Theoretische Informatik: Endliche Automaten, Formale Sprachen und Grammatiken

Marko Livajusic

10. Oktober 2024

Inhaltsverzeichnis

1 Q3.2: Deterministische Endliche Automaten (DEAs)

1.1 Transduktor

1.2 Akzeptor

1.3 Mealy- und Mooreautomat (irrelevant)

1.4 Minimerung von DEAs

2 Q3.2: Nichtdeterministische Endliche Automaten (NEAs)

2.1 $\epsilon-$NEAs

2.1.1 ϵ-NEA $\rightarrow$ NEA (6 Konstruktionsregeln)

2.1.2 ϵ-NEA $\rightarrow$ DEA (Potenzmengenkonstruktion)

2.2 NEA $\rightarrow$ DEA (Potenzmengenkonstruktion)

3 Q3.2: RegEx

3.1 RegEx $\rightarrow$ ϵ-NEA

3.2 Formale Sprachen

3.2.1 Reguläre Sprachen

Eine Sprache L ist dann *regulär*, wenn diese sich darstellen lässt mithilfe eines:

1. deterministischen endlichen Automatens
2. regulären Ausdrucks

4 Q3.2: Grammatiken

Eine Grammatik G ist ein 4-Tupel $G = \{N, T, P, S\}$, wobei

- N das **Nichtterminalalphabet**
- T das **Terminalalphabet**
- P die **Produktionen**
- S das **Startsymbol** ist.

4.1 Typ 3 Grammatik (regulär)

Eine Grammatik G ist dann *regulär*, wenn in den Produktionen P

- links ein Nichtterminal und rechts ein oder mehrere Terminale vorkommen gefolgt von maximal einem Nichtterminal

5 Ableitung

Gegeben sei folgende Grammatik:

$$
\begin{aligned}
&T = \{x, y, z\} \\
&N = \{S, M, A, V\} \\
&P = \{ \\
&\quad S \rightarrow A|M|V \\
&\quad A \rightarrow (S + S) \\
&\quad M \rightarrow (S \cdot S) \\
&\quad V \rightarrow x|y|z \\
&\quad \}
\end{aligned}
$$

Wie wird das Wort $(x \cdot (y + z))$ gebildet?

$$
\begin{aligned}
S \Rightarrow M \Rightarrow (S \cdot S) \\
\Rightarrow (v \cdot S) \Rightarrow (x \cdot S) \Rightarrow (x \cdot A) \Rightarrow \\
(x \cdot (S + S)) \Rightarrow (x \cdot (v + S)) \Rightarrow (x \cdot (y + S)) \Rightarrow (x \cdot (y + v)) \Rightarrow (x \cdot (y + z))
\end{aligned}
$$

5.1 Ableitungsbaum

Dies kann man auch mit einem Ableitungsbaum darstellen:

6 Kontextfreie Sprachen

Gegeben sei folgende kontextfreie Grammatik:

$$
\begin{aligned}
&N = \{A, B, S\}\\
&T = \{a, b, \epsilon\}\\
&S = S\\
&P = \{\\
&\quad S \to AB\\
&\quad S \to ABA\\
&\quad A \to aA\\
&\quad A \to a\\
&\quad B \to Bb\\
&\quad B \to \epsilon\\
&\}
\end{aligned}
$$

6.1 Chomsky-Normalform (klausurrelevant, abitur-irrelevant)

6.1.1 1. ϵ-Elimination

Zuerst wird $B \to \epsilon$ entfernt. Die aktualisierte Grammatik lautet:

$$
\begin{aligned}
&N = \{A, B, S\}\\
&T = \{a, b, \epsilon\}\\
&S = S\\
&P = \{\\
&\quad S \to AB\\
&\quad \mathbf{S} \to \mathbf{A}\\
&\quad \mathbf{S} \to \mathbf{AA}\\
&\quad A \to aA\\
&\quad A \to a\\
&\quad B \to b\\
&\}
\end{aligned}
$$

6.1.2 2. Elimination von Kettenregeln

Die Kettenregeln, d.h. überall da, wo ein Nichtterminal auf ein anderes Nichtterminal folgt, d.h. $S \to A$, werden entfernt.

$$
\begin{aligned}
N &= \{A, B, S\} \\
T &= \{a, b, \epsilon\} \\
S &= S \\
P = \{ & \\
& S \to AB \\
& S \to AA \\
& A \to aA \\
& A \to a \\
& B \to b \\
& \}
\end{aligned}
$$

6.1.3 3. Separation von Terminalzeichen

Jedes Terminal wird durch ein Nichtterminal ersetzt:

$$
\begin{aligned}
N &= \{A, B, S\} \\
T &= \{a, b, \epsilon\} \\
S &= S \\
P = \{ & \\
& S \to AB \\
& S \to aA \quad | \ V_a = a \\
& S \to V_a A \\
& S \to a \\
& S \to ABA \\
& S \to AA \\
& A \to a \\
& B \to BV_b \\
& B \to b \\
& V_a \to a \\
& V_b \to b \\
& \}
\end{aligned}
$$

6.1.4 4. Elimination von mehrelementigen Nonterminalketten

In diesem Schritt wird die Anzahl von Nichtterminalen auf 2 reduziert, d.h. $S \to ABA$ wird zu $S \to S_2A$, wobei S_2 als $S_2 \to AB$ definiert wird.

$$
\begin{aligned}
N &= \{A, B, S\} \\
T &= \{a, b, \epsilon\} \\
S &= S \\
P &= \{ \\
&S \to AB \\
&S \to V_aA \\
&S \to a \\
&\mathbf{S \to S_2A} \\
S_2 \to AB&S \to AA \\
&A \to a \\
&B \to BV_b \\
&B \to b \\
&V_a \to a \\
&V_b \to b \\
&\}
\end{aligned}
$$

6.2 CYK-Algorithmus (klausurrelevant, abitur-irrelevant)

Mit dem CYK-Algorithmus lässt sich sagen, ob ein Wort ω in einer kontextfreien Sprache liegt. Voraussetzung für den CYK-Algorithmus ist die Chomsky-Normalform (6.1).