II.4.1. Equivalence Theorem for Regular Languages

II.4.1.1. Regular Grammars and NFAs (13.5)

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

II.4.1.3. Main Theorem

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 $G$. 
II.4.1. Equivalence Theorem for Regular Languages

II.4.1.1. Regular Grammars and NFAs (13.5)

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

II.4.1.3. Main Theorem

Proof of Theorem II.4.1.1

We show that $L(A) = L(G)$:

- Assume $w = a_1 \cdots a_n \in L(A)$.

Then there exists a sequence of transitions in $A$

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

or

$$S = A_0 \xrightarrow{a_1} A_1 \xrightarrow{a_2} A_2 \xrightarrow{a_3} \cdots \xrightarrow{a_n} A_n \in F$$

But from this we obtain derivations

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

or

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

Then there exists a sequence of transitions in $A$

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

or

$$S = A_0 \xrightarrow{a_1} A_1 \xrightarrow{a_2} A_2 \xrightarrow{a_3} \cdots \xrightarrow{a_n} A_n \in F$$

So $w \in L(G)$.

Proof of Theorem II.4.1.2

Theorem II.4.1.2

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 $A$. 

CS_275  Sect. II.4.1.2.  7/ 42

II.4.1.1. Regular Grammars and NFAs (13.5)

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

II.4.1.3. Main Theorem
Before proving Theorem II.4.1.2 we give an example:
Consider the following automaton for the language \( L = \? \)

We define regular expressions and simplify them at each intermediate step in order to keep them simple.

Let \( L_{q,q'}^\emptyset \) be the set of strings which allows us to get from \( q \) to \( q' \) with intermediate states in \( \emptyset \), i.e. without any intermediate states.
We define a regular expression \( E_{q,q'}^\emptyset \), s.t. \( L(E_{q,q'}^\emptyset) = L_{q,q'}^\emptyset \). We can define

- \( E_{q,q'}^\emptyset := a_1 \mid \cdots \mid a_n \),
  if \( q \neq q' \) and we have transitions \( q \xrightarrow{a_i} q' \),
- \( E_{q,q'}^\emptyset = a_1 \mid \cdots \mid a_n \mid \epsilon \),
  if \( q = q' \) and we have transitions \( q \xrightarrow{a_i} q' \).

**Calculation of \( L_{q,q'}^\emptyset \)**

Original automaton:

States with \( E_{q,q'}^\emptyset \):

\[
\begin{align*}
E_{q_0,q_0}^\emptyset &= 1 \mid \epsilon \\
E_{q_0,q_1}^\emptyset &= 0 \\
E_{q_1,q_0}^\emptyset &= \emptyset \\
E_{q_1,q_1}^\emptyset &= 0 \mid 1 \mid \epsilon
\end{align*}
\]
From $E_{q,q'}^0$ to $E_{q,q'}^{q_0}$

States with $E_{q,q'}^0$:

Let $L_{q,q'}^{q_0}$ be the set of strings which allows us to get from $q$ to $q'$ with intermediate states in $\{q_0\}$.

We define $E_{q,q'}^{q_0}$ s.t. $L(E_{q,q'}^{q_0}) = L_{q,q'}^{q_0}$:

$$E_{q,q'}^{q_0} = E_{q,q'}^0 \cup (E_{q,q}^0 (E_{q_0,q_0}^0)^* E_{q_0,q'}^0)$$

Calculation of $E_{q,q'}^{q_0}$:

$$E_{q,q'}^{q_0} = E_{q,q'}^0 \cup (E_{q,q}^0 (E_{q_0,q_0}^0)^* E_{q_0,q'}^{q_0})$$

From $E_{q,q'}^{q_0}$ to $E_{q,q'}^{q_0, q_1}$

States with $E_{q,q'}^{q_0}$:

Let $L_{q,q'}^{q_0, q_1}$ be the set of strings which allows us to get from $q$ to $q'$ with intermediate states in $\{q_0, q_1\}$.

We define $E_{q,q'}^{q_0, q_1}$ s.t. $L(E_{q,q'}^{q_0, q_1}) = L_{q,q'}^{q_0, q_1}$:

$$E_{q,q'}^{q_0, q_1} = E_{q,q'}^{q_0} \cup (E_{q,q}^{q_0} (E_{q_1,q_1}^{q_0})^* E_{q_1,q'}^{q_0})$$
II.4.2. Translating NFAs into Regular Expressions

The Language of $A$: $L(A)$

States with $E_{q_0, q_1}$:

- $L(E_{q_0, q_1})$ is the set of strings which allows us to get from $q_0$ to $q_1$ using any intermediate states.
- The language $L(A)$ is the set of strings which allow us to get from $q_0$ to any accepting state.
- In the example there is only one accepting state ($q_1$), so the language accepted by $A$ is the language given by

$$E_{q_0, q_1}^{q_0, q_1} = 1^*0(0 \mid 1)^*$$

From $E_{q_0, q_1}$ to $E_{q_0, q_0}$

States with $E_{q_0, q_0}$:

