Satz 1.2.1 Eine Sprache ist genau dann Typ-2-Sprache, wenn sie von einer ε-produktionsfreien Typ-2-Grammatik erzeugt wird.

Es gilt:

$$\{\text{alle Sprachen}\} \supsetneqq \{\text{Typ-0-Spr.}\} \supseteq \{\text{Typ-1-Spr.}\} \supseteq \{\text{Typ-2-Spr.}\} \supseteq \{\text{Typ-3-Spr.}\}.$$

Wortproblem: Unter dem ***Wortproblem*** versteht man folgende Fragestellung:

> Gibt es einen Algorithmus, der bei Eingabe einer Grammatik $G = (V, \Sigma, P, S)$ und eines Wortes $w \in \Sigma^*$ entscheidet, ob $w \in \mathcal{L}(G)$ ist oder nicht?

Algorithmus zur Entscheidung des Wortproblems für Typ-1-Sprachen

Eingabe: Grammatik $G = (V, \Sigma, P, S)$, $w \in \Sigma^*$;

```
n := |w|;
IF n = 0 THEN T := {u | S → u Regel von G}
ELSE T := ∅;
    T_neu := {S};
    WHILE w ∉ T AND T ≠ T_neu DO
        T := T_neu;
        T_neu := T ∪ {u ∈ (V ∪ Σ)* | |u| ≤ n und es gibt u' ∈ T mit u' ⇒ u}
    ENDDO
ENDIF
```

Ausgabe: "Ja" (d.h.: $w \in \mathcal{L}(G)$), falls $w \in T$, sonst "Nein".

1.3 Reguläre Sprachen, endliche Automaten und reguläre Ausdrücke

Grammatik: erzeugt Zeichenreihen ausgehend von Startvariable.

Automat: "Maschine", die auf ein Eingabewort $w \in \Sigma^*$ angesetzt wird, dieses liest (zeichenweise) und "akzeptiert" ("erkennt") oder nicht.

Definition. Ein ***deterministischer endlicher Automat*** (***deterministic finite automaton***, ***DFA***) $M = (Z, \Sigma, \delta, z_0, E)$ ist gegeben durch:

- eine endliche Menge Z von ***Zuständen***,
- ein Alphabet Σ (***Eingabealphabet***),
- eine totale Funktion $\delta : Z \times \Sigma \to Z$ (***Überführungsfunktion, Zustandsübergangsfunktion, Transitionsfunktion***),

- $z_0 \in Z$ (***Startzustand, Anfangszustand***),
- $E \subseteq Z$ (Menge der ***Endzustände, akzeptierenden Zustände***).

Sei $M = (Z, \Sigma, \delta, z_0, E)$ ein DFA. Rekursive Definition der Funktion $\hat{\delta} : Z \times \Sigma^* \to Z$:

1. $\hat{\delta}(z, \varepsilon) = z$,
2. $\hat{\delta}(z, aw) = \hat{\delta}(\delta(z, a), w) \quad (a \in \Sigma,\, w \in \Sigma^*)$.

Es gilt:

1. $a \in \Sigma \implies \hat{\delta}(z, a) = \delta(z, a)$.
2. $a_1, \dots, a_n \in \Sigma \implies \hat{\delta}(z, a_1 \dots a_n) = \delta(\dots\delta(\delta(z, a_1), a_2), \dots, a_n)$.
3. $u, v \in \Sigma^* \implies \hat{\delta}(z, uv) = \hat{\delta}(\hat{\delta}(z, u), v)$

Definition. Sei $M = (Z, \Sigma, \delta, z_0, E)$ ein DFA. Die ***von M akzeptierte Sprache*** $\mathcal{L}(M) \subseteq \Sigma^*$ ist $\mathcal{L}(M) = \{w \in \Sigma^* \mid \hat{\delta}(z_0, w) \in E\}$.

Satz 1.3.1 Jede von einem DFA akzeptierte Sprache ist regulär.

Definition. Ein ***nichtdeterministischer endlicher Automat*** (***nondeterministic finite automaton***, ***NFA***) $M = (Z, \Sigma, \delta, z_0, E)$ ist gegeben durch:

- Z, Σ, z_0, E wie bei DFA,
- Zustandsübergangsfunktion $\delta : Z \times \Sigma \to \mathfrak{P}(Z)$.

Die Funktion $\hat{\delta} : \mathfrak{P}(Z) \times \Sigma^* \to \mathfrak{P}(Z)$ kann wieder rekursiv definiert werden:

1. $\hat{\delta}(Z', \varepsilon) = Z'$ für $Z' \subseteq Z$,
2. $\hat{\delta}(Z', aw) = \hat{\delta}(\bigcup_{z \in Z'} \delta(z, a), w)$ für $Z' \subseteq Z$, $a \in \Sigma$, $w \in \Sigma^*$.

Definition. Sei $M = (Z, \Sigma, \delta, z_0, E)$ ein NFA. Die ***von M akzeptierte Sprache*** $\mathcal{L}(M) \subseteq \Sigma^*$ ist $\mathcal{L}(M) = \{w \in \Sigma^* \mid \hat{\delta}(\{z_0\}, w) \cap E \neq \emptyset\}$.

Bemerkungen

1. Statt Funktion $\delta : Z \times \Sigma \to \mathfrak{P}(Z)$ oft auch ***Übergangsrelation*** $\delta' \subseteq Z \times \Sigma \times Z$; dabei: $(z, a, z') \in \delta' \iff z' \in \delta(z, a)$.
2. Statt Startzustand z_0 auch oft Menge von Startzuständen.