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)

Proof of Theorem II.4.1.1

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

1. 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

$$S = A_0 \rightarrow a_1 A_1 \rightarrow a_1 a_2 A_2 \rightarrow \cdots \rightarrow a_1 a_2 \cdots a_n A_n \rightarrow a_1 a_2 \cdots a_n = w$$

or

$$S = A_0 \rightarrow a_1 A_1 \rightarrow a_1 a_2 A_2 \rightarrow \cdots \rightarrow a_1 a_2 \cdots a_n \in F$$

So $w \in L(G)$.

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

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.4.1.1. 5/42

CS,275 Sect. II.4.1.2. 6/42
Example

Before proving Theorem II.4.1.2 we give an example:
Consider the following automaton for the language $L = \emptyset$:

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

Calculation of $L_{q,q'}^\emptyset$

Original automaton:

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 $q \xrightarrow{a_i} q'$.

States with $E_{q,q'}^\emptyset$:
II.4.1.2. Translating NFAs into Regular Expressions

Let \( L_{q,q'} \) 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.

\[
E_{q,q'}^{q_0} = E_0^{q_0} | (E_0^{q_0} (E_0^{q_0,q_0})^* E_0^{q_0,q_0'})
\]

From \( E_{q,q'}^{q_0} \) to \( E_{q,q'}^{q_0,q_1} \)

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

We define \( E_{q,q'}^{q_0,q_1} \), s.t.

\[
E_{q,q'}^{q_0,q_1} = E_0^{q_0} | (E_0^{q_0} (E_0^{q_0,q_0})^* E_0^{q_0,q_0'})
\]
Calculation of $E_{q_0, q_1}^{q_0, q_1}$

$E_{q_0, q_1}^{q_0, q_1}$:

$\begin{align*}
E_{q_0, q_1}^{q_0, q_0} &= 1^* (1^*0(0 | 1 | \epsilon)^*0) \\
&= 1^* \\
E_{q_0, q_1}^{q_0, q_1} &= (1^*0) (1^*0(0 | 1 | \epsilon)^*(0 | 1 | \epsilon)) \\
&= 1^*0(0 | 1)^* \\
E_{q_0, q_1}^{q_0, q_1} &= \emptyset ((0 | 1 | \epsilon)(0 | 1 | \epsilon)^*0) \\
&= \emptyset \\
E_{q_0, q_1}^{q_0, q_1} &= (0 | 1 | \epsilon) ((0 | 1 | \epsilon)(0 | 1 | \epsilon)^*(0 | 1 | \epsilon)) \\
&= (0 | 1)^*
\end{align*}$

States with $E_{q_0, q_1}^{q_0, q_1}$:

\[
\begin{array}{c}
q_0 \\
\cup \\
q_1 \\
\cup \\
\emptyset \\
\cup \\
(0 | 1)^*
\end{array}
\]

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

States with $E_{q_0, q_1}^{q_0, q_1}$:

\[
\begin{array}{c}
q_0 \\
\cup \\
q_1 \\
\cup \\
\emptyset \\
\cup \\
(0 | 1)^*
\end{array}
\]

$\bullet$ $L(E_{q_0, q_1}^{q_0, q_1})$ is the set of strings which allows us to get from $q_0$ to $q_1$

$\bullet$ The language $L(A)$ is the set of strings which allow us to get from $q_0$ to any accepting state.

$\bullet$ In the example there is only one accepting state ($q_1$), so the language accepted by $A$ is the language given by

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

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

States with $E_{q_0, q_1}^{q_0, q_1}$:

\[
\begin{array}{c}
q_0 \\
\cup \\
q_1 \\
\cup \\
\emptyset \\
\cup \\
(0 | 1)^*
\end{array}
\]

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

States with $E_{q_0, q_1}^{q_0, q_1}$:

\[
\begin{array}{c}
q_0 \\
\cup \\
q_1 \\
\cup \\
\emptyset \\
\cup \\
(0 | 1)^*
\end{array}
\]

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

$E_{q_0, q_1}^{q_0, q_1} | E_{q_0, q_1}^{q_0, q_1} = 1^*0(0 | 1)^*$
II.4.1.2. Translating NFAs into Regular Expressions (13.10)

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 $L(A) = L(\emptyset)$).

