<Satz> $\rightarrow$ <Subjekt> <Prädikat> <Objekt>

<Subjekt> $\rightarrow$ <Artikel> <Attribut> <Substantiv>

<Artikel> $\rightarrow$ ε

<Artikel> $\rightarrow$ der

<Artikel> $\rightarrow$ die

<Artikel> $\rightarrow$ das

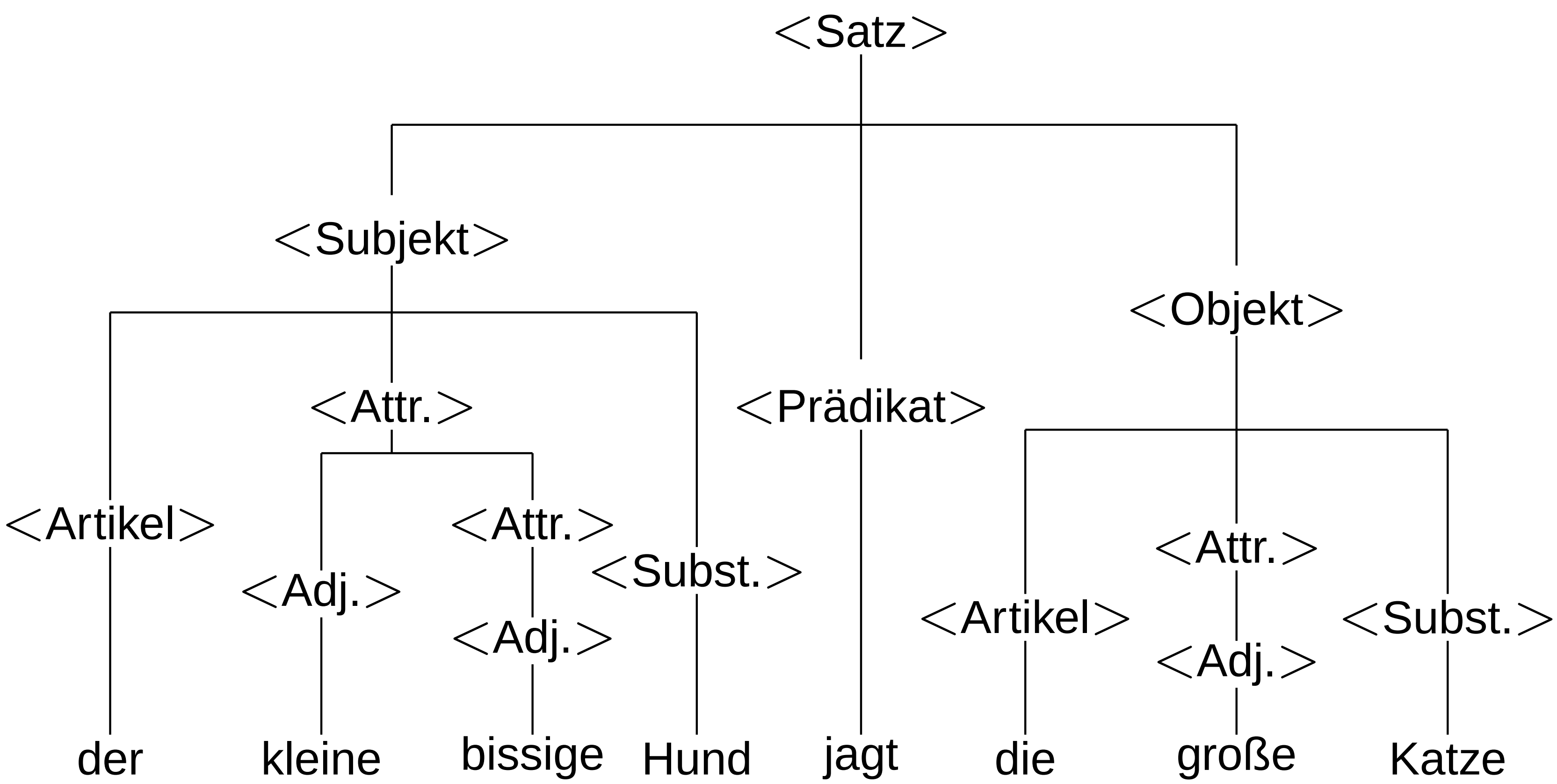
<Attribut> $\rightarrow$ ε

<Attribut> $\rightarrow$ <Adjektiv>

<Attribut> $\rightarrow$ <Adjektiv> <Attribut>

<Objekt> $\rightarrow$ <Artikel> <Attribut> <Substantiv>

<Adjektiv>	$\rightarrow$	kleine
<Adjektiv>	$\rightarrow$	bissige
<Adjektiv>	$\rightarrow$	große
<Substantiv>	$\rightarrow$	Hund
<Substantiv>	$\rightarrow$	Katze
<Prädikat>	$\rightarrow$	jagt


<Satz>
<Subjekt>
<Objekt>
