a regular set over $\Sigma = \{a, b\}$:

Example

Language $L = \{a, b\}^*\{bb\}\{a, b\}^*$ is a regular set over $\Sigma = \{a, b\}$:

1. From the basis, $\{a\}$ and $\{b\}$ are regular sets

Example

Language $L = \{a, b\}^*\{bb\}\{a, b\}^*$ is a regular set over $\Sigma = \{a, b\}$:

1. From the basis, $\{a\}$ and $\{b\}$ are regular sets
2. Applying union and Kleene star, we obtain $\{a, b\}^*$

Example

Language $L = \{a, b\}^*\{bb\}\{a, b\}^*$ is a regular set over $\Sigma = \{a, b\}$:

1. From the basis, $\{a\}$ and $\{b\}$ are regular sets
2. Applying union and Kleene star, we obtain $\{a, b\}^*$
3. Using concatenation, $\{b\}\{b\} = \{bb\}$ is regular

Example

Language $L = \{a, b\}^*\{bb\}\{a, b\}^*$ is a regular set over $\Sigma = \{a, b\}$:

1. From the basis, $\{a\}$ and $\{b\}$ are regular sets
2. Applying union and Kleene star, we obtain $\{a, b\}^*$
3. Using concatenation, $\{b\}\{b\} = \{bb\}$ is regular
4. Applying concatenation twice yields $\{a, b\}^*\{bb\}\{a, b\}^*$

Regular Expression Identities

Useful to algebraically manipulate regular expressions, and construct equivalent ones.

They relate **syntactically different** regular expressions that denote the **same** language:

- $\emptyset u = u\emptyset = \emptyset$
- $\epsilon u = u\epsilon = u$

Regular Expression Identities

Useful to algebraically manipulate regular expressions, and construct equivalent ones.

They relate **syntactically different** regular expressions that denote the **same** language:

- $\emptyset u = u\emptyset = \emptyset$
- $\epsilon u = u\epsilon = u$
- $\emptyset^* = \emptyset$
- $\epsilon^* = \epsilon$

Regular Expression Identities

Useful to algebraically manipulate regular expressions, and construct equivalent ones.

They relate **syntactically different** regular expressions that denote the **same** language:

- $\emptyset u = u\emptyset = \emptyset$
- $\epsilon u = u\epsilon = u$
- $\emptyset^* = \emptyset$
- $\epsilon^* = \epsilon$
- $u \mid v = v \mid u$
- $u \mid \emptyset = u$
- $u \mid u = u$
- $u(v \mid w) = uv \mid uw$
- $(u \mid v)w = uw \mid vw$
- $u^* = (u^*)^*$
- $(uv)^*u = u(vu)^*$
- $$\begin{aligned}(u \mid v)^* &= (u^* \mid v)^* \\ &= u^*(u \mid v)^* = (u \mid vu^*)^* \\ &= (u^*v^*)^* = u^*(vu^*)^* \\ &= (u^*vu^*)^*\end{aligned}$$

Regular Expression Identities

Useful to algebraically manipulate regular expressions, and construct equivalent ones.

They relate **syntactically different** regular expressions that denote the **same** language:

- $\emptyset u = u\emptyset = \emptyset$
- $\epsilon u = u\epsilon = u$
- $\emptyset^* = \emptyset$
- $\epsilon^* = \epsilon$
- $u \mid v = v \mid u$
- $u \mid \emptyset = u$
- $u \mid u = u$
- $u(v \mid w) = uv \mid uw$
- $(u \mid v)w = uw \mid vw$
- $u^* = (u^*)^*$
- $(uv)^*u = u(vu)^*$
- $$\begin{aligned} (u \mid v)^* &= (u^* \mid v)^* \\ &= u^*(u \mid v)^* = (u \mid vu^*)^* \\ &= (u^*v^*)^* = u^*(vu^*)^* \\ &= (u^*vu^*)^* \end{aligned}$$

An Example and a Non-Example

Language $L = a^* b^*$ is another regular set over $\{a, b\}$:

An Example and a Non-Example

Language $L = a^* b^*$ is another regular set over $\{a, b\}$:

- From the basis, $\{a\}$ and $\{b\}$ are regular sets; then we use Kleene star (twice), and finally we use concatenation.

