Vorlesung Informatik III – Theoretische Informatik

Formale Sprachen, Berechenbarkeit & Komplexitätstheorie

Prof. Dr. Peter Thiemann

Basierend auf einem Mitschrieb im WS 2015/16 von Ralph Lesch†

WS 2016/17

Zuletzt aktualisiert: 2017-08-02

†alph.lesch@neptun.uni-freiburg.de

Inhaltsverzeichnis

1 Vorspann: Sprachen

Vorlesung: 19.10.16

Def. 1.1: Ein *Alphabet* Σ ist eine endliche Menge von *Zeichen*. ⊕

Zeichen sind hier beliebige abstrakte Symbole. Im Folgenden steht Σ immer für ein Alphabet.

Bsp.:

- $\Sigma = \{a, \ldots, z\}$
- $\Sigma = \{0, 1\}$

Def. 1.2: Die Menge Σ^* der *Worte* über Σ ist induktiv definiert durch:

1. $\varepsilon \in \Sigma^*$
2. Wenn $a \in \Sigma$ und $w \in \Sigma^*$ dann auch $aw \in \Sigma^*$

Die *Länge* eines Wortes, $|\cdot| : \Sigma^* \to \mathbb{N}$, ist induktiv definiert durch

1. $|\varepsilon| = 0$
2. $|aw| = 1 + |w|$

⊕

Ein Wort ist also immer eine endliche Folge von Zeichen.

Bsp.:

- `rambo` (Länge 5)
- `pizza`, `zipza` (ungleich)
- ε (Länge 0)
- `rambopizza`

Wörter lassen sich „verketten"/„hintereinanderreihen". Die entsprechende Operation heißt *Konkatenation*, geschrieben „$\cdot$" (wie Multiplikation).

Bsp.:

- `rambo` $\cdot$ `pizza` = `rambopizza`
- `rambo` $\cdot \varepsilon =$ `rambo` $= \varepsilon \cdot$ `rambo`

Def. 1.3 (Konkatenation von Wörtern)**:** Die *Konkatenation*, $\cdot : \Sigma^* \times \Sigma^* \to \Sigma^*$, ist induktiv definiert durch:

1. $\varepsilon \cdot v = v$
2. $(aw) \cdot v = a(w \cdot v)$

$\oplus$

Eigenschaften von „$\cdot$“:

- Assoziativität
- ε ist neutrales Element
- *nicht* kommutativ

Lemma 1.1: Für alle $w \in \Sigma^*$ gilt $w \cdot \varepsilon = w$.

Beweis: Per struktureller Induktion über das Wort w:

IA $w = \varepsilon$.
Es folgt aus Fall 1 der Definition von „$\cdot$“: $\varepsilon \cdot \varepsilon = \varepsilon$.

IV $w' \cdot \varepsilon = w'$.

IS $w = aw'$.
Mit Fall 2 von „$\cdot$“ folgt $aw' \cdot \varepsilon = a(w' \cdot \varepsilon) \overset{\mathsf{IV}}{=} aw'$

$\square$

Beweis: Alternativ, per Induktion über $n = |w|$:

IA $n = 0$.

Es folgt aus der Definition von „$|\cdot|$“ (und dem arithmetischen Fakt, dass $1+x \neq 0$), dass $w = \varepsilon$.

Es folgt aus Fall 1 der Definition von „$\cdot$“, dass $\varepsilon \cdot \varepsilon = \varepsilon$.

IV Für alle w' mit $|w'| = n'$ gilt $w' \cdot \varepsilon = w'$.

IS $n = n' + 1$.

Es folgt aus der Definition von „$|\cdot|$“ (und dem arithmetischen Fakt, dass $1+x \neq 0$), dass $w = aw'$.

Somit gilt $|aw'| = 1 + |w'| = 1 + n'$ und daher auch $|w'| = n'$.

Mit Fall 2 von „$\cdot$“ folgt $aw' \cdot \varepsilon = a(w' \cdot \varepsilon) \overset{\mathsf{IV}}{=} aw'$

□

Lemma 1.2: Für $v, w \in \Sigma^*$ gilt $|v \cdot w| = |v| + |w|$.

Beweis: Per Induktion über v.

IA $v = \varepsilon$.

Nach Def. von „$\cdot$" gilt $|\varepsilon \cdot w| = |w|$.

Mit Def. von „$|\cdot|$", Fall 1 folgt $|w| = 0 + |w| = |\varepsilon| + |w|$

IV $|v' \cdot w| = |v'| + |w|$.

IS $v = av'$.

Mit Def. von „$|\cdot|$" und „$\cdot$" folgt:

$$|av' \cdot w| = |a(v' \cdot w)| = 1 + |v' \cdot w| \overset{\text{IV}}{=} 1 + |v'| + |w| = (1 + |v'|) + |w| = |v| + |w|$$

□

Der Konkatenationsoperator „$\cdot$" wird oft weggelassen (ähnlich wie der Mulitplikationsoperator in der Arithmetik). Ebenso können durch die Assoziativität Klammern weggelassen werden:

$$w_1 w_2 w_3 \quad \text{heißt also} \quad w_1 \cdot w_2 \cdot w_3 \quad = (w_1 \cdot w_2) \cdot w_3 = w_1 \cdot (w_2 \cdot w_3)$$

Wörter lassen sich außerdem *potenzieren*:

Def. 1.4: Die *Potenzierung* von Worten, $\cdot^{\cdot} : \Sigma^* \times \mathbb{N} \to \Sigma^*$, ist induktiv definiert durch

1. $w^0 = \varepsilon$
2. $w^{n+1} = w \cdot w^n$

⊕

Bsp.:

- $\texttt{rambo}^1 = \texttt{rambo}$
- $\texttt{rambo}^0 = \varepsilon$
- $\texttt{rambo}^3 = \texttt{ramboramborambo}$

Def. 1.5: Eine *Sprache* über Σ ist eine Menge $L \subseteq \Sigma^*$. ⊕

Bsp.:

- $\{\texttt{banane}, \texttt{aprikose}, \texttt{orange}, \dots\}$
- $\{\texttt{rot}, \texttt{gelb}, \texttt{grün}\}$
- $\{\texttt{rambo}, \texttt{pizza}, \varepsilon, \texttt{blümchen}\}$
- $\{\}$ (die „leere Sprache")
- $\{\varepsilon\}$
- Σ^*

Sämtliche Mengenoperationen sind auch Sprachoperationen, insbesondere Schnitt ($L_1 \cap L_2$), Vereinigung ($L_1 \cup L_2$), Differenz ($L_1 \setminus L_2$) und Komplement ($L_1^{-1} = \Sigma^* \setminus L_1$).

Weitere Operationen auf Sprachen sind Konkatenation und Potenzierung, sowie der *Kleene Abschluss*.

Def. 1.6 (Konkatenation und Potenzierung von Sprachen)**:** Sei $U, V \subseteq \Sigma^*$ dann ist die *Konkatenation* U und V definiert durch

$$U \cdot V = \{uv \mid u \in U, v \in V\}$$

und die *Potenzierung* von U induktiv definiert durch

1. $U^0 = \{\varepsilon\}$
2. $U^{n+1} = U \cdot U^n$

$\oplus$

Bsp.:

- $\{\texttt{rambo}, \texttt{pizza}\} \cdot \{\texttt{rot}, \texttt{gelb}\} = \{\texttt{ramborot}, \texttt{pizzarot}, \texttt{rambogelb}, \texttt{pizzagelb}\}$
- $\{\texttt{rambo}, \texttt{pizza}\} \cdot \{\varepsilon, \texttt{gelb}\} = \{\texttt{rambo}, \texttt{pizza}, \texttt{rambogelb}, \texttt{pizzagelb}\}$
- $\{\texttt{rambo}, \varepsilon\}^3 = \{\varepsilon, \texttt{rambo}, \texttt{ramborambo}, \texttt{ramboramborambo}\}$

Wie bei der Konkatenation von Wörtern lässt man den Konkatenationsoperator oft weg.