<Attr.>
<Prädikat>
<Artikel>
<Attr.>
<Adj.>
<Subst.>
<Attr.>
<Adj.>
<Artikel>
<Subst.>
<Adj.>
der
kleine
bissige
Hund
jagt
die
große
Katze

Def: Eine **Grammatik** ist ein 4-Tupel $G = (V, \Sigma, P, S)$.

V endliche *Variablenmenge*

Σ Menge der Terminalsymbole $\Sigma \cap V = \emptyset$

P endliche Menge der *Regeln* oder *Produktionen*.

$P \subseteq (V \cup \Sigma)^+ \ \times \ (V \cup \Sigma)^*$.

$S \in V$ ist die *Startvariable*.

Die von G erzeugte *Sprache* ist

$$L(G) = \{w \in \Sigma^* \mid S \Rightarrow_G^* w\}$$

Chomsky-Hierarchie

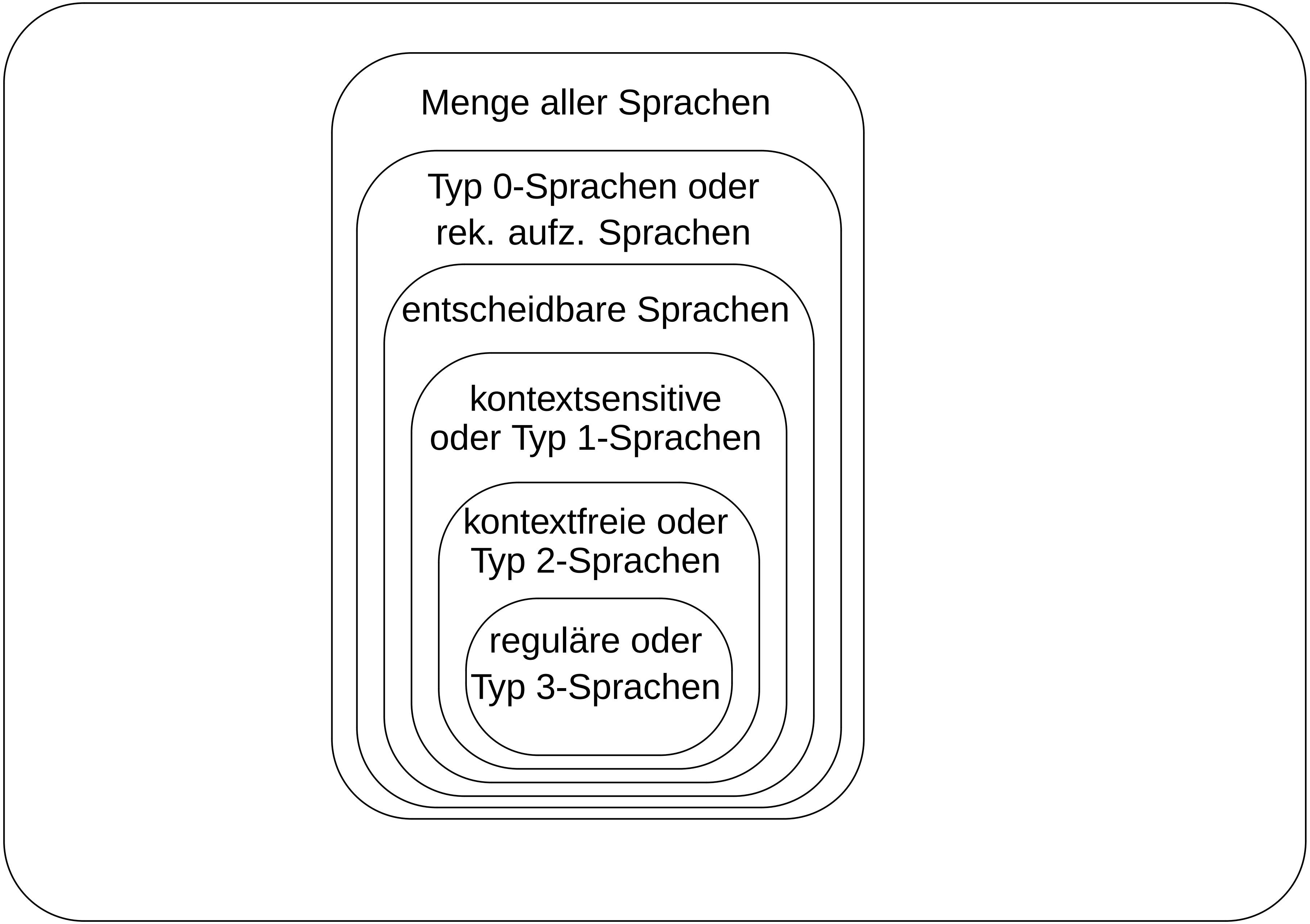
Def: Jede Grammatik ist zunächst automatisch vom *Typ 0*.

