Technische Informatik

Vorlesungsskript

Mitschrift von Falk-Jonatan Strube

Vorlesung von Dr. Boris Hollas

16. März 2016

Inhaltsverzeichnis

Inhalte

Grundlage: Grundkurs Theoretische Informatik [1]

- Formale Sprachen
  - Reguläre Sprachen
    - Endliche Automaten
    - Reguläre Ausdrücke
  - Nichtreguläre Sprachen
  - Kontextfreie Sprachen
    - Kellerautomaten
    - Grammatiken
- Berechenbarkeit
  - Halteproblem
- Komplexitätsklassen
  - P
  - NP
  - NP-vollständige Probleme

1 Automaten und Formale Sprachen

1.1 Formale Sprachen

Def.: Ein Alphabet ist eine Menge $\Sigma \neq \emptyset$ (Symbole in Σ – müssen nicht einzelne Buchstaben sein, auch Wörter usw. [bspw. „if“ oder „else“ im Alphabet der Programmiersprache C]).

Def.: Für $w_1, ..., w_n \in \Sigma$ ist $w = w_1 ... w_n$ ein Wort der Länge n.
Σ^n beschreibt alle Worte mit der Länge genau n
Das Wort ε ist das *leere Wort*.
Die Menge aller Wörter bezeichnen wir mit Σ^* (einschließlich dem leeren Wort).

Bsp.: $\Sigma = \{a, b, c\} \quad \rightarrow \Sigma^* = \{\varepsilon, a, b, c, aa, ab, ac, aaa, ...\}$

Def.: Für Wörter $a, b \in \Sigma^*$ ist ab die Konkatenation dieser Wörter.
Für ein Wort w ist w^n die n-fache Konkatenation von w, wobei $w^0 = \varepsilon$.

Bemerkung: Für alle $w \in \Sigma^*$ gilt $\varepsilon w = w = w\varepsilon$. ε ist also das neutrale Element der Konkatenation.

Def.: Eine *formale Sprache* ist eine Teilmenge von σ^*.

Def.: Für Sprachen A, B ist $AB = \{ab \mid a \in A,\ b \in B\}$ sowie $A^n = \prod_{i=1}^{n} A$, wobei $A^0 = \{\varepsilon\}$.

Bemerkung: $\emptyset, \varepsilon, \{\varepsilon\}$ sind unterschiedliche Dinge (leere Menge, leeres Wort, Menge mit leerem Wort).

Bemerkung: Σ^* lässt sich ebenfalls definieren durch $\Sigma^* = \bigcup\limits_{n \geq 0} \Sigma^n$.
Ferner ist $\Sigma^+ = \Sigma^* - \{\varepsilon\}$.

1.2 Reguläre Sprachen

1.2.1 Deterministische endliche Automaten (DFA)

Bsp.:
ABB1
(Pfeil zeigt auf Startzustand, Endzustand ist doppelt umrandet)
Dieser DFA akzeptiert alle Wörter über $\Sigma = \{a, b, c\}$, die abc enthalten.
Deterministisch: Es gibt genau ein Folgezustand. Von jedem Knoten aus gibt es genau eine Kante für jedes Zeichen, nicht mehrere und nicht keine.

Bsp.:
ABB2
Dieser DFA erkennt die Sprache $\{a, b, c\}^*$.

Def.: Ein DFA ist ein Tupel $\mathcal{M} = (Z, \Sigma, \delta, z_0,)$

- Z: Menge der Zustände
- Σ: Eingabealphabet
- δ: Überführungsfunktion $Z \times \Sigma \to Z$. Dabei bedeutet $\delta(z, a) = z'$, dass $\mathcal{M}$ im Zustand z für das Zeichen a in den Zustand z' wechselt.
- $z_0 \in Z$: Startzustand
- E: Menge der Endzustände

δ: ABB3

Def.: Die erweiterte Überführungsfunktion $\hat{\delta} : Z \times \Sigma^* \to Z$ ist definiert durch

$$\hat{\delta}(z, w) = \begin{cases} z & \text{für } w = \varepsilon \\ \hat{\delta}(\delta(z, a), x) & \text{für } w = ax \text{ mit } a \in \Sigma, x \in \Sigma^* \end{cases}$$

Dazu vergleichbarer C-Code:

```
int δ̂(int z, char* w){
  if ( strlen(w) == 0 )
    return z;
  else
    return δ̂(δ(z, w[0]), w[1]);
```

Veranschaulichung:
ABB4
Die erweiterte Überführungsfunktion bestimmt den Zustand nach dem vollständingen Lesen eines Wortes.

Bsp.:
ABB5

$$
\begin{aligned}
\hat{\delta}(z_0, aaba) &= \hat{\delta}(\delta(z_0, a), aba) = \\
\hat{\delta}(z_0, aba) &= \hat{\delta}(\delta(z_0, a), ba) = \\
\hat{\delta}(z_0, ba) &= \hat{\delta}(\delta(z_0, b), a) = \\
\hat{\delta}(z_E, a) &= \hat{\delta}(\delta(z_E, a), \varepsilon) = \\
\hat{\delta}(z_E, \varepsilon) &= z_E
\end{aligned}
$$

Die von $\mathcal{M}$ *akzeptierte Sprache* ist $L(\mathcal{M}) = \{w \in \Sigma^* \mid \hat{\delta}(z_0, w) \in E\}$

Literatur

[1] Boris Hollas. *Grundkurs Theoretische Informatik: Mit Aufgaben und Anwendungen*. Springer Berlin, 2015.