Def. 1.7 (Kleene-Abschluss, Kleene-Stern)**:** Sei $U \subseteq \Sigma^*$. Der *Kleene-Abschluss* ist induktiv definiert als

1. $U^* = \bigcup_{n \in \mathbb{N}} U^n \quad [\ni \varepsilon]$
2. $U^+ = \bigcup_{n \geq 1} U^n$

$\oplus$

2 Reguläre Sprachen und endliche Automaten

Vorlesung: 21.10.16

Endliche Automaten sind ein einfaches, formales Maschinenmodell. Ein endlicher Automat A berechnet, für eine bestimmte Sprache $L(A)$, ob ein gegebenes Wort w in ihr enthalten ist (Wortproblem, $w \in L(A)$?). Die Berechnungen, die sich mit endliche Automaten ausdrücken lassen sind stark beschränkt, allerdings erlaubt diese Einschränkung *die Entscheidung* von Fragen wie dem Wortploblem oder dem Leerheitsproblem ($L(A) \neq \varnothing$). D.h. für jede dieser Fragen existiert ein Algorithmus.

2.1 Endliche Automaten

Wir beschreiben zunächst die Bestandteile eines endlichen Automaten:

Endliches Band (read-only, jede Zelle enthält ein $a_i \in \Sigma$, der Inhalt des Bandes ist das *Eingabewort*, bzw. die *Eingabe*)

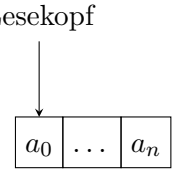


Abb. 1: Endliches Band

Lesekopf

- Der *Lesekopf* zeigt auf ein Feld des Bandes, oder hinter das letzte Feld.
- Er bewegt sich feldweise nach rechts; andere Bewegungen (vor- bzw. zurückspulen) sind nicht möglich.
- Wenn er hinter das letzte Zeichen zeigt, *stoppt* der Automat. Er muss sich nun „entscheiden" ob er das Wort *akzeptiert* oder nicht.

Zustände q aus *endlicher* Zustandsmenge Q.

Startzustand $q_0 \in Q$.

Akzeptierende Zustände $F \subseteq Q$

Transitionsfunktion Im Zustand q beim Lesen von a gehe nach Zustand $\delta(q) = q'$.

Der endliche Automat akzeptiert eine Eingabe, falls er in einem akzeptierenden Zustand stoppt.

Bsp. 2.1: Aufgabe:
„Erkenne alle Stapel von Maccarons in denen höchstens ein grüner Maccaron vorkommt."


†

Ein passendes Alphabet wäre $\Sigma = \{\texttt{grün}, \texttt{nicht-grün}\}$. Wir definieren die folgenden Zustände. (die Metapher hier ist: „wenn ich mehr als einen grünen Maccaron esse wird mir übel, und das wäre nicht akzeptabel")

Zustand	Bedeutung
q_0	„alles gut"
q_1	„mir wird schon flau"
q_2	„mir ist übel"

Der Startzustand ist q_0. Akzeptierende Zustände sind q_0 und q_1. Die Transistionsfunktion δ ist

	`grün`	`nicht-grün`	
q_0	q_1	q_0	wechsle nach q_1 falls `grün`, ansonsten verweile
q_1	q_2	q_1	wechsle nach q_2 falls `grün`, ansonsten verweile
q_2	q_2	q_2	verweile, da es nichts mehr zu retten gibt

Def. 2.1 (DEA)**:** Ein *deterministischer endlicher Automat (DEA)*, (DFA $\hat{=}$ deterministic finite automaton) ist ein 5-Tupel

$$M = (Q, \Sigma, \delta, q_0, F)$$

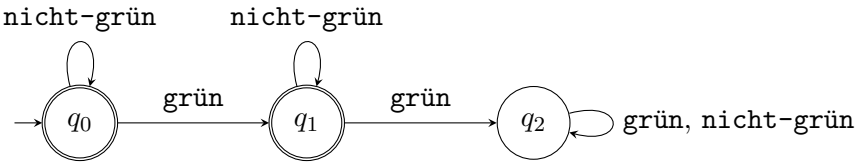
- Q *endliche* Zustandsmenge
- Σ *endl.* Alphabet
- $\delta : Q \times \Sigma \to Q$ Transitionsfunktion
- $q_0 \in Q$ Startzustand

† Von links nach rechts:
By Mariajudit - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=48726001
By Michelle Naherny - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=44361114
By Keven Law - originally posted to Flickr as What's your Colour???, CC BY-SA 2.0, https://commons.wikimedia.org/w/index.php?curid=6851868

- $F \subseteq Q$ akzeptierende Zustande

$\oplus$

DEAs lassen sich auch graphisch darstellen. Dabei gibt man für den Automaten einen gerichteten Graphen an. Die Knoten des Graphen sind die Zustände und mit Zeichen gelabelte Kanten zeigen welchen Zustandsübergang die Transitionsfunktion für das nächste Zeichen erlaubt. Der Startzustand ist mit einem ungelabelten Pfeil markiert und finale Zustände sind doppelt eingekreist. Hier ist die graphische Darstellung von $A_{\texttt{Maccaron}}$ aus Beispiel 2.1



DEAs charakterisieren die Sprachen durch die Menge an Wörtern, die sie akzeptieren.

Bsp. 2.2: Sei $M = (Q, \Sigma, \delta, q_0, F)$ ein DEA.

- Wenn $F = Q$, dann ist $L(M) = \Sigma^*$.
- Wenn $F = \emptyset$, dann ist $L(M) = \emptyset$.

Def. 2.2: Die Erweiterung von $\delta : Q \times \Sigma \to Q$ auf Worte $\hat{\delta} : Q \times \Sigma^* \to Q$ ist induktiv definiert durch

1. $\hat{\delta}(q, \varepsilon) = q$ (Wortende erreicht)
2. $\hat{\delta}(q, aw) = \hat{\delta}(\delta(q, a), w)$ (Rest im Folgezustand verarbeiten)

$\oplus$

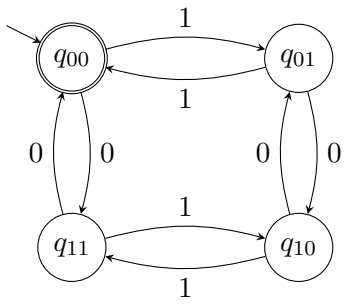
Def. 2.3: Sei $M = (Q, \Sigma, \delta, q_0, F)$.
Die *von M erkannte Sprache* ist

$$L(M) = \{w \in \Sigma^* \mid \hat{\delta}(q_0, w) \in F\}$$

Eine durch einen DEA erkannte Sprache heißt *regulär*. $\oplus$

Es folgen zwei Beispiele für reguläre Sprachen:

Bsp. 2.3: $L = \{w \in \{0, 1\}^* \mid w$ enthält gerade Anzahl von 0 und gerade Anzahl von $1\}$



Bsp. 2.4: Sei $A \geq 0$ nat. Zahl, $\Sigma = \{0, 1, \ldots, A\}$

$$L = \{a_1 \ldots a_n \mid \exists J \subseteq \{1, \ldots, n\}, \sum_{i \in J} a_i = A\} \subseteq \Sigma^*$$

D.h. gegeben eine Liste von Zahlen $\in \Sigma$. Akzeptiere diejenigen Listen, für die eine Teilliste existiert, deren Summe genau A ist.

$$\begin{aligned} Q &= \mathcal{P}\{0, 1, \ldots, A\} \\ \delta(q, a) &= q \cup \{x \in \{0, \ldots, A\} \mid x - a \in q\} \\ q_0 &= \{0\} \\ F &= \{q \in Q \mid A \in q\} \end{aligned}$$

$q \in Q$ bezeichnet die Menge an möglichen Summen $\leq A$, die mit den bisher gelesenen Zeichen gebildet werden kann. Die Transitionsfunktion δ fügt die Summen zum aktuellen Zustand hinzu, die sich durch addieren der aktuell gelesenen Ziffer zu den alten Möglichkeiten ergeben.