An Example and a Non-Example

Language $L = a^* b^*$ is another regular set over $\{a, b\}$:

- From the basis, $\{a\}$ and $\{b\}$ are regular sets; then we use Kleene star (twice), and finally we use concatenation.
- $L = \{a^n b^m \mid n \geq 0, m \geq 0\}$

An Example and a Non-Example

Language $L = a^*b^*$ is another regular set over $\{a, b\}$:

- From the basis, $\{a\}$ and $\{b\}$ are regular sets; then we use Kleene star (twice), and finally we use concatenation.
- $L = \{a^n b^m \mid n \geq 0, m \geq 0\}$
- Given $u \in L$, the number of occurrences of a in u is **independent** from the number of occurrences of b. That is, we don't necessarily have that $n_a(u) = n_b(u)$.

An Example and a Non-Example

Language $L = a^* b^*$ is another regular set over $\{a, b\}$:

- From the basis, $\{a\}$ and $\{b\}$ are regular sets; then we use Kleene star (twice), and finally we use concatenation.
- $L = \{a^n b^m \mid n \geq 0, m \geq 0\}$
- Given $u \in L$, the number of occurrences of a in u is **independent** from the number of occurrences of b. That is, we don't necessarily have that $n_a(u) = n_b(u)$.

What about the language $L' = \{a^k b^k \mid k \geq 0\}$?

An Example and a Non-Example

Language $L = a^* b^*$ is another regular set over $\{a, b\}$:

- From the basis, $\{a\}$ and $\{b\}$ are regular sets; then we use Kleene star (twice), and finally we use concatenation.
- $L = \{a^n b^m \mid n \geq 0, m \geq 0\}$
- Given $u \in L$, the number of occurrences of a in u is **independent** from the number of occurrences of b. That is, we don't necessarily have that $n_a(u) = n_b(u)$.

What about the language $L' = \{a^k b^k \mid k \geq 0\}$?

- Intuition: we must "remember" $k = n_a(u)$ when generating occurrences of b

An Example and a Non-Example

Language $L = a^* b^*$ is another regular set over $\{a, b\}$:

- From the basis, $\{a\}$ and $\{b\}$ are regular sets; then we use Kleene star (twice), and finally we use concatenation.
- $L = \{a^n b^m \mid n \geq 0, m \geq 0\}$
- Given $u \in L$, the number of occurrences of a in u is **independent** from the number of occurrences of b. That is, we don't necessarily have that $n_a(u) = n_b(u)$.

What about the language $L' = \{a^k b^k \mid k \geq 0\}$?

- Intuition: we must "remember" $k = n_a(u)$ when generating occurrences of b
- There are regular expressions for *specific* strings in L': `a b`, `aa bb`, `aaa bbb`, ... This is not general enough.
- L' is not a regular language!
  What is it then? How can we generate it?

Context-Free Grammars

A formal system used to generate the strings of a language.
A quadruple (V, Σ, P, S) where

- V is a set of variables or nonterminals
- Σ is an alphabet of terminals, disjoint from V
- P is a finite set of production rules, taken from set $V \times (V \cup \Sigma)^*$.
  We write $A \to w$ instead of (A, w).
- $S \in V$ is the start symbol.

Context-Free Grammars

A formal system used to generate the strings of a language.
A quadruple (V, Σ, P, S) where

- V is a set of variables or nonterminals
- Σ is an alphabet of terminals, disjoint from V
- P is a finite set of production rules, taken from set $V \times (V \cup \Sigma)^*$.
  We write $A \to w$ instead of (A, w).
- $S \in V$ is the start symbol.