- Let $A'$ be as $A$, but with additional accepting state $q_0$, then we get that $L(A')$ is given by

$$E_{q_0, q_1}^{q_0, q_1} \mid E_{q_0, q_0}^{q_0, q_0} = 1^*0(0 \mid 1)^* = (0 \mid 1)^*$$
Proof of Theorem II.4.1.2

Let for states \( q, q' \) of \( A \)

\[
L_{q,q'} := \{ w \in T^* \mid q \xrightarrow{w} q' \}
\]

We construct for states \( q, q' \) of \( A \) a regular expression \( E_{q,q'} \) s.t.

\[
L(E_{q,q'}) = L_{q,q'}
\]

If \( F = \{ q_1, \ldots, q_k \} \) then we obtain

\[
L(A) = L_{q_0,q_1} \mid \cdots \mid L_{q_{k-1},q_k} = L(E_{q_0,q_1} \mid E_{q_0,q_2} \mid \cdots \mid E_{q_0,q_k})
\]

(If \( F \) is empty, then \( L(A) = L(\emptyset) \)).

Base case \( k = 0 \):

Let \( a_1, \ldots, a_k \) be the \( a_i \) s.t. \( q \xrightarrow{a_i} q' \). Then

\[
E_{q,q'}^\emptyset := \begin{cases} a_1 \mid \cdots \mid a_k & \text{if } q \neq q' \\ a_1 \mid \cdots \mid a_k \mid \epsilon & \text{if } q = q' \end{cases}
\]

(in case of \( k = 0 \) we have \( E_{q,q'}^\emptyset = \emptyset \) or \( = \epsilon \)).

Proof of Theorem II.4.1.2

We define regular expressions \( E_{q,q'} \) in stages by referring to \( E_{q_1,\cdots,q_i} \), s.t.

\[
L(E_{q_1,\cdots,q_i}) = L_{q_1,\cdots,q_i} := \{ a_1 \cdots a_k \in T^* \mid \exists p_i \in \{ q_1, \ldots, q_i \}. q \xrightarrow{a_1} p_1 \xrightarrow{a_2} p_2 \cdots \xrightarrow{a_{k-1}} p_{k-1} \xrightarrow{a_k} q' \}
\]

So \( L_{q_1,\cdots,q_i} \) is the set of words which allow us to get from \( q \) to \( q' \) by using as intermediate states \( q_1, \ldots, q_i \) only.

We define \( E_{q,q'}^{q_1,\cdots,q_k} \) by induction on \( k \).

Then we can define \( E_{q,q'} := E_{q,q'}^Q \).
II.4.1. Equivalence Theorem for Regular Languages

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

Proof of Theorem II.4.1.2

So we have

\[ q \xrightarrow{v} q_k \xrightarrow{w_1} q_k \xrightarrow{w_2} q_k \xrightarrow{\cdots} q_k \xrightarrow{w_j} q' \]

where \( j = 0 \) is possible, all intermediate transitions avoid \( q_k \) and \( w = vw_1 w_2 \cdots w_k v' \).

II.4.1.3. Main Theorem

In the second part we have

\( v \in L_{q,q}^{q_1,\cdots,q_k-1} \)
\( w_i \in L_{q_i,q_i}^{q_1,\cdots,q_k-1} \)
\( v' \in L_{q_k,q_k}^{q_1,\cdots,q_k-1} \)

Therefore \( w = vw_1 \cdots w_k v' \in L_{q,q}^{q_1,\cdots,q_k-1}(L_{q_1,q_k}^{q_1,\cdots,q_k-1})^*L_{q_k,q_k}^{q_1,\cdots,q_k-1} \).

Therefore

\[ L_{q,q'}^{q_1,\cdots,q_k-1} \supseteq L_{q,q'}^{q_1,\cdots,q_k-1} \]
Theorem II.4.1.3

**Theorem (II.4.1.3)**

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

We are going to show that

- languages definable by regular expressions,
- languages definable by regular grammars,
- languages definable by NFAs with empty moves,
- languages definable by NFAs,
- languages definable by DFAs

all define the same language.

We already have everything in order to show the above with regular grammars restricted to right-linear grammars.