Eine Grammatik ist vom *Typ 1* oder **kontextsensitiv**, falls für alle Regeln $w_1 \to w_2$ in P gilt: $|w_1| \leq |w_2|$.

Eine Typ 1–Grammatik ist vom *Typ 2* oder **kontextfrei**, falls für alle Regeln $w_1 \to w_2$ in P gilt, dass w_1 eine einzelne Variable ist, d.h. $w_1 \in V$.

Eine Typ 2–Grammatik ist vom *Typ 3* oder **regulär**, falls zusätzlich gilt: $w_2 \in \Sigma \cup \Sigma V$, d.h. die rechten Seiten von Regeln sind entweder einzelne Terminalzeichen oder ein Terminalzeichen gefolgt von einer Variablen.

Eine Sprache $L \subseteq \Sigma^*$ heißt vom Typ 0 (Typ 1, Typ 2, Typ 3), falls es eine Typ 0 (Typ 1, Typ 2, Typ 3)–Grammatik G gibt mit $L(G) = L$.


Menge aller Sprachen
Typ 0-Sprachen oder
rek. aufz. Sprachen
entscheidbare Sprachen
kontextsensitive
oder Typ 1-Sprachen
kontextfreie oder
Typ 2-Sprachen
reguläre oder
Typ 3-Sprachen

Das Wortproblem

Satz: Das *Wortproblem* für Typ 1-Sprachen (und damit auch für Typ 2, Typ 3-Sprachen) ist entscheidbar.

Genauer:

Es gibt einen Algorithmus, der bei Eingabe einer kontext-sensitiven Grammatik $G = (V, \Sigma, P, S)$ und eines Wortes $x \in \Sigma^*$ in endlicher Zeit entscheidet, ob $x \in L(G)$ oder $x \notin L(G)$.

INPUT $(G, x);\ \{\ |x| = n\ \}$

$T := \{S\};$

REPEAT

$T_1 := T;$

$T := Abl_n(T_1)$

UNTIL $(x \in T)$ OR $(T = T_1);$

IF $x \in T$

THEN WriteString('x liegt in L(G)')

ELSE WriteString('x liegt nicht in L(G)')

END

$$Abl_n(X) = X \cup \{w \in (V \cup \Sigma)^* \mid |w| \leq n \wedge w' \Rightarrow w \text{ für ein } w' \in X\}$$

Reguläre Sprachen

Def: Ein *(deterministischer) endlicher Automat* M wird spezifiziert durch ein 5-Tupel

$$M = (Z, \Sigma, \delta, z_0, E).$$

Z bezeichnet die (endliche) Menge der *Zustände*

Σ ist das *Eingabealphabet*, $Z \cap \Sigma = \emptyset$.