Bsp. 2.5: Beispiel für eine nicht-reguläre Sprache.

$$L = \{0^n 1^n \mid n \in \mathbb{N}\}$$

erkennbar durch TM die immer anhält, *aber nicht* von einem DEA [*nicht* regulär] akzeptiert werden kann.

Beweis: Angenommen $L = L(M)$ für DEA $M = (Q, \Sigma, q_0, \delta, F)$

Beobachtung: $\exists m \neq n$, so dass $\hat{\delta}(q_0, 0^m) = \hat{\delta}(q_0, 0^n) = q'$ weil Q endlich.

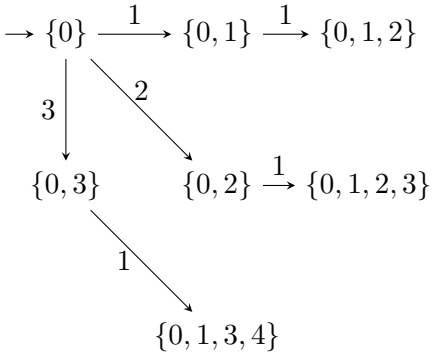
- Falls nun $\hat{\delta}(q', 1^m) \in F$, dann ist auch $\hat{\delta}(q_0, 0^n 1^m) \in F$ und somit $0^n 1^m \in L(M)$ mit $n \neq m$ ↯
- Falls $\hat{\delta}(q', 1^m) \notin F$, dann gilt auch $\hat{\delta}(q_0, 0^m 1^m) \notin F$ und somit $0^m 1^m \notin L$ ↯

Also kann M nicht existieren! □

2.2 Minimierung endlicher Automaten

Vorlesung: 26.10.16

Betrachte den Automaten aus Beispiel 2.4. Sei $A = 4$, $\Sigma = \{0, 1, 2, 3, 4\}$ mit Zustandsmenge $Q = \mathcal{P}(\Sigma)$. D.h. unter anderem: $\{0, 1, 3\} \in Q$. Hier ist ein Ausschnitt aus dem Zustandsdiagramm:



Es ist zu bemerken, dass manche Zustände von Q nie erreicht werden können, z.B. $\emptyset$. Sei für die folgenden Überlegungen $M = (Q, \Sigma, \delta, q_0, F)$ ein DEA.

Def. 2.4: Ein Zustand $q \in Q$ ist *erreichbar*, falls ein $w \in \Sigma^*$ existiert, so dass $\hat{\delta}(q_0, w) = q$. M heißt *reduziert*, falls alle Zustände erreichbar sind. ⊕

Satz 2.1: Die Menge der erreichbaren Zustände kann in $O(|Q| * |\Sigma|)$ berechnet werden.

Beweis:

- Fasse A als Graphen auf.
- Wende Tiefensuche an, markiere dabei alle besuchten Zustände.
- Die markierten Zustände bilden die Menge der erreichbaren Zustände.

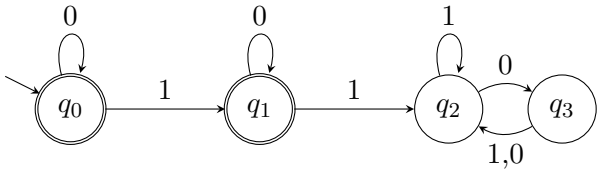
□

. . . (TODO: hier fehlt noch etwas)

Vorlesung: 28.10.16

Beobachtung: Auch ein Automat mit lauter erreichbaren Zuständen muss nicht minimal sein.

Bsp. 2.6:



Dieser Automat erkennt die gleiche Sprache wie in (2.3) („höchstens eine 1"), hat nur erreichbare Zustände, aber mehr Zustände als in (2.3).

Beobachtung: Aber q_2 und q_3 verhalten sich gleich in dem Sinn, dass

$$\forall w : \hat{\delta}(q_2, w) \notin F \text{ und } \hat{\delta}(q_3, w) \notin F$$

Def. 2.5: Zwei Zustände $q, p \in Q$ eines DFA sind *äquivalent*, geschrieben $p \equiv q$, falls

$$\forall w \in \Sigma^*, \hat{\delta}(p, w) \in F \text{ gdw } \hat{\delta}(q, w) \in F$$

⊕

Lemma 2.2: Die Relation „≡" ist eine *Äquivalenzrelation*, das heißt, „≡" ist reflexiv, transitiv und symmetrisch.

Beweis: ≡ ist offensichtlich reflexiv.

Die Symmetrie und Transitivität von ≡ folgt aus der Transitivität und Symmetrie der logischen Interpretation von „genau dann wenn" (gdw). □

Also sind q_2, q_3 aus Bsp. 2.6 äquivalent.

Erinnerung: Hauptlemma über Äquivalenzrelationen

$$[q] = \{p \in Q \mid p \equiv q\} \qquad [q] \text{ ist Äquivalenzklasse von } q$$

Äquivalenzklassen sind paarweise disjunkt:

Für alle $p, q \in Q$ gilt entweder $[p] = [q]$ oder $[p] \cap [q] = \emptyset$ (folgt aus Transitivität).

D.h. Q wird in disjunkte Äquivalenzklassen aufgeteilt. Anzahl der Äquivalenzklassen ist der *Index*. ⊕

Entwurf für Vorlesung: 28.10.16

(2.3):

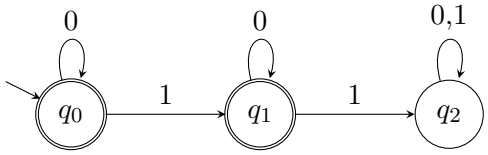


Abb. 2: Automat zu (2.3)

Allgemein gilt für alle $p, q \in Q$:

$$p \equiv q \Rightarrow \forall a \in \Sigma : \ \delta(p, a) \equiv \delta(q, a) \qquad (1)$$

Denn

$$\begin{aligned} p \equiv q &\Leftrightarrow \forall w \in \Sigma^* : \hat{\delta}(p,w) \in F \Leftrightarrow \hat{\delta}(q,w) \in F \\ &\Leftrightarrow (p \in F \Leftrightarrow q \in F) \wedge \forall a \in \Sigma : \forall w \in \Sigma^* : \hat{\delta}(p,aw) \in F \Leftrightarrow \hat{\delta}(q,aw) \in F \\ &\Rightarrow \forall a \in \Sigma : \forall w \in \Sigma^* : \hat{\delta}(\delta(p,a),w) \in F \Leftrightarrow \hat{\delta}(\delta(q,a),w) \in F \\ &\Leftrightarrow \forall a \in \Sigma : \delta(p,a) \equiv \delta(q,a) \end{aligned}$$

Also können wir äquivalente Zustände zusammenfassen und Transitionen verschmelzen, wie in der folgenden Definition formalisiert.

