II.4.1. Left Linear Grammars vs NFAs (13.7)

We will show that regular expressions coincide with regular languages and with languages recognised by a DFA or NFA. Here we prove one part of this result:

**Theorem (II.4.1.1)**

*For every right linear grammar $G$ there exists an NFA $A$ s.t. $L(G) = L(A)$*

*A can be computed from $G$.*
Proof Idea

A derivation of a word in $G$ has the form

$$S = A_0 \Rightarrow a_1 A_1 \Rightarrow a_1 a_2 A_2 \Rightarrow \cdots \Rightarrow a_1 a_2 \cdots a_{n-1} A_{n-1}$$

$$\Rightarrow a_1 a_2 \cdots a_{n-1} a_n$$

where we have productions

$$A_i \rightarrow a_{i+1} A_{i+1} \quad A_{n-1} \rightarrow a_n$$

or

$$S = A_0 \Rightarrow a_1 A_1 \Rightarrow a_1 a_2 A_2 \Rightarrow \cdots \Rightarrow a_1 a_2 \cdots a_{n-1} A_{n-1}$$

$$\Rightarrow a_1 a_2 \cdots a_{n-1}$$

where we have productions

$$A_i \rightarrow a_{i+1} A_{i+1} \quad A_{n-1} \rightarrow \epsilon$$

So we have:

- If $B \rightarrow aB'$, then $B \xrightarrow{a} B'$.
- If $B \rightarrow a$ then $B \xrightarrow{a} q_F$.
- $q_F \in F$.
- If $B \rightarrow \epsilon$, then $B \in F$.

Proof Idea

Define $A$ with states $N \cup \{q_F\}$ for a special new accepting state $q_F$ s.t.
the derivation

$$S = A_0 \Rightarrow a_1 A_1 \Rightarrow a_1 a_2 A_2 \Rightarrow \cdots \Rightarrow a_1 a_2 \cdots a_{n-1} A_{n-1}$$

$$\Rightarrow a_1 a_2 \cdots a_{n-1} a_n$$

corresponds to a sequence of transitions

$$S = A_0 \xrightarrow{a_1} A_1 \xrightarrow{a_2} A_2 \xrightarrow{a_3} \cdots \xrightarrow{a_{n-1}} A_{n-1} \xrightarrow{a_n} q_F$$

and a derivation

$$S = A_0 \Rightarrow a_1 A_1 \Rightarrow a_1 a_2 A_2 \Rightarrow \cdots \Rightarrow a_1 a_2 \cdots a_{n-1} A_{n-1}$$

$$\Rightarrow a_1 a_2 \cdots a_{n-1}$$

corresponds to a sequence of transitions

$$S = A_0 \xrightarrow{a_1} A_1 \xrightarrow{a_2} A_2 \xrightarrow{a_3} \cdots \xrightarrow{a_{n-1}} A_{n-1} \in F$$

Constructed NFA

We obtain from $G = (N, T, S, P)$ the following NFA:

- **automaton**: $A$
- **states**: $N \cup \{q_F\}$
- **terminals**: $T$
- **start**: $S$
- **final**: $B \in N$ s.t. $B \rightarrow \epsilon$. $q_F$
- **transitions**: $B \xrightarrow{a} B'$ if $B \rightarrow aB'$.
  $B \xrightarrow{a} q_F$ if $B \rightarrow a$. 

II.4.1. Left Linear Grammars vs NFAs (13.7)

Proof of Theorem II.4.1.1

Consider the Grammar:

- terminals: 0, 1
- nonterminals: S, T
- start symbol: S
- productions:
  - \( S \rightarrow 0 \)
  - \( S \rightarrow 1T \)
  - \( T \rightarrow 0T \)
  - \( T \rightarrow 1T \)
  - \( T \rightarrow \epsilon \)

Corresponding Automaton

(Note that it is non-deterministic.)

Example

Consider the Grammar:

\[
\begin{align*}
\text{grammar} & \quad G \\
\text{terminals} & \quad 0, 1 \\
\text{nonterminals} & \quad S, T \\
\text{start symbol} & \quad S \\
\text{productions} & \quad S \rightarrow 0, S \rightarrow 1T, \\
& \quad T \rightarrow 0T, T \rightarrow 1T, \\
& \quad T \rightarrow \epsilon, T \rightarrow 0, T \rightarrow 1
\end{align*}
\]
II.4.1. Left Linear Grammars vs NFAs (13.7)

Computing from NFA a Right Linear Grammar

- One can as well easily compute from an NFA an equivalent right linear grammar by inverting the above procedure:
  - Non-terminals are the set of states of the NFA.
  - Productions are
    \[ B \rightarrow aB' \text{ if } B \xrightarrow{a} B' \]
    and
    \[ B \rightarrow \epsilon \text{ if } B \text{ final state} \]
  - Start symbol = start state of the NFA.