$z_0 \in Z$ ist der *Startzustand*

$E \subseteq Z$ ist die Menge der *Endzustände* und

$\delta : Z \times \Sigma \longrightarrow Z$ heißt die *Überführungsfunktion*.

Satz: Jede durch endliche Automaten erkennbare Sprache ist regulär (also Typ 3).

Nichtdeterministische Automaten

Def: Ein nichtdeterministischer, endlicher Automat (kurz: NFA) wird spezifiziert durch ein 5-Tupel

$$M = (Z, \Sigma, \delta, S, E)$$

- Z ist die endliche *Zustandsmenge*
- Σ ist das *Eingabealphabet*, $Z \cap \Sigma = \emptyset$
- δ ist die Überführungsfunktion von $Z \times \Sigma$ nach Z^*,
- $S \subseteq Z$ die Menge der *Startzustände*
- $E \subseteq Z$ die Menge der *Endzustände*

Die von einem NFA *akzeptierte Sprache* ist

$$T(M) = \{x \in \Sigma^* \mid \delta(S, x) \cap E \neq \emptyset\}$$

Satz: (Rabin, Scott) Jede von einem NFA akzeptierbare Sprache ist auch durch einen DFA akzeptierbar.

Satz: Für jede reguläre Grammatik G gibt es einen NFA M mit $L(G) = T(M)$.

Reguläre Ausdrücke

- $\emptyset$ ist ein regulärer Ausdruck.
- ε ist ein regulärer Ausdruck.
- Für jedes $a \in \Sigma$ ist a ist ein regulärer Ausdruck.
- Wenn α und β reguläre Ausdrücke sind, dann auch $\alpha\beta$, $(\alpha|\beta)$, sowie $(\alpha)^*$.

Einem regulären Ausdruck γ wird in eindeutiger Weise – induktiv über den Aufbau von γ – eine Sprache zugeordnet, die wir mit $L(\gamma)$ bezeichnen.

- Falls $\gamma = \emptyset$, so ist $L(\gamma) = \emptyset$.
- Falls $\gamma = \varepsilon$, so ist $L(\gamma) = \{\varepsilon\}$.
- Falls $\gamma = a$, so ist $L(\gamma) = \{a\}$.
- Falls $\gamma = \alpha\beta$, so ist $L(\gamma) = L(\alpha)L(\beta)$.
- Falls $\gamma = (\alpha|\beta)$, so ist $L(\gamma) = L(\alpha) \cup L(\beta)$.
- Falls $\gamma = (\alpha)^*$, so ist $L(\gamma) = L(\alpha)^*$.

Satz:(Kleene) Die Menge der durch reguläre Ausdrücke beschreibbaren Sprachen ist genau die Menge der regulären Sprachen.

$R_{i,j}^k = \{x \in \Sigma^* \mid$ die Eingabe x überführt den Automaten, gestartet im Zustand z_i in den Zustand z_j so, dass keiner der "Zwischenzustände" – außer z_i und z_j selbst – einen Index größer als k hat $\}$

Das Pumping Lemma

Satz: (Pumping Lemma, uvw-Theorem)

Sei L eine reguläre Sprache. Dann gibt es eine Zahl n, so dass sich alle Wörter $x \in L$ mit $|x| \geq n$ zerlegen lassen in $x = uvw$, so dass folgende Eigenschaften erfüllt sind:

1. $|v| \geq 1$,
2. $|uv| \leq n$,
3. für alle $i = 0, 1, 2, \ldots$ gilt: $uv^i w \in L$.

Äquivalenzrelationen und Minimalautomaten

Man kann jeder Sprache L eine Äquivalenzrelation R_L auf Σ^* zuordnen.

Def: Es gilt xR_Ly genau dann, wenn für alle Wörter $z \in \Sigma^*$ gilt:

$$xz \in L \Leftrightarrow yz \in L$$

Satz: (Myhill, Nerode) Eine Sprache L ist genau dann regulär, wenn der Index von R_L endlich ist.

Algorithmus Minimalautomat

Eingabe: ein DFA M (Zustände, die vom Startzustand aus nicht erreichbar sind, sind bereits entfernt).