Def. 2.6: Der Äquivalenzklassenautomat $M' = (Q', \Sigma, \delta', q_0', F')$ zu M ist bestimmt durch:

$$\begin{aligned} Q' &= \{[q] \mid q \in Q\} & \delta'([q],a) &= [\delta(q,a)] \\ q_0' &= [q_0] & F' &= \{[q] \mid q \in F\} \end{aligned}$$

⊕

Dabei ist $[q] = \{p \in Q \mid p \equiv q\}$

Satz 2.3: Der Äquivalenzklassenautomat ist wohldefiniert und $L(M) = L(M')$.

Beweis:

1. Wohldefiniert: zu zeigen $\delta'([q],a) = [\delta(q,a)]$ ist nicht abhängig von der Wahl des Repräsentanten $q \in [q]$. Das folgt direkt aus (1) gezeigt.
2. $L(M) = L(M')$ zeige für alle $w \in \Sigma^*$ und alle q: $\hat{\delta}(q,w) \in F \Leftrightarrow \hat{\delta}'([q],w) \in F'$
   Induktion über w:
   I.A. $w = \varepsilon$: $\hat{\delta}(q,\varepsilon) = q \in F \Leftrightarrow \hat{\delta}'([q],\varepsilon) = [q] \in F'$ nach Definition.
   I.V.: $\forall w' \in \Sigma$, $\forall q \in Q$, $\hat{\delta}(q,w') \in F \Leftrightarrow \hat{\delta}'([q],w') \in F'$
   I.S.:

$$\begin{aligned} \hat{\delta}(q,aw') \in F &\Longleftrightarrow \hat{\delta}(\delta(q,a),w') \in F \\ &\overset{I.V.}{\Longleftrightarrow} \hat{\delta}'([\delta(q,a)],w') \in F' \\ &\Longleftrightarrow \hat{\delta}'(\delta'([q],a),w') \in F' \\ &\Longleftrightarrow \hat{\delta}'([q],aw') \in F' \end{aligned}$$

   Also für $q = q_0$: $\forall w \in \Sigma^*$, $w \in L(M) \Longleftrightarrow \hat{\delta}(q_0,w) \in F \Leftrightarrow \hat{\delta}'([q_0],w) \in F' \Longleftrightarrow w \in L(M')$

□

Bem: Die Konstruktion von M' kann in $O(|Q||\Sigma| \log |Q|)$ passieren.

Warum ist nun der Äquivalenzklassenautomat minimal?
$\rightarrow$ Satz von Myhill-Nerode

Def. 2.7: Eine Äquivalenzrelation $R \subseteq \Sigma^* \times \Sigma^*$ heißt rechtsinvariant, falls

$$(u, v) \in R \Rightarrow \forall w \in \Sigma^*, (u \cdot w, v \cdot w) \in R$$

$\oplus$

Bsp. 2.7: Für einen DEA M definiere

$$R_M = \{(u, v) \mid \hat{\delta}(q_0, u) = \hat{\delta}(q_0, v)\}$$

- ist Äquivalenzrelation
- ist rechtsinvariant
- Anzahl der Äquivalenzklassen(Index von R_M)
  = Anzahl der „nützlichen“ Zustände, die von q_0 erreichbar sind.

Bsp. 2.8: Für eine Sprache $L \subseteq \Sigma^*$ definiere die Nerode Relation

$$R_L = \{(u, v) \mid \forall w \in \Sigma^* : uw \in L \Leftrightarrow vw \in L\}$$

- ist Äquivalenzrel.
- ist rechtsinvariant.

  Beweis: Sei $(u, v) \in R_L$. Zeige $\forall w \in \Sigma^*$, dass $(uw, vw) \in R_L$:
  I.A. Für $w = \varepsilon$ ist $(u\varepsilon, v\varepsilon) = (u, v) \in R_L$.
  I.S. Sei Aussage gezeigt für alle Wörter w' mit $|w'| \leq k$.
  I.V. Betrachte nun $w = w'a$. Dann ist auch $(uw', vw') \in R_L$. Damit gilt:

$$\begin{aligned}(uw', vw') \in R_L &\Leftrightarrow \forall z \in \Sigma^*, \quad uw'z \in L \Leftrightarrow vw'z \in L \\ &\Rightarrow \forall a \in \Sigma, z' \in \Sigma^* : uw'az' \in L \Leftrightarrow vw'az' \in L \\ &\Leftrightarrow \ (uw'a, vw'a) \in R_L\end{aligned}$$

$\square$

Beispiel: Drei Sprachen L_a, L_b und L_c mit unterschiedlichem Index (Anzahl an Äquivalenzklassen) bzgl. der Nerode Relation:

$$\left.\begin{array}{ll} L_a = \{\varepsilon\} & [\varepsilon] \equiv [\varepsilon] \\ w, v \in \Sigma^*,\ w, v \neq \varepsilon & [w] = [v] \end{array}\right\} \text{Index} = 2$$
$$L_b = \varnothing, L_c = \Sigma^* \qquad \text{Index} = 1$$

Die Menge der Äquivalenzklassen der Nerode Relation R_{L_a} auf L_a ist

$$L_a / R_{L_a} = \{[\varepsilon], [w]\}$$

Satz 2.4 (Nerode)**:** Die folgende Aussagen sind äquivalent:

1. $L \subseteq \Sigma^*$ wird von DEA akzeptiert.
2. L ist Vereinigung von Äquivalenzklassen einer rechtsinvarianten Äquivalenzrelation mit *endlichem* Index.
3. Die Nerode Relation R_L hat *endlichen* Index

Vorlesung: 2.11.16

Beweis: Wir beweisen die gegenseitige Äquivalenz in drei Schritten:

$$(1) \Rightarrow (2) \quad (2) \Rightarrow (3) \quad (3) \Rightarrow (1)$$

(1) ⇒ (2) : Sei M ein DEA mit

$$L(M) = \{w \mid \hat{\delta}(q_0, w) \in F\} = \bigcup_{q \in F} \{w \mid \hat{\delta}(q_0, w) = q\}$$

Nun sind $\{w \mid \hat{\delta}(q_0, w) = q\} = [q]_M$ genau die Äquivalenzklassen von R_M aus Bsp 2.7, einer rechtsinvarianten Äquivalenzrelation. Der Index ist die Anzahl der erreichbaren Zustände.

Also: $\text{Index}(R_M) \leq |Q| < \infty$.

(2) ⇒ (3) Sei R rechtsinvariante Äquivalenzrelation mit endlichem Index, so dass $L = \bigcup R$-Äquivalenzklassen

Es genügt zu zeigen, dass wenn $(u, v) \in R$ auch $(u, v) \in R_L$, d.h. dass $R \subseteq R_L$.[†]

[†] Zur Erklärung: Falls $R \subseteq R_L$, dann $\text{Index}(R) \geq \text{Index}(R_L)$. Intuitiv: Je mehr Elemente eine Äquivalenzrelation R enthält, desto mehr Elemente sind bzgl. dieser Relation äquivalent, d.h. desto weniger unterschiedliche Klassen gibt es.