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

$$L(E_{q_1,\ldots,q_i}) = L_{q_1,\ldots,q_i} := \{ a_1 \ldots a_i \in T^* \mid \exists p_i \in \{q_1, \ldots, q_i\}. q \xrightarrow{a_i} p_i \xrightarrow{a_{i-1}} p_{i-1} \ldots p_2 \xrightarrow{a_1} q' \}$$

So $L_{q_1,\ldots,q_i}$ 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 $E_{q,q'}^{q_1,\ldots,q_k}$ by induction on $k$.

Then we can define $E_{q,q'} := E_{q,q'}^q$.

Induction Step: Assume we have defined $E_{p,p'}^{q_1,\ldots,q_{k-1}}$ for all $p, p' \in Q$.

We define $E_{q,q'}^{q_1,\ldots,q_{k-1}}$.

A transition $q \xrightarrow{w} q'$ which uses only intermediate states $q_1, \ldots, q_k$ can have two forms:

- Either we don’t use $q_k$ as an intermediate state.
  So we have only intermediate states $q_1, \ldots, q_{k-1}$ and have
  $w \in L_{q_1,\ldots,q_{k-1}}$.

- Or we reach $q_k$ as an intermediate state. We single out
  - the first part of the transition which doesn’t use state $q_k$ until one
    reaches for the first time as an intermediate state $q_k$ (note that $q = q_k$
    or $q' = q_k$ is possible)
  - the second part where we several times go from $q_k$ to $q_k$ with
    intermediate states $\neq q_k$,
  - and the last part where we get from $q_k$ to $q'$ without using $q_k$. 
Proof of Theorem II.4.1.2

So we have
\[ q \rightarrow q_k \xrightarrow{w_1} q_k \xrightarrow{w_2} q_k \cdots \xrightarrow{w_j} q_k \xrightarrow{v'} q' \]
where \( j = 0 \) is possible, all intermediate transitions avoid \( q_k \) and \( w = vw_1w_2 \cdots w_kv' \).

In the second part we have
- \( v \in L^{q_1, \ldots, q_{k-1}}_{q, q'} \)
- \( w_i \in L^{q_{i+1}, q_{i+1}}_{q, q'} \)
- \( v' \in L^{q_k, q'}_{q, q'} \)
- Therefore \( w = vw_1 \cdots w_kv' \in L^{q_1, \ldots, q_{k-1}}_{q, q_k}(L^{q_1, \ldots, q_{k-1}}_{q, q_k})^*L^{q_1, \ldots, q_{k-1}}_{q, q'} \)
- One can see easily as well that for an element \( w \) in the right hand side we can derive that \( w \) is in the left hand side as well, i.e.
\[ L^{q_1, \ldots, q_k}_{q, q'} \subseteq L^{q_1, \ldots, q_{k-1}}_{q, q_k} | (L^{q_1, \ldots, q_{k-1}}_{q, q_k})^*L^{q_1, \ldots, q_{k-1}}_{q, q_k} \]

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
II.4.1. Equivalence Theorem for Regular Languages

II.4.1.3. Main Theorem

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.
- 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.

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 $G$.

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

\[ S \Rightarrow_G w \text{ iff } S \Rightarrow_{G'} w^R \]
II.4.1. Equivalence Theorem for Regular Languages

II.4.1.3. Main Theorem

Proof of Lemma II.4.1.5

Therefore

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

II.4.1. Equivalence Theorem for Regular Languages

II.4.1.3. Main Theorem

Regular Expressions Closed Under \( L \mapsto L^R \)

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 \( G \).

II.4.1. Equivalence Theorem for Regular Languages

II.4.1.3. Main Theorem

Left-Linear and Right-Linear Grammars are Equivalent

Lemma (II.4.1.7)

Let \( L \) be a language over an alphabet \( T \). The following are equivalent:

1. \( L = L(G) \) for a left-linear grammar \( G \).
2. \( L = L(G) \) for a right-linear grammar \( G \).

The left-linear and right-linear grammars can be computed from each other.

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 \( E = E_1E_2 \) 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 \( L \rightarrow L^R \).
  - 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 \( L(G'') = L(G')^R = (L(G)^R)^R = L(G) = L \).

By the above.