Ausgabe: Minimalautomat für $T(M)$.

1. Stelle eine Tabelle aller Zustandspaare $\{z, z'\}$ mit $z \neq z'$ von M auf.
2. Markiere alle Paare $\{z, z'\}$ mit $z \in E$ und $z' \notin E$ (oder umgekehrt).
3. Für jedes noch unmarkierte Paar $\{z, z'\}$ und jedes $a \in \Sigma$ teste, ob $\{\delta(z, a), \delta(z', a)\}$ bereits markiert ist. Wenn ja: markiere $\{z, z'\}$.
4. Wiederhole den letzten Schritt, bis sich keine Änderung in der Tabelle mehr ergibt.
5. Alle jetzt noch unmarkierten Paare können jeweils zu einem Zustand verschmolzen werden.

Abschlußeigenschaften

Satz: Die regulären Sprachen sind abgeschlossen unter:

- Vereinigung
- Schnitt
- Komplement
- Produkt
- Stern

Entscheidbarkeit

Das *Wortproblem* (gegeben: x ; gefragt: liegt x in $L(G)$ bzw. $T(M)$)

Das *Leerheitsproblem* (gegeben: G (bzw. M); gefragt ob $L(G) = \emptyset$ (bzw. $T(M) = \emptyset$))

Das *Endlichkeitsproblem* (gegeben: G (bzw. M); gefragt: $|L(G)| < \infty$ (bzw. $|T(M)| < \infty$)),

Das *Schnittproblem* (gegeben G_1, G_2 (bzw. M_1, M_2); gefragt: $L(G_1) \cap L(G_2) = \emptyset$ (bzw. $T(M_1) \cap T(M_2) = \emptyset$)).

Das *Äquivalenzproblem* (gegeben: G_1, G_2 bzw. M_1, M_2; gefragt: $L(G_1) = L(G_2)$ bzw. $T(M_1) = T(M_2)$).

Kontextfreie Sprachen: Eliminierung der ε Regeln

1: Die Menge der Variablen V wird in V_1 und V_2 zerlegt so, dass für alle $A \in V_1$ (und nur für diese) gilt $A \Rightarrow^* \varepsilon$.

2: Als nächstes werden alle ε-Regeln aus P entfernt und für jede Regel der Form $B \to xAy$ mit $B \in V$, $A \in V_1$, $xy \in (V \cup \Sigma)^+$ wird eine weitere Regel der Form $B \to xy$ zu P zugefügt.

Kontextfreie Sprachen: Eliminierung der Regeln der Form $A \to B$

1: Falls es eine Menge von Variablen $B_1, \ldots, B_k$ gibt mit $B_1 \to B_2, \ldots$, $B_{k-1} \to B_k$ und $B_k \to B_1$, dann werden die Variablen $B_1, \ldots, B_k$ durch eine einzige Variable B ersetzt.

2: Die Variablen können so durchnumeriert werden, $V = \{A_1, A_2, \ldots, A_n\}$, dass aus $A_i \to A_j \ \in P$ folgt $i < j$. Wir gehen nun *von hinten nach vorne* vor und eliminieren für $k = n-1, \ldots, 1$ alle Regeln der Form $A_k \to A_{k'},\ k' > k$. Seien die Regeln mit $A_{k'}$ auf der linken Seite gegeben durch

$$A_{k'} \to x_1 | x_2 | \ldots | x_k.$$

Wir fügen dann die Regeln

$$A_k \to x_1 | x_2 | \ldots | x_k \text{ hinzu.}$$

Def: Eine kontextfreie Grammatik G mit $\varepsilon \notin L(G)$ heißt in *Chomsky Normalform*, falls alle Regeln eine der beiden Formen haben:

$$A \to BC$$

bzw.

$$A \to a$$

Hierbei stehen A, B, C für Variablen und a für ein Terminalsymbol.

Satz: Zu jeder kontextfreien Grammatik G mit $\varepsilon \notin L(G)$ gibt es eine Chomsky Normalform Grammatik G' mit $L(G) = L(G')$.