Betrachte also ein beliebiges $(u, v) \in R$.

$$
\begin{aligned}
(u,v) \in R &\to u \in L \Leftrightarrow v \in L, \text{ da L Vereinigung von Äquivalenzklassen ist} \\
&\to uw \in L \Leftrightarrow vw \in L \ \forall w \in \Sigma^*, \text{ da } R \text{ rechtsinvariant} \\
&\to (u,v) \in R_N, \text{ nach Definition} \\
&\to R \subseteq R_N \\
&\to \#\text{Klassen}(R_L) \leq \#\text{Klassen}(R) < \infty
\end{aligned}
$$

(3) ⇒ (1) Gegeben R_L, konstruiere $\mathcal{A}' = (Q', \Sigma, \delta', q_0', F')$

- $Q' = \{[w]_{R_L} \mid w \in \Sigma^*\}$ endlich, weil $index(R_L)$ endlich
- $\delta'([w], a) = [wa]$ wohldefiniert, da R_L rechtsinvariant
- $q_0' = [\varepsilon]$
- $F' = \{[w] \mid w \in L\}$

Zeige $L(\mathcal{A}') = L$, d.h.

1. $\forall w \in \Sigma^* : w \in L(\mathcal{A}')$ gdw $\hat{\delta}([\varepsilon], w) \in F'$
2. $w \in L$ gdw $[w] \in F'$

Um zu zeigen, dass 1 gdw 2 genügt zu zeigen, dass $\hat{\delta}([\varepsilon], w) = [w]$. Dafür müssen wir wie folgt verallgemeinern um eine funktionierende Induktionsvoraussetzung zu erhalten.

$$\forall w \in \Sigma^* : \forall v \in \Sigma^* : \hat{\delta}'([v], w) = [v \cdot w]$$

Induktion über w:

IA $(w = \varepsilon)$: $\hat{\delta}'([v], \varepsilon) = [v] = [v \cdot \varepsilon]$

IV $\forall v \in \Sigma^* : \hat{\delta}'([v], w') = [v \cdot w]$

IS $w = aw'$:

$$
\begin{aligned}
\hat{\delta}'([v], aw') &= \hat{\delta}'(\delta'([v], a), w') \\
&= \hat{\delta}'([v \cdot a], w') \\
&\overset{\text{I.V.}}{=} [va \cdot w'] \\
&= [v \cdot \underbrace{aw'}_{=w}]
\end{aligned}
$$

Das gewünschte Ergebnis $\hat{\delta}([\varepsilon], w) = [w]$ ergibt sich für $v = \varepsilon$. □

Korollar 2.5: Der im Beweisschritt (3) ⇒ (1) konstruierte Automat $\mathcal{A}'$ ist minimaler Automat für L. ⊕

BEWEIS: L regulär. Sei $\mathcal{A}$ *beliebiger* DEA mit $L = L(\mathcal{A})$

$\mathcal{A}$ induziert $R_{\mathcal{A}}$ mit $|Q| \geq \text{index}(R_{\mathcal{A}})$ (da 1⇔2)

In $2 \Rightarrow 3$: $R_{\mathcal{A}} \subseteq R_L$, $\text{index}(R_{\mathcal{A}}) \geq \text{index}(R_L)$

In $3 \Rightarrow 1$ A' mit $|Q'| = \text{index}(R_L) \leq \text{index}(R_{\mathcal{A}}) \leq |Q|$

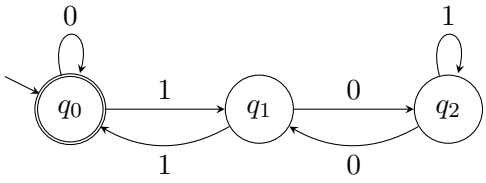
Minimalität von A' folgt aus freier Wahl von $\mathcal{A}$: Angenommen $\exists\, \mathcal{A}$ mit $L = L(\mathcal{A})$ und $|\mathcal{A}| < |\mathcal{A}'|$. Nach Folgerung gilt aber $|\mathcal{A}'| \leq |\mathcal{A}|$; ein Widerspruch. □

2.3 Pumping Lemma (PL) für reguläre Sprachen

Vorlesung: 26.10.16 (Eingeschoben)

Suche: Notwendiges Kriterium für Regularität

Bsp.: $L = \{w \in \{0,1\}^* \mid \text{bin}(w) \equiv_3 0\}$ ist regulär, dabei ist „bin“ die Dekodierung von einem Bitstring in eine natürliche Zahl.



- Es gilt offensichtlich, dass $11 \in L$
- Es auch, dass $1\underline{00}1 \in L$.
- Der Automat hat eine Schleife bei $\hat{\delta}(q_1, 00) = q_1$, die mehrfach „abgelaufen” werden kann ohne die Akzeptanz zu beinflussen.
- Also gilt auch $100001 \in L$,
- und im Allgemeinen $\forall i \in \mathbb{N} : 1(00)^i 1 \in L$

Lemma 2.6 (Pumping Lemma)**:** Sei L eine reguläre Sprache. Dann gilt:

$$\begin{aligned} &\exists n \in \mathbb{N},\ n > 0 \quad \forall z \in L,\ |z| \geq n : \\ &\exists u, v, w \in \Sigma^* : \\ &z = uvw,\ |uv| \leq n,\ |v| \geq 1 \\ \text{sodass } &\forall i \in \mathbb{N} :\ uv^i w \in L \end{aligned}$$

BEWEIS: Sei $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ ein DEA für L.
Wähle $n = |Q|$ und $z \in L$ mit $|z| \geq n$.

Beim Erkunden von z durchläuft $\mathcal{A}$ $\underbrace{|z| + 1}_{\geq n+1}$ Zustände.

$\rightarrow \exists\, q$, das mehrmals besucht wird.

Wähle das q, dessen zweiter Besuch zuerst passiert.

$$\begin{aligned}
\text{D.h.}: \quad & \hat{\delta}(q_0, u) = q && u \text{ Präfix von } z \\
\exists v: \quad & \hat{\delta}(q, v) = q && uv \text{ Präfix von } z \\
\exists w: \quad & \hat{\delta}(q, w) \in F && uvw = z \\
& |v| \geq 1 && \\
& |uv| \leq n && \text{ergibt sich aus Wahl von } q
\end{aligned}$$

$$\begin{aligned}
\text{jetzt:} \quad & \hat{\delta}(q_0, uv^i w) && i \in \mathbb{N} \\
& = \hat{\delta}(q, v^i w) && \\
& = \hat{\delta}(q, w) && \text{denn } \forall i : \hat{\delta}(q, v^i) = q \\
& \in F &&
\end{aligned}$$

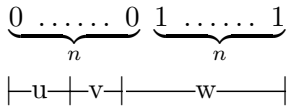
□

Bsp.: $L = \{0^n 1 0^n \mid n \in \mathbb{N}\}$ ist nicht regulär.
Sei n die Konstante aus dem PL.

Wähle $z = 0^n 1 0^n$. Also $|z| = 2n + 1 \geq n$
Laut PL existieren u, v, w, sodass $z = uvw$ mit $|v| \geq 1, |uv| \leq n$ und $\forall i \in \mathbb{N}\ uv^i w \in L$.
Nach Wahl von z gilt nun

- $uv = 0^m$ mit $m \leq n$
- $v = 0^k$ mit $k \geq 1$
- $w = 0^{n-m} 1 0^n$

Betrachte $uv^2w = 0^{m+k} 0^k 0^{n-m} 1 0^n = 0^{n+k} 1 0^n \notin L$. Also ist L nicht regulär. Zur Illustration:



Bsp.: $L = \{0^{x^2} \mid x \in \mathbb{N}\}$ ist nicht regulär.
Sei n die Konstante aus dem PL.