Example. We write

$$\begin{aligned} G:\ S &\to aSa \mid aBa \\ B &\to bB \mid b \end{aligned}$$

for the grammar $G = (V, \Sigma, P, S)$ where

- nonterminals $V \supseteq \{S, B\}$, with start symbol S
- Terminals $\Sigma \supseteq \{a, b\}$
- Production rules $P = \{(S, aSa), (S, aBa), (B, bB), (B, b)\}$

Some Terminology, by Example

$$
\begin{array}{rcl}
G: \; S & \rightarrow & aSa \mid aBa \\
B & \rightarrow & bB \mid b
\end{array}
$$

- $L(G)$, the language of G, is

Some Terminology, by Example

$$\begin{aligned} G:\ S &\rightarrow aSa \mid aBa \\ B &\rightarrow bB \mid b \end{aligned}$$

- $L(G)$, the language of G, is $\{a^i\, b^j\, a^i \mid i \geq 1, j \geq 1\}$

Some Terminology, by Example

$$\begin{array}{rrcl} G: & S & \rightarrow & aSa \mid aBa \\ & B & \rightarrow & bB \mid b \end{array}$$

- $L(G)$, the language of G, is $\{a^i\, b^j\, a^i \mid i \geq 1, j \geq 1\}$
- Some derivation steps: $S \Rightarrow_{(1)} aSa$ and $baB \Rightarrow_{(3)} babB$
- Let $\Rightarrow^*$ denote the reflexive, transitive closure of $\Rightarrow$.
  Some (G-)derivations: $baB \Rightarrow^* baB$ and $baB \Rightarrow^*_{(3,3,4)} babbb$

Some Terminology, by Example

$$\begin{array}{rcl} G: \; S & \rightarrow & aSa \mid aBa \\ B & \rightarrow & bB \mid b \end{array}$$

- $L(G)$, the language of G, is $\{a^i \, b^j \, a^i \mid i \geq 1, j \geq 1\}$
- Some derivation steps: $S \Rightarrow_{(1)} aSa$ and $baB \Rightarrow_{(3)} babB$
- Let $\Rightarrow^*$ denote the reflexive, transitive closure of $\Rightarrow$.
  Some (G-)derivations: $baB \Rightarrow^* baB$ and $baB \Rightarrow^*_{(3,3,4)} babbb$
- Some sentential forms:

$$S \Rightarrow_{(1)} aSa \Rightarrow_{(1)} aaSaa \Rightarrow_{(2)} aaaBaaa \Rightarrow_{(4)} aaabaaa$$

- The sentential form $aaabaaa$ is a sentence: it only has terminals.
  Its derivation can be shown using a derivation (or parse) tree.

Examples

$$
\begin{aligned}
G_1 : \quad S &\rightarrow A\ b\ A\ b\ A \\
A &\rightarrow a\ A \mid \epsilon
\end{aligned}
$$

$$
\begin{aligned}
G_2 : \quad S &\rightarrow a\ S \mid b\ A \\
A &\rightarrow a\ A \mid b\ C \\
C &\rightarrow a\ C \mid \epsilon
\end{aligned}
$$

What are $L(G_1)$ and $L(G_2)$?

Examples

$$
\begin{aligned}
G_1 : \quad S &\rightarrow A\ b\ A\ b\ A \\
A &\rightarrow a\ A \mid \epsilon
\end{aligned}
\qquad
\begin{aligned}
G_2 : \quad S &\rightarrow a\ S \mid b\ A \\
A &\rightarrow a\ A \mid b\ C \\
C &\rightarrow a\ C \mid \epsilon
\end{aligned}
$$

What are $L(G_1)$ and $L(G_2)$?

- $L(G_1) = L(G_2)$ contains strings over $\{a, b\}$ with **exactly** two occurrences of b. Regular expression: `a*ba*ba*`.

Examples

$$G_1: \quad S \rightarrow A\,b\,A\,b\,A$$
$$A \rightarrow a\,A \mid \epsilon$$

$$G_2: \quad S \rightarrow a\,S \mid b\,A$$
$$A \rightarrow a\,A \mid b\,C$$
$$C \rightarrow a\,C \mid \epsilon$$

What are $L(G_1)$ and $L(G_2)$?