Def: Eine kontextfreie Grammatik G mit $\varepsilon \notin L(G)$ heißt in *Greibach Normalform*, falls alle Regeln die Form haben:

$$A \to aB_1B_2 \dots B_k \quad (k \geq 0)$$

Hierbei stehen $A, B_1, \dots, B_k$ für Variablen und a für ein Terminalsymbol.

Satz: Zu jeder kontextfreien Grammatik G mit $\varepsilon \notin L(G)$ gibt es eine Greibach Normalform Grammatik G' mit $L(G) = L(G')$.

Satz: (Pumping Lemma, $uvwxy$-Theorem) Sei L eine kontextfreie Sprache. Dann gibt es eine Zahl $n \in N$, so dass sich alle Wörter $z \in L$ mit $|z| \geq n$ zerlegen lassen in $z = uvwxy$ mit folgenden Eigenschaften:

1. $|vx| \geq 1$
2. $|vwx| \leq n$
3. für alle $i \geq 0$ gilt: $uv^iwx^iy \in L$

Satz: Die kontextfreien Sprachen sind abgeschlossen unter:

- Vereinigung
- Produkt
- Stern

Die kontextfreien Sprachen sind nicht abgeschlossen unter:

- Schnitt
- Komplement

CYK-Algorithmus

Eingabe: $x = a_1 a_2 \ldots a_n$

FOR $i := 1$ TO n DO $(*$ Fall $j = 1 *)$

  $T[i,1] := \{A \in V \mid A \to a_i \; \in P\}$

END;

FOR $j := 2$ TO n DO $(*$ Fall $j > 1 *)$

  FOR $i := 1$ TO $n + 1 - j$ DO

    $T[i,j] := \emptyset$;

    FOR $k := 1$ TO $j - 1$ DO

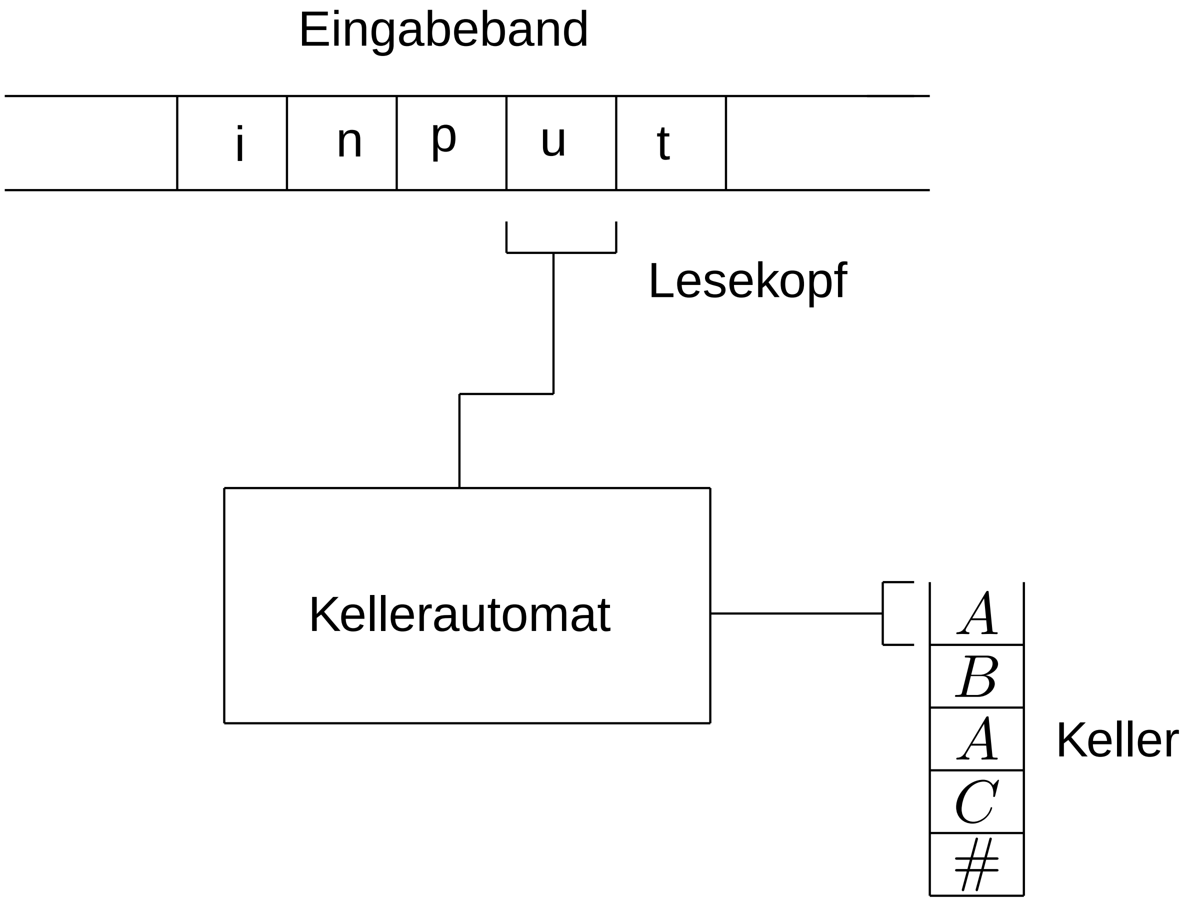
      $T[i,j] := T[i,j] \cup \{A \mid A \to BC \; \in P \wedge$
        $B \in T[i,k] \; \wedge \; C \in T[i+k, j-k]\}$