Wähle $z = 0^{n^2}$. Also $|z| = n^2 \geq n$
Laut PL existieren u, v, w, sodass $z = uvw$ mit $|v| \geq 1, |uv| \leq n$ und $\forall i \in \mathbb{N}\ uv^iw \in L$.
Nach Wahl von z gilt nun

- $uv = 0^m$ mit $m \leq n$
- $v = 0^k$ mit $k \geq 1$
- $w = 0^{n^2-m}$ mit $k \geq 1$

Betrachte $uv^2w = uvvw = 0^m0^k0^{n^2-m} = 0^{n^2+k}$. Da $n^2 + k$ keine Quadratzahl sein kann ist $uv^2w \notin L$, und somit ist L nicht regulär. Begründung: betrachte $(n+1)^2 - n^2 = n^2 + 2n + 1 - n^2 = 2n + 1$. Aber $k \leq m \leq n \leq 2n + 1$.

Bsp.: $L_2 = \{0^p \mid p \text{ ist Primzahl}\}$ ist nicht regulär.

Sei n Konst. aus dem PL, p Primzahl mit $p \geq n$.
Wähle $z = 0^p \in L_2$

$$
\begin{aligned}
&\text{PL} : z = uvw \text{ mit } |uv| \leq n && , |v| \geq 1 \\
&\quad \curvearrowright |z| = p = a + b && , a = |uw| \quad , b = |v| \\
&\quad \curvearrowright |uv^iw| = a + ib && , \text{wähle } i = p + 1 \\
&\quad \curvearrowright |uv^{p+1}w| = a + (p+1)b && = a + pb + b = p + pb \text{ keine Primzahl} \\
&\text{Also } uv^{p+1}w \notin L_2 \\
&\quad \curvearrowright L_2 \text{ nicht regulär.}
\end{aligned}
$$

2.4 nichtdeterministischer endlicher Automat (NEA) (Vortsetzung)

Entwurf für Vorlesung: 2.11.16

Bsp.: Mustererkennung
kommt ein String (konsistent) in einem anderen vor?

Gegeben: festes Wort w.
Gesucht: Sprache aller Worte, in denen w als Teilwort vorkommt.

$$L = \{v \in \Sigma^* \mid \exists u, x \in \Sigma^*, v = uwx\}$$
$$\Sigma = \{a, b, c\}$$
konkretes Beispiel:
$$w = abac$$

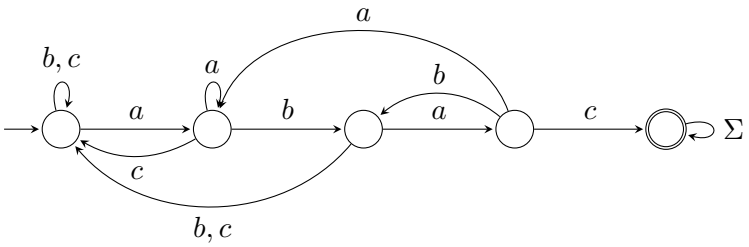


Abb. 3: DEA für L

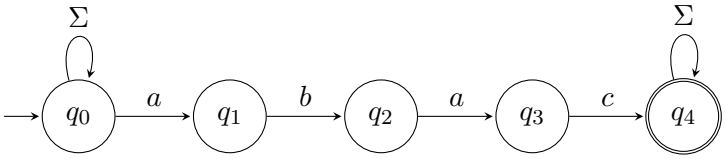


Abb. 4: Bsp.: Mustererkennung

Abbildung 3 enthält einen DEA für die Sprache L. Beobachtung: nicht-trivial zu konstruieren.

Abbildung 4 enthält einen nicht-deterministischen endlichen Automat für die Sprache L. Idee: Ein Wort w wird akzeptiert, falls es einen mit w markierten Pfad von q_0 zu einen akzeptierenden Zustand gibt.

Abbildung 5 zeigt (einen Ausschnitt) aus dem deterministischen Automaten, der schematisch aus dem NEA in Abb. 4 konstruiert werden kann. Idee: bei Schritt mit Symbol a ist der NEA gleichzeitig in allen Zuständen, die durch a von (der Menge der) aktuellen Zustände erreichbar sind.

Variante: erkenne **Subwort** $w = a_1, \ldots, a_n$

$$L' = \{v \in \Sigma^* \mid \exists x_0, \ldots, x_n \in \Sigma^*, v = x_0 a_1 x_1 a_2 \ldots a_n x_n\}$$

Nicht det. Automat für L' mit $(w = abac)$ ist sehr einfach. Der entsprechende deterministische Automat ist deutlich komplizierter. (selbst)

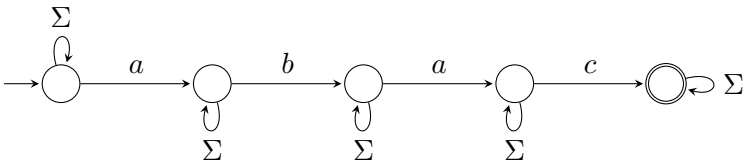


Abb. 6: Nichtdet. Automat für L'

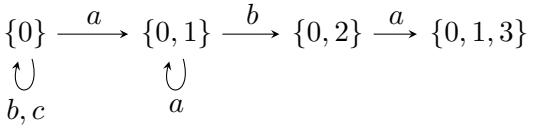


Abb. 5: Potenzmengenkonstruktion auf dem NEA

Weiteres Beispiel, bei dem der deterministische Automat beweisbar exponentiell größer ist.

$$L_n = \{w \in \{0,1\}^* \mid \text{das } n\text{-letzte Symbol von } w \text{ ist } 1\}$$

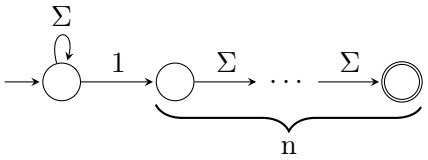


Abb. 7: Nichtdet. Automat für L_n

Der entsprechende deterministische Automat für L_n hat $\sim 2^n$ Zustände.

Def. 2.8: Ein NEA (NEA = nichtdeterministischer endlicher Automat) $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ mit

- Q endliche Zustandsmenge
- Σ endl. Alphabet
- $\delta : Q \times \Sigma \to \mathcal{P}(Q)$ Transitionsfunktion
- $q_0 \in Q$ Startzustand
- $F \subseteq Q$ akzeptierende Zustände

$\oplus$

Def. 2.9: Ein *Lauf des nichtdet. Automaten* $\mathcal{A}$ *auf* $w = a_1 \dots a_n$ ist eine Folge $q_0 q_1 \dots q_n$ mit $q_i \in Q$, wobei q_0 der Startzustand ist und $q_i \in \delta(q_{i-1}, a_i)$ für alle i mit $1 \leq i \leq n$. Ein Lauf heißt *akzeptierend*, falls $q_n \in F$. $\oplus$

Def. 2.10: $L(\mathcal{A}) = \{w \in \Sigma^* \mid \exists \text{ akzeptierender Lauf von } \mathcal{A} \text{ auf } w\}$ $\oplus$

Satz 2.7 (Rabin)**:** Zu jedem NEA $\mathcal{A}$ mit n Zuständen gibt es einen DEA $\mathcal{A}'$ mit 2^n Zuständen, so dass $L(\mathcal{A}) = L(\mathcal{A}')$.