- $L(G_1) = L(G_2)$ contains strings over $\{a, b\}$ with **exactly** two occurrences of b. Regular expression: $\mathtt{a^*ba^*ba^*}$.
- G_2 builds strings in a left-to-right manner

Examples

$$G_1: \quad \begin{aligned} S &\rightarrow A\, b\, A\, b\, A \\ A &\rightarrow a\, A \mid \epsilon \end{aligned}$$

$$G_2: \quad \begin{aligned} S &\rightarrow a\, S \mid b\, A \\ A &\rightarrow a\, A \mid b\, C \\ C &\rightarrow a\, C \mid \epsilon \end{aligned}$$

What are $L(G_1)$ and $L(G_2)$?

- $L(G_1) = L(G_2)$ contains strings over $\{a, b\}$ with **exactly** two occurrences of b.
  Regular expression: $\texttt{a}^*\texttt{ba}^*\texttt{ba}^*$.
- G_2 builds strings in a left-to-right manner
- Modify G_1 to generate strings with **at least** two occurrences of b:

$$G_1': \quad \begin{aligned} S &\rightarrow A\, b\, A\, b\, A \\ A &\rightarrow a\, A \mid b\, A \mid \epsilon \end{aligned}$$

More Terminology

- A derivation can transform any nonterminal in the string
- A leftmost derivation transforms the first nonterminal that occurs in a left-to-right reading of the string
- A grammar is ambiguous if there is a sentence with two different leftmost derivations.

Example. Consider the grammar

$$S \quad \rightarrow \quad aSb \mid aSbb \mid \epsilon$$

A sentence that shows that this grammar is ambiguous: $aabbb$.

Regular Grammars

A grammar (V, Σ, P, S) is regular if every production rule in P has one of the following forms ($a \in \Sigma$ and $A, B \in V$):

- $A \to aB$ or
- $A \to \epsilon$

A language is regular iff it is generated by a regular grammar.

Regular Grammars

A grammar (V, Σ, P, S) is regular if every production rule in P has one of the following forms ($a \in \Sigma$ and $A, B \in V$):

- $A \to aB$ or
- $A \to \epsilon$

A language is regular iff it is generated by a regular grammar.

Example. A non-regular grammar for the regular expression (ab)*a*:

$$\begin{array}{lcl} S & \to & abSA \mid \epsilon \\ A & \to & Aa \mid \epsilon \end{array}$$

An equivalent regular grammar:

$$\begin{array}{lcl} S & \to & aB \mid \epsilon \\ B & \to & bS \mid bA \\ A & \to & aA \mid \epsilon \end{array}$$

Proving Equality of Languages

Suppose we are given

- $L_1 = \{a^k b^m \mid k \geq 1, m \geq 0\}$, represented by $\mathtt{aa^*b^*}$
- G is defined as

$$\begin{array}{rcl} A & \rightarrow & aA \mid aB \\ B & \rightarrow & bB \mid \epsilon \end{array}$$

How to prove $L_1 = L(G)$?

Proving Equality of Languages

We split the thesis in two implications: $L_1 \subseteq L(G)$ and $L(G) \subseteq L_1$.
A sketch for each proof:

Proving Equality of Languages

We split the thesis in two implications: $L_1 \subseteq L(G)$ and $L(G) \subseteq L_1$.
A sketch for each proof:

- $L_1 \subseteq L(G)$
  Two steps to show that $u = a^k b^m$ is in $L(G)$:
  1. $A \Rightarrow^* a^k B$ (proven by induction on k)
  2. $B \Rightarrow^* b^m$ (proven by induction on m)

  This suffices to show that $A \Rightarrow^* u$.

Proving Equality of Languages

We split the thesis in two implications: $L_1 \subseteq L(G)$ and $L(G) \subseteq L_1$.
A sketch for each proof:

- $L_1 \subseteq L(G)$
  Two steps to show that $u = a^k b^m$ is in $L(G)$:
  1. $A \Rightarrow^* a^k B$ (proven by induction on k)
  2. $B \Rightarrow^* b^m$ (proven by induction on m)

  This suffices to show that $A \Rightarrow^* u$.
- $L(G) \subseteq L_1$
  If $u \in L(G)$ then there is a derivation $A \Rightarrow^* u$.
  To prove $u = a^k b^m$, note that for u to be a sentence we need:
  1. $n \in \mathbb{N}$ applications of $A \to aA$
  2. one application of $A \to aB$
  3. $m \in \mathbb{N}$ applications of $B \to bB$
  4. one application of $B \to \epsilon$

  Thus, $u \in L(G)$ implies $u = a^n ab^m = a^k b^m$, with $k = n + 1$. Therefore, $u \in \mathtt{aa^*b^*}$.