    END;

  END;

END

Kellerautomaten



Def: Ein *(nichtdeterministischer) Kellerautomat* (PDA) wird angegeben durch ein 6-Tupel

$$M = (Z, \Sigma, \Gamma, \delta, z_0, \#)$$

Hierbei sind:

- Z die endliche Menge der *Zustände*
- Σ das *Eingabealphabet*
- Γ das *Kelleralphabet*
- $\delta : Z \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \longrightarrow \mathcal{P}_e(Z \times \Gamma^*)$ die *Überführungsfunktion*

  (Hierbei bedeutet $\mathcal{P}_e$ die Menge aller *endlichen* Teilmengen)
- $z_0 \in Z$ der *Startzustand*
- $\# \in \Gamma$ das *unterste Kellerzeichen*

Def: Eine *Konfiguration* k eines Kellerautomaten ist gegeben durch ein Tripel $k \in Z \times \Sigma^* \times \Gamma^*$.

Wir definieren die Relation $\vdash$

$$(z, a_1 \dots a_n, A_1 \dots A_m) \ \vdash \begin{cases} (z', a_2 \dots a_n, B_1 \dots B_k A_2 \dots A_m), \\ \qquad \text{falls } \ \delta(z, a_1, A_1) \ni (z', B_1 \dots B_k) \\ (z', a_1 a_2 \dots a_n, B_1 \dots B_k A_2 \dots A_m) \\ \qquad \text{falls } \ \delta(z, \varepsilon, A_1) \ni (z', B_1 \dots B_k) \end{cases}$$

$$N(M) = \{x \in \Sigma^* \mid (z_0, x, \#) \ \vdash^* \ (z, \varepsilon, \varepsilon) \text{ für ein } z \in Z\}$$

Satz: Eine Sprache L ist kontextfrei genau dann, wenn L von einem nichtdeterministischen Kellerautomaten erkannt wird.

Deterministisch kontextfreie Sprachen

Def: Ein Kellerautomat M heißt *deterministisch*, falls für alle $z \in Z,\ a \in \Sigma$ und $A \in \Gamma$ gilt:

$$|\delta(z,a,A)| + |\delta(z,\varepsilon,A)| \leq 1$$

Es kommt hinzu, daß deterministisch kontextfreie Kellerautomaten *per Endzustand* akzeptieren und nicht *per leerem Keller*.

Eine Sprache heißt *deterministisch kontextfrei*, falls sie von einem deterministischen Kellerautomaten erkannt wird.

Abschlußeigenschaften

Die deterministisch kontextfreien Sprachen sind unter Komplementbildung abgeschlossen.

Die deterministisch kontextfreien Sprachen sind *nicht* unter Schnitt und Vereinigung abgeschlossen.

Der Schnitt einer (deterministisch) kontextfreien Sprache mit einer regulären Sprache ist wieder (deterministisch) kontextfrei.