Example: Consider the following Automaton

![Diagram of an Automaton]

1, 2, ..., 9

1, 2, ..., 9

0

0, 1, ..., 9

Left Linear Grammar obtained from the NFA

Assume an NFA \( A \) with states \( Q \), terminals \( T \), start state \( q_0 \), final states \( F \), transitions \( \rightarrow \).

The following is an equivalent NFA:

| grammar | \( A' \) |
| terminals | \( T \) |
| nonterminals | \( Q \) |
| start symbol | \( q_0 \) |
| productions | \[ q \rightarrow aq' \text{ if } q \xrightarrow{a} q' \]
  \[ q \rightarrow \epsilon \text{ if } q \in F \] |

Left Linear Grammar obtained from the NFA

Grammar: \( G \) NumbersNoLeadingZeros

| terminals | \( 0, 1, \ldots, 9 \) |
| nonterminals | \( q_0, q_1, q_2 \) |
| start symbol | \( q_0 \) |
| productions | \[ q_0 \rightarrow xq_1 \text{ for } x \in \{1, \ldots, 9\} \]
  \[ q_1 \rightarrow xq_1 \text{ for } x \in \{0, \ldots, 9\} \]
  \[ q_0 \rightarrow 0q_2 \]
  \[ q_1 \rightarrow \epsilon \]
  \[ q_2 \rightarrow \epsilon \] |
II.4.1. Left Linear Grammars vs NFAs (13.7)

II.4.1.2. Translating NFAs into Regular Expressions (13.10)

II.4.1.3. Equivalence Theorem

II.4.2. Closure Properties and Decidability of Regular Languages

II.4.3. The Pumping Lemma for Regular Languages (12.4, 12.5)

Theorem II.4.1.2

Let $A = (Q, q_0, F, T, \rightarrow)$ be an NFA. Then there exist a regular expression $E$ s.t. $L(E) = L(A)$. $E$ can be computed from $A$.

Proof of Theorem II.4.1.2

This proof is very interesting, so I recommend looking at it. It can be found together with an example in the additional material.
II.4.1.3. Equivalence Theorem

Let $L$ be a language over an alphabet $T$. The following are equivalent:

1. $L$ is definable by a regular expression.
2. $L$ is a regular.
3. $L$ is definable by a right-linear grammar.
4. $L$ is definable by a left-linear grammar.
5. $L$ is definable by an NFA with empty moves.
6. $L$ is definable by an NFA.
7. $L$ is definable by a DFA.

Furthermore, the corresponding regular expressions, right linear grammars, left-linear grammars, NFAs with empty moves, NFAs, DFAs can be computed from each other.

Proof of Theorem II.4.1.3

- The following directions have been introduced or at least sketched above:
  - Translations between DFA, NFA, NFA with empty moves.
  - Translation of regular expressions into left/right linear grammars.
  - Translation between NFA and left linear grammars.
- The following direction is more complicated:
  - Translation of NFA into regular expressions.
  - Translation between left linear and right linear grammars.
    - Left linear grammars can be translated into regular expressions and then into right linear grammars.
    - A regular expression for $L$ can easily be translated into a regular expression for $L^R$.
    - If we reverse the right hand sides of productions we obtain from a right linear grammar for $L$ a left linear grammar for $L^R$.
    - Now from a right linear grammar we obtain a left linear grammar for $L^R$; then a regular expression for $L^R$; then a regular expression for $L$; then a left linear grammar for $L$.
- Together the above translations provide a proof of Theorem II.4.1.3.
- Full details can be found in the additional material.

Directions in Equivalence Theorem

II.2.2.2. DFA

II.3.5.1. Left Linear
Reverse right side of transitions
Use Left Linear closed under reversing
II.3.4.1. NFA with empty moves
II.4.1.1. NFA
II.4.1.2. Regular Exp
II.2.2.2. Right Linear

II.4.1. Left Linear Grammars vs NFAs (13.7)

II.4.1.2. Translating NFAs into Regular Expressions (13.10)

II.4.1.3. Equivalence Theorem

II.4.2. Closure Properties and Decidability of Regular Languages

II.4.3. The Pumping Lemma for Regular Languages (12.4, 12.5)
Closure Properties

Theorem (II.4.4.1.)

Regular languages are closed under
1. complement,
2. intersection,
3. the operation \( L \mapsto L^R \).

So if \( L, L' \) are regular languages over alphabet \( T \), so are
1. \( L^c \) (the complement of \( L \), i.e. \( \{ t \in T^* \mid t \notin L \} \)),
2. \( L \cap L' \),
3. \( L^R \) (i.e. \( \{ w^R \mid w \in L \} \), where \( w^R \) is the result of reverting \( w \)).