Productivity

Theorem

*Let G be a **productive** grammar.*
Assume that $w \in L(G)$ has length $|w| = k$.
Then every derivation of w according to G has length $\leq 2k + 1$.

Goal

We want to show that every context-free language has a **productive** grammar in which every symbol is **useful**.

Obtaining a productive grammar is a prerequisite for obtaining particular normal forms (such as Chomsky's)

Productive Grammars

A grammar (V, Σ, P, S) is called **productive** if it satisfies:

1. The start symbol S is nonrecursive, i.e., it does not occur at the righthand side of any production rule in P.
2. For every production rule $(A \to w) \in P$ with $A \neq S$, we have $w \in \Sigma$ or $|w| \geq 2$.

Note: The empty string ϵ is generated iff $S \to \epsilon$.

A Recipe

1. Make the start symbol nonrecursive
2. Remove all forbidden ϵ-productions
   - Ensure that ϵ is not produced by nonterminals different from S
   - Essentially noncontracting grammars, nullable nonterminals
3. Remove forbidden chain productions
   - Production rules of the form $A \to B$, with $A, B \in V$
   - Reflexive-transitive closure of $\to$

Given a grammar G and any of its transformations G', we must check that $L(G) = L(G')$.

Running Example

Consider the grammar G:

$$
\begin{array}{lcl}
A & \rightarrow & aA \mid B \\
B & \rightarrow & bB \mid \epsilon
\end{array}
$$

We have, e.g., $\{\epsilon, a, b, ab, abbb\} \subseteq L(G)$.

Running Example

Consider the grammar G:

$$\begin{array}{rcl} A & \rightarrow & aA \mid B \\ B & \rightarrow & bB \mid \epsilon \end{array}$$

We have, e.g., $\{\epsilon, a, b, ab, abbb\} \subseteq L(G)$. Why is G not productive?

Running Example

Consider the grammar G:

$$
\begin{aligned}
A &\rightarrow aA \mid B \\
B &\rightarrow bB \mid \epsilon
\end{aligned}
$$

We have, e.g., $\{\epsilon, a, b, ab, abbb\} \subseteq L(G)$. Why is G not productive?

Consider now G', a productive variant of G:

$$
\begin{aligned}
T &\rightarrow A \mid B \mid \epsilon \\
A &\rightarrow aA \mid a \mid aB \\
B &\rightarrow bB \mid b
\end{aligned}
$$

Do we have $\{\epsilon, a, b, ab, abbb\} \subseteq L(G')$?

Step 1: Nonrecursive Start Symbol

$$\begin{aligned} A &\rightarrow aA \mid B \\ B &\rightarrow bB \mid \epsilon \end{aligned} \quad \longrightarrow \quad \begin{aligned} T &\rightarrow A \\ A &\rightarrow aA \mid B \\ B &\rightarrow bB \mid \epsilon \end{aligned}$$

Step 2: Remove forbidden ϵ-productions

- $G = (V, \Sigma, P, S)$ is essentially noncontracting if S is nonrecursive and $(A \to \epsilon) \notin P$ for any $A \neq S$.
- $A \in V$ is nullable for G if there is a G-derivation $A \Longrightarrow^* \epsilon$.

Step 2: Remove forbidden ϵ-productions

1. Obtain the set of nullable nonterminals for the input grammar:

$$
\begin{array}{rcl}
T & \rightarrow & A \\
A & \rightarrow & aA \mid B \\
B & \rightarrow & bB \mid \epsilon
\end{array}
$$

The set is $\{T, A, B\}$, obtained using Algorithm 1 in the Reader.