Lemma II.4.1.4

**Lemma (II.4.1.4)**

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

1. $L$ is definable by a regular expression.
2. $L$ is definable by a right-linear grammar.
3. $L$ is definable by an NFA with empty moves.
4. $L$ is definable by an NFA.
5. $L$ 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 Lemma II.4.1.4

- $(1) \rightarrow (2)$ was shown in II.2.2.2.
  - (Finite languages are definable regular grammars. Languages definable by regular grammars are closed under the operations for forming regular expressions).
- $(2) \rightarrow (4)$ was shown in Theorem II.4.1.1
  - Right-linear grammars can be simulated by an NFA.
- $(4) \rightarrow (1)$ was shown in Theorem II.4.1.2
  - We can determine the language between states of an NFA as a regular expression.
- So $(1), (2), (4)$ are equivalent.
II.4.1. Equivalence Theorem for Regular Languages

II.4.1.3. Main Theorem

Proof of Lemma II.4.1.4

- (3) → (4) was shown in Theorem II.3.4.1.
  - We can omit the empty moves in NFAs with empty moves.
- (4) → (5) was shown in Theorem II.3.5.1.
  - NFAs can be translated into DFAs using as states sets of states.
  - (5) → (4) → (3) are trivial.
  - DFAs are special cases of NFAs.
  - NFAs are special cases of NFAs with empty moves.
  - So (3), (4), (5) are equivalent.
  - So (1), (2), (3), (4), (5) are equivalent.

Equivalence of Left-Linear and Right-Linear Grammars

- It remains to show that left-linear and right-linear grammars are equivalent.
- This is shown as follows:
  - The languages derived by left-linear grammars are $L^R$ for languages derived by right-linear grammars.
  - Regular Expressions are closed under the reverse operation, i.e. if $L$ is definable by a regular expression, so is $L^R$.
  - Therefore as well right-linear grammars are closed under the reverse operation, therefore left-linear and right-linear grammars are equivalent.

Proof of Lemma II.4.1.5

We prove only (1), (2) is analogously.
Let $G$ be a left-linear grammar with alphabet $T$, nonterminals $N$ and start symbol $S$.
Let $G'$ be identical to $G$ but with rules

$$ B \rightarrow aC $$

$(B, C \in N, a \in T)$ replaced by

$$ B \rightarrow Ca $$

$G'$ is right-linear. Further it follows immediately for any $w \in (N \cup T)^*$ that

$$ S \Rightarrow_G w \quad \text{iff} \quad S \Rightarrow_{G'} w^R $$

Right-Linear Languages are the Reverse of Left-Linear Ones

Lemma (II.4.1.5)

1. Let $G$ be a left-linear grammar.
   Then there exist a right-linear grammar $G'$ over the same alphabet
   s.t. $L(G) = L(G'^R)$.
   $G'$ can be computed from $G$.

2. Let $G$ be a right-linear grammar.
   Then there exist a left-linear grammar $G'$ over the same alphabet
   s.t. $L(G) = L(G'^R)$.
   $G'$ can be computed from $G$. 
II.4.1. Equivalence Theorem for Regular Languages

II.4.1.3. Main Theorem

Proof of Lemma II.4.1.5

Therefore

\[ L(G') = \{ w \in T^* \mid S \Rightarrow_G' w \} = \{ w^R \in T^* \mid S \Rightarrow_G w \} = \{ w \in T^* \mid S \Rightarrow_G w^R \} = L(G)^R \]

Lemma (II.4.1.6)

1. For every regular expression \( E \) there exists a regular expression \( E^R \) s.t. \( L(E^R) = L(E)^R \).
   \( E^R \) can be computed from \( E \).

2. Similarly for every language \( L \) definable by a right-linear grammar \( G \) there exists a right-linear grammar \( G^R \) defining \( L^R \).
   \( G^R \) can be computed from \( G \).

Proof of Lemma II.4.1.6

(1) We show the existence of \( E^R \) by induction on \( E \):
   - For \( E = \emptyset \), \( E = \epsilon \) or \( E = a \) \( L(E)^R = L(E) \), so define \( E^R := E \).
   - For \( E = E_1 \mid E_2 \) we have define \( E^R = E_1^R \mid E_2^R \).
   - For \( E = E_1 E_2 \) define \( E^R = E_2^R E_1^R \).
   - For \( E = E_1^* \) define \( E^R = (E_1^R)^* \).

(2) Follows since languages definable by right-linear grammars are exactly the languages definable by regular expressions.
Proof of Lemma II.4.1.7

- Assume $L = L(G)$ for a left-linear grammar $G$.
  - Then $L^R = L(G')$ for a right-linear grammar $G'$.
  - Right-linear grammars are closed under $L \mapsto L^R$.
  - Therefore there exists a right-linear grammar $G''$ s.t.
    $L(G'') = (L(G'))^R = (L^R)^R = L$.

- Assume $L = L(G)$ for a right-linear grammar $G$.
  - There exists a right-linear grammar $G'$ s.t. $L(G') = L^R$.
  - There exists a left-linear grammar $G''$ s.t. $L(G'') = L(G')^R$.
  - Now $L(G'') = L(G')^R = (L(G)^R)^R = L(G)^R = L$.

By the above.