Furthermore, regular grammars for \( L^c, L \cap L' \) and \( L^R \) can be computed from those for \( L \).

Decision Problems

Theorem (II.4.4.3.)

- We can decide for regular languages whether \( L = \emptyset \).
- We can decide for regular languages \( L \) and \( L' \) whether \( L \subseteq L' \).
- We can decide for regular languages \( L \) and \( L' \) whether \( L = L' \).

Proof Idea for Theorem II.4.4.1.

- We will use that regular expressions, languages definable by DFAs and regular languages are equivalent.
- Languages definable by regular expressions are closed under \( L \mapsto L^R \).
  (See additional material, Lemma II.4.3.4).
- One can easily see that languages definable by a DFA are closed under \( L \mapsto L^c \) and under intersection.
- Therefore the assertion follows.
- Full details can be found in the additional material.

Proof of Theorem II.4.4.3.

A proof of Theorem II.4.4.3. can be found in the additional material.
Motivation

- We want to show that there are languages which are context-free but not regular.
- In order to do this we prove the pumping lemma, which uses the fact that an NFA has only finitely many states. (We could use as well the fact that a regular grammar has only finitely many nonterminals).

- **Note** The following slides contain some coloured parts. The colours are indistinguishable in the black and white handouts. It is recommended to look at them using the online version.

Using the Finiteness of an NFA

Consider an NFA

This NFA has 5 states.

Any run of the NFA for a word of length $\geq 5$ uses at least 6 states. Therefore it must visit one state at least twice. So there must be a loop within the first 5 letters of such a word.

Using the Finiteness of an NFA

Here is the run for the word $z = ababa = uvw$ using colours blue, red and green.

- The **blue part** is the part before we reached a state visited twice, corresponding to the word $u = a$.
- The **red part** is the part from the state visited twice until we reach it again, corresponding to the word $v = bab$.
- The **green part** is the remaining part, corresponding to the word $w = a$.
- The loop must occur within the first 5 letters, so $|uv| \leq 5$. Because $v$ is along a loop, $|v| \geq 1$. 
Using the Finiteness of an NFA

If we repeat the loop several times, we obtain as well an accepting run of the automaton.
- If we start with $u = a$, then repeat the loop following the word $v = bab$ $i$ times, then the follow the word $w = a$, we obtain an accepting run.
- It accepts the word $a(bab)^i a$.
  - E.g. in case $i = 2$ the word is $abababa$.
  - In case $i = 0$ the word is $aa$.
- In general we get that the word $uv^i w$ is an element of the language as well.

### Generalisation

Assume an NFA $A$ having $k$ states. Then for every word $x \in L(A)$ s.t. $|x| \geq k$ there exist words $u, v, w$ s.t.

$$x = uvw, |uv| \leq k, |v| \geq 1$$

and s.t.

$$uv^i w \in L(A) \text{ for all } i \in \mathbb{N}$$

This follows by the above considerations.

So we have proved the following theorem:

### Example 1

**Lemma**

The language $L := \{a^i b^i \mid i \geq 1\}$ is context-free but not regular.
II.4.3. The Pumping Lemma for Regular Languages (12.4, 12.5)

Proof (Example 1)

- We have already seen that $L$ is context-free.
- Assume $L$ is regular.
- Let $k$ be as in the pumping lemma.
- Consider $x := a^k b^k \in L$.
- $|x| \geq k$, so there exist $u, v, w$ s.t.
  \[ x = uvw, \ |uv| \leq k, \ |v| \geq 1, \]
  and s.t.
  \[ uv^i w \in L \text{ for all } i \in \mathbb{N}. \]
- Since $|uv| \leq k$, $u$ and $v$ are substrings of $a^k$.
- Therefore $uv^2w = a^{k+l} b^k$ where $l = |v|$.
- But $a^{k+l} b^k \notin L$, a contradiction.

Lemma

The language $L := \{xx^R \mid x \in \{a, b\}^*\}$ is context-free but not regular.

Proof (Example 2)

- We have already seen that $L$ is context-free.
- Assume $L$ is regular.
- Let $k$ be as in the pumping lemma.
- Consider $x := a^k bba^k \in L$.
- $|x| \geq k$, so there exist $u, v, w$ s.t.
  \[ x = uvw, \ |uv| \leq k, \ |v| \geq 1, \]
  and s.t.
  \[ uv^i w \in L \text{ for all } i \in \mathbb{N}. \]
- Since $|uv| \leq k$, $u$ and $v$ are substrings of $a^k$.
- Therefore $uv^2w = a^{k+l} bba^k$ where $l = |v|$.
- But $a^{k+l} bba^k \notin L$, a contradiction.