Step 2: Remove forbidden ϵ-productions

1. Obtain the set of nullable nonterminals for the input grammar:

$$\begin{array}{rcl} T & \rightarrow & A \\ A & \rightarrow & aA \mid B \\ B & \rightarrow & bB \mid \epsilon \end{array}$$

   The set is $\{T, A, B\}$, obtained using Algorithm 1 in the Reader.
2. Use that set to extend the set of production rules:

$$\begin{array}{rcl} T & \rightarrow & A \mid \epsilon \\ A & \rightarrow & aA \mid B \mid a \mid \epsilon \\ B & \rightarrow & bB \mid \epsilon \mid b \end{array}$$

Step 2: Remove forbidden ϵ-productions

1. Obtain the set of nullable nonterminals for the input grammar:

$$\begin{array}{rcl} T & \rightarrow & A \\ A & \rightarrow & aA \mid B \\ B & \rightarrow & bB \mid \epsilon \end{array}$$

   The set is $\{T, A, B\}$, obtained using Algorithm 1 in the Reader.

2. Use that set to extend the set of production rules:

$$\begin{array}{rcl} T & \rightarrow & A \mid \epsilon \\ A & \rightarrow & aA \mid B \mid a \mid \epsilon \\ B & \rightarrow & bB \mid \epsilon \mid b \end{array}$$

3. Remove forbidden production rules $A \rightarrow \epsilon$:

$$\begin{array}{rcl} T & \rightarrow & A \mid \epsilon \\ A & \rightarrow & aA \mid B \mid a \\ B & \rightarrow & bB \mid b \end{array}$$

Does this grammar still generate $L(G)$?

Step 3: Remove chain production rules

1. Given the chain relation $\rightarrow$, get its reflexive, transitive closure:

$$\begin{array}{rcl} T & \rightarrow & A \mid \epsilon \\ A & \rightarrow & aA \mid B \mid a \\ B & \rightarrow & bB \mid b \end{array}$$

Chain relation: $T \rightarrow A$, $A \rightarrow B$

Closure: $T \rightarrow^* A$, $A \rightarrow^* B$, $T \rightarrow^* T$, $A \rightarrow^* A$, $B \rightarrow^* B$, $T \rightarrow^* B$

Step 3: Remove chain production rules

1. Given the chain relation $\rightarrow$, get its reflexive, transitive closure:

$$\begin{array}{rcl} T & \rightarrow & A \mid \epsilon \\ A & \rightarrow & aA \mid B \mid a \\ B & \rightarrow & bB \mid b \end{array}$$

   Chain relation: $T \rightarrow A$, $A \rightarrow B$

   Closure: $T \rightarrow^* A$, $A \rightarrow^* B$, $T \rightarrow^* T$, $A \rightarrow^* A$, $B \rightarrow^* B$, $T \rightarrow^* B$

2. Extend the production rules using $\rightarrow^*$:

$$\begin{array}{rcl} T & \rightarrow & A \mid \epsilon \mid B \\ A & \rightarrow & aA \mid B \mid a \mid aB \\ B & \rightarrow & bB \mid b \end{array}$$

Step 3: Remove chain production rules

1. Given the chain relation $\rightarrow$, get its reflexive, transitive closure:

$$\begin{array}{lcl} T & \rightarrow & A \mid \epsilon \\ A & \rightarrow & aA \mid B \mid a \\ B & \rightarrow & bB \mid b \end{array}$$

   Chain relation: $T \rightarrow A$, $A \rightarrow B$
   Closure: $T \rightarrow^* A$, $A \rightarrow^* B$, $T \rightarrow^* T$, $A \rightarrow^* A$, $B \rightarrow^* B$, $T \rightarrow^* B$
2. Extend the production rules using $\rightarrow^*$:

$$\begin{array}{lcl} T & \rightarrow & A \mid \epsilon \mid B \\ A & \rightarrow & aA \mid B \mid a \mid aB \\ B & \rightarrow & bB \mid b \end{array}$$

3. Remove all chain rules $A \rightarrow B$, with $A \neq T$:

$$\begin{array}{lcl} T & \rightarrow & A \mid \epsilon \mid B \\ A & \rightarrow & aA \mid a \mid aB \\ B & \rightarrow & bB \mid b \end{array}$$

Summing up

1. Make start symbol nonrecursive:

$$\begin{array}{lcl} A & \to & aA \mid B \\ B & \to & bB \mid \epsilon \end{array} \longrightarrow \begin{array}{lcl} T & \to & A \\ A & \to & aA \mid B \\ B & \to & bB \mid \epsilon \end{array}$$

2. Remove ϵ-productions $\rightsquigarrow$ Essentially contracting grammar

$$\begin{array}{lcl} T & \to & A \\ A & \to & aA \mid B \\ B & \to & bB \mid \epsilon \end{array} \longrightarrow \begin{array}{lcl} T & \to & A \mid \epsilon \\ A & \to & aA \mid B \mid a \\ B & \to & bB \mid b \end{array}$$

3. Remove chain production rules $\rightsquigarrow$ Productive grammar

$$\begin{array}{lcl} T & \to & A \mid \epsilon \\ A & \to & aA \mid B \mid a \\ B & \to & bB \mid b \end{array} \longrightarrow \begin{array}{lcl} T & \to & A \mid \epsilon \mid B \\ A & \to & aA \mid a \mid aB \\ B & \to & bB \mid b \end{array}$$

Chomsky Normal Form

A grammar $G = (V, \Sigma, P, S)$ is in **Chomsky normal form** if every production rule has one of the following forms:

1. $A \to BC$ with nonterminals A, B, C and $B \neq S$ and $C \neq S$
2. $A \to a$ with a nonterminal A and a terminal symbol a
3. $S \to \epsilon$ for the start symbol S.

Chomsky Normal Form

A grammar $G = (V, \Sigma, P, S)$ is in **Chomsky normal form** if every production rule has one of the following forms:

1. $A \to BC$ with nonterminals A, B, C and $B \neq S$ and $C \neq S$
2. $A \to a$ with a nonterminal A and a terminal symbol a
3. $S \to \epsilon$ for the start symbol S.

A productive grammar can be transformed into Chomsky normal form by introducing new nonterminals with new production rules:

$$\begin{array}{rcl} T & \to & A \mid \epsilon \\ A & \to & aA \mid B \mid a \\ B & \to & bB \mid b \end{array} \quad\longrightarrow\quad \begin{array}{rcl} T & \to & a \mid b \mid NA \mid MB \mid \epsilon \\ N & \to & a \\ M & \to & b \\ A & \to & NA \mid a \mid NB \\ B & \to & MB \mid b \end{array}$$

Useful, Generating & Generated Symbols

Let $G = (V, \Sigma, P, S)$ be a grammar. Let $x \in V \cup \Sigma$ be a symbol.

- x is useful if there is a derivation

$$S \implies^* uxv \implies^* w \qquad \text{with } u, v \in (V \cup \Sigma)^* \text{ and } w \in \Sigma^*$$

- x is called useless if it is not useful
- x is generating if $x \implies^* w$ holds for some $w \in \Sigma^*$
- x is generated if there are $u, v \in (V \cup \Sigma)^*$ with $S \implies^* uxv$.

Therefore:

- Useful symbols are both generating and generated
- However, generating and generated symbols may not be useful

Example 2.14

$$
\begin{aligned}
S &\rightarrow AB \mid cS \mid \epsilon \\
A &\rightarrow a \\
B &\rightarrow bB
\end{aligned}
$$

- B is not useful: it is useless, as it doesn't lead to any sentence
- A is generating, thanks to rule $A \rightarrow a$
- A is generated, thanks to rule $S \rightarrow AB$
- Still, A is useless: if it occurs in a sentential form, it comes with B, which is useless

Removal of Useless Symbols (Alg. 2)

To remove useless symbols in a given grammar G:

1. Compute the set T of generating nonterminals.
2. Assume $S \in T$ (i.e. non-empty $L(G)$).
   Transform G into a grammar G'' by
   - removing all nonterminals not in T, and
   - removing all production rules in which these nonterminals occur
3. Compute the set U of symbols generated by G''.
4. Transform G'' into G' by removing symbols not in U, and removing all production rules in which such symbols occur.

Taking Stock

- There are languages that are not regular
- Context-free grammars and languages
- Proving equality of languages
- Normal forms for context-free grammars
- Briefly: Useless symbols

Next time

Finite state machines: Recognizing regular languages (Sec 3.1 - 3.2)