Entwurf für Vorlesung: 4.11.16

BEWEIS (Potenzmengenkonstruktion): Definiere $\mathcal{A}'$ durch

$$\begin{aligned} Q' &= \mathcal{P}(Q) \\ \delta'(q', a) &= \bigcup_{q \in q'} \delta(q, a) \\ q_0' &= \{q_0\} \\ F' &= \{q' \in Q' \mid q' \cap F \neq \varnothing\} \end{aligned}$$

Zeige $L(\mathcal{A}) = L(\mathcal{A}')$

$$\begin{aligned} \text{Es gilt } w \in L(\mathcal{A}') &\Leftrightarrow \hat{\delta}'(q_0', w) \in F' \\ &\Leftrightarrow \hat{\delta}'(q_0', w) \cap F \neq \varnothing \\ &\Leftrightarrow \exists \text{ akzeptierender Lauf von } \mathcal{A} \text{ auf } w \\ &\Leftrightarrow w \in L(\mathcal{A}). \end{aligned}$$

Zeige $\forall w \in \Sigma^*$:

$$\begin{aligned} \forall q' \in Q' \quad & \hat{\delta}'(q', w) \cap F \neq \varnothing \\ \Leftrightarrow\ & \exists \text{ akzeptierender Lauf von } \mathcal{A} \text{ ab } q' \text{ auf } w \\ \Leftrightarrow\ & \exists \underbrace{p_0 p_1 \dots p_n}_{\text{Lauf}} \in Q \quad p_0 = q',\ p_n \in F,\ n = |w| \end{aligned}$$

Induktion nach w

I.A.

$$\begin{aligned} & \varepsilon: \\ & \quad \forall q' \in Q'\ \hat{\delta}(q', \varepsilon) \cap F \neq \varnothing \\ \Leftrightarrow & \qquad\qquad q' \cap F \neq \varnothing \end{aligned}$$

Wähle einen beliebigen Zustand $p_0 = p_n \in q' \cap F$

I.S.

$$\begin{aligned} & aw' \\ & \quad \hat{\delta}'(q', aw') \cap F \neq \varnothing \\ \Leftrightarrow & \quad \hat{\delta}'(\underbrace{\delta'(q', a)}_{\in Q'}, w') \cap F \neq \varnothing \\ \overset{\text{I.V.}}{\Longleftrightarrow} & \quad \exists \text{Lauf } p_1 \dots p_{n+1} \in Q : p_0 \in \hat{\delta}'(q', a),\ p_{n+1} \in F \end{aligned}$$

Suche $p_0 \in q'$ mit $p_1 \in \delta(p_0, a)$
existiert, denn

$$\begin{aligned} & p_1 \in \delta'(q', a) = \bigcup_{q \in q'} \delta(q, a) \\ \Leftrightarrow \quad & \exists p_0 \in q' : \delta(p_0, a) \ni p_1 \end{aligned}$$

Gesuchter Lauf ist

$$p_0 p_1 \dots p_{n+1}$$

□

Also: Eine Sprache L ist regulär, falls

- $L = L(\mathcal{A})$ für einen DEA
  oder
- $L = L(\mathcal{A})$ für NEA

2.5 Abschlusseigenschaften

Def. 2.11: Eine Menge $\mathcal{L} \subseteq \mathcal{P}(\Sigma^*)$ von Sprachen heißt *abgeschlossen* unter Operation $f : \mathcal{P}(\Sigma^*)^n \to \mathcal{P}(\Sigma^*)$ falls $\forall L_1, \dots, L_n \in \mathcal{L} : f(L_1, \dots, L_n) \in \mathcal{L}$. ⊕

Satz 2.8: Die Menge REG der regulären Sprachen ist abgeschlossen unter $\cup$ (Vereinigung), $\cap$ (Durchschnitt), $\overline{\ }$ (Komplement), Produkt (Konkatenation), Stern. Beispielsweise ist für L_1, L_2 reguläre Sprachen also auch $L_1 \cup L_2$, wie auch $L_1 \cap L_2$ etc. wieder eine reguläre Sprache.

Beweis: Sei $\mathcal{A}_i := (Q_i, \Sigma, \delta_i, q_{0i}, F_i), \quad i = 1, 2$ NEAs

- $\cup$: Def $\mathcal{A}$ durch (vgl. Abb. 8):

$$\begin{aligned} Q &= Q_1 \dot\cup Q_2 \dot\cup \{q_0\} \\ \delta(q, a) &= \begin{cases} \delta_1(q, a) & q \in Q_1 \\ \delta_2(q, a) & q \in Q_2 \\ \delta_1(q, a) \cup \delta_2(q, a) & q = q_0 \end{cases} \\ F &= F_1 \dot\cup F_2 \dot\cup (q_{01} \in F_1 \vee q_{02} \in F_2) \rhd \{q_0\} \end{aligned}$$

  Zeige $L(\mathcal{A}) = L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$ (Aufgabe zum Eigenstudium: Betrachte die Läufe).

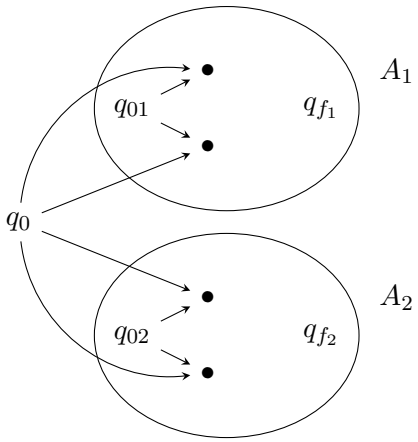


Abb. 8: NEA für Vereinigung

- $\cap$: Annahme: Seien $\mathcal{A}_1$ und $\mathcal{A}_2$ zwei DEAs. Konstruiere nun den *Produktautomaten* $\mathcal{A}$, für den gilt (Aufgabe zum Eigenstudium!): $L(\mathcal{A}) = L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$.

$$\begin{aligned} Q &= Q_1 \times Q_2 \\ \delta((q_1, q_2), a) &= (\delta_1(q_1, a), \delta_2(q_2, a)) \\ q_0 &= (q_{01}, q_{02}) \\ F &= F_1 \times F_2 \end{aligned}$$

- Komplement: Ang. $\mathcal{A}_1$ ist DEA, der L erkennt. Ersetze F_1 durch $Q_1 \setminus F_1$ und erhalte einen DEA $\mathcal{A}_1'$, der genau das Komplement von L erkennt.
- Produkt: Seien L_1, L_2 regulär.
  Zeige $L_1 \cdot L_2$ regulär.

$$\begin{aligned} Q &= Q_1 \dot\cup Q_2 \\ \delta(q, a) &= \begin{cases} \delta_1(q, a) & q \in Q_1 \setminus F_1 \\ \delta_1(q, a) \cup \delta_2(q_{02}, a) & q \in F_1 \\ \delta_2(q, a) & q \in Q_2 \end{cases} \\ q_0 &= q_{01} \\ F &= F_2 \cup (q_{02} \in F_2) \rhd F_1 \end{aligned}$$

  Zeige $L(\mathcal{A}) = L(\mathcal{A}_1) \cdot L(A_2)$

- Stern

$$
\begin{aligned}
Q &= Q_1 \dot\cup \{q_0\} \\
\delta(q,a) &= \begin{cases} \delta_1(q,a) & q \in Q_1 \setminus F_1 \\ \delta_1(q,a) \cup \delta_1(q_{01},a) & q \in F_1 \\ \delta_1(q_{01},a) & q = q_0 \end{cases} \\
F &= \{q_0\} \cup F_1 \\
\ldots\ L(\mathcal{A}) &= L(\mathcal{A}_1)^*
\end{aligned}
$$

□

2.6 Reguläre Ausdrücke

Entwurf für Vorlesung: 9.11.16

Def. 2.12: Die Menge $RE(\Sigma)$ der *regulären Ausdrücke über* Σ ist induktiv definiert durch:

- $\mathbf{0} \in RE(\Sigma)$
- $\mathbf{1} \in RE(\Sigma)$
- $\forall a \in \Sigma,\ a \in RE(\Sigma)$
- falls $r, s \in RE(\Sigma)$
  - $r + s \in RE(\Sigma)$
  - $r \cdot s \in RE(\Sigma)$
  - $r^* \in RE(\Sigma)$

⊕

Def. 2.13: Die Semantik eines regulären Ausdrucks $[\![\cdot]\!] : RE(\Sigma) \to \mathcal{P}(\Sigma^*)$ ist induktiv definiert durch

$$
\begin{aligned}
[\![\mathbf{0}]\!] &= \varnothing \\
[\![\mathbf{1}]\!] &= \{\varepsilon\} \\
[\![a]\!] &= \{a\} \quad a \in \Sigma \\
[\![r + s]\!] &= [\![r]\!] \cup [\![s]\!] \\
[\![r \cdot s]\!] &= [\![r]\!] \cdot [\![s]\!] \\
[\![r^*]\!] &= [\![r]\!]^*
\end{aligned}
$$

⊕

Bsp.: Mustererkennung

- Akzeptiere alle Wörter, die *abac* enthalten:

$$\begin{aligned}&\Sigma^* abac \Sigma^* \\ \Leftrightarrow &(a_1 + a_2 + ...)^* abac (a_1 + a_2 + ...)^* \forall a_i \in \Sigma\end{aligned}$$

- Sei $\Sigma = \{0, 1\}$. Die Sprache aller Wörter, deren n-letztes Symbol $= 1$, ist nicht regulär, denn:

$$(0+1)^* 1 \underbrace{(0+1)\dots(0+1)}_{n-1}$$
$$\cancel{0^n 1^n} \notin RE(\Sigma)$$

- Ein regulärer Ausdrück für alle Wörter $\omega \in (0,1)^*$, s.d. $\omega \mod 3 = 0$:

$$(0 + 1(01^*0)^*1)^*$$

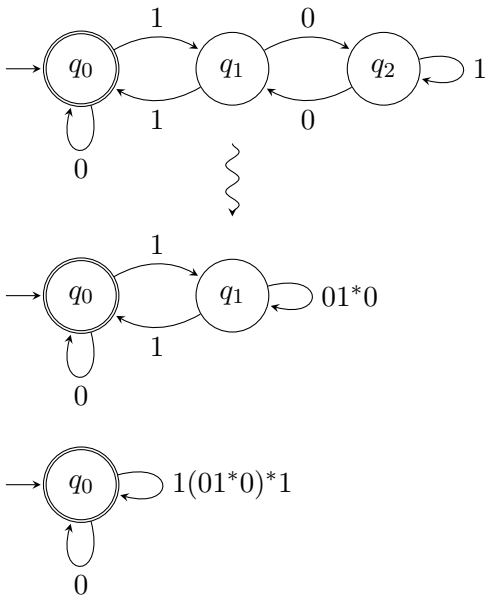


Abb. 9: Informell vom Automaten zum regulären Ausdruck für mod 3

Satz 2.9 (Kleene)**:** L ist regulär $\Leftrightarrow$ L ist Sprache eines regulären Ausdrucks.

Beweis (Kleene, $\Leftarrow$): Betrachte zu einem regulärem Ausdruck $r \in RE(\Sigma)$ die durch diesen erzeugte Sprache $L = [\![r]\!]$. Zeige per Induktion über r, dass $\forall r \in RE(\Sigma)$ gilt: $[\![r]\!]$ ist regulär.

I.A.:
- $r = \mathbf{0}$, $[\![r]\!] = \emptyset$ ist regulär
- $r = \mathbf{1}$, $[\![r]\!] = \{\varepsilon\}$ ist regulär
- $r = a$, $[\![r]\!] = \{a\}$ ist regulär
  ( NEA)

I.V.: Für $i \in \{1, 2\}$ gilt: $[\![r_i]\!]$ ist regulär.

I.S.:
- $r = r_1 + r_2$, $[\![r]\!] = [\![r_1]\!] \cup [\![r_2]\!]$ ist regulär nach Satz 2.8
- $r = r_1 \cdot r_2$, $[\![r]\!] = [\![r_1]\!] \cdot [\![r_2]\!]$ ist regulär nach Satz 2.8
- $r = r_1^*$, $[\![r]\!] = [\![r_1]\!]^*$ ist regulär nach Satz 2.8

□

Zum Beweis (bzw. zur Konstruktion) der Richtung „⇒" benötigen wir eine Rechenregel zum Lösen von Gleichungen zwischen regulären Sprachen und Ausdrücken:

Lemma 2.10 (Ardens Lemma)**:**
Sei die lineare Gleichung $X = A \cdot X + B$ über $A, B, X \subseteq \Sigma^*$ gegeben. Dann ist $X = A^*B$ eine Lösung. Falls $\varepsilon \notin A$ ist diese Lösung ausserdem eindeutig.

BEWEIS: Sei $X = AX + B$ mit $\varepsilon \notin A$. Zeige, dass $A^*B \subseteq X$:

$$A^*B = AA^*B + B = (\mathbf{1} + AA^*)B = B + A(A^*B) \quad \checkmark$$

Angenommen $A^*B \subsetneq X$, d.h. $\exists w \in X$ mit $w \notin A^*B$, davon sei w das kürzeste.

$$\exists n \geq 1: \quad X = \underbrace{A^n X}_{\ni w} + \underbrace{A^{n-1}B + \cdots + AB + B}_{\not\ni w}$$

$$\curvearrowright \quad w = u_1 \dots u_n w' \text{ mit } u_1, \dots, u_n \in A \text{ und } w' \in X$$

$$\curvearrowright \quad |w'| < |w|$$

$$\text{Falls} \quad w' \in A^*B \curvearrowright w \in A^n A^*B \subseteq A^*B \quad \lightning$$

$$\text{Also} \quad w' \notin A^*B \quad \lightning \text{ gegen Minimalität von } w$$

$$\curvearrowright \quad X \subseteq A^*B$$

$$\rightarrow \quad X = A^*B$$

□

Korollar 2.11: Ardens Lemma lässt sich auch für reguläre Ausdrücke formulieren: Seien r_X, r_A, r_B reguläre Ausdrücke mit $\varepsilon \notin [\![r_A]\!]$ für die die folgende Gleichung gilt:

$$[\![r_x]\!] = [\![r_A \cdot r_x + r_B]\!]$$

Dann ist

$$r_x := r_A^* r_B$$

eine eindeutige Lösung für r_x, die die Gleichung erfüllt. ⊕

Vorlesung: 11.11.16

BEWEIS (Kleene, $\Rightarrow$): Sei $L = L(\mathcal{A})$ für einen DEA $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ mit $Q = \{q_0, q_1, \ldots, q_n\}$. Definiere $L_i = \{w \mid \hat{\delta}(q_i, w) \in F\}$ als die Sprache der Worte, die von Zustand q_i aus in einen akzeptierenden Zustand führen. Insbesondere gilt $L_0 = L$.

Wir leiten nun ein lineares Gleichungssystem zwischen den Sprachen L_i her. Zunächst teilen wir Li in den Teil der (potentiell) das leere Wort enthält und den, der die nicht-leeren Wörter enthält.

$$\begin{aligned} L_i &= \{w \mid \hat{\delta}(q_i, w) \in F\} \\ &= \{\varepsilon \mid \hat{\delta}(q_i, \varepsilon) \in F\} \cup \{aw' \mid a \in \Sigma, w' \in \Sigma^*, \hat{\delta}(\delta(q_i, a), w') \in F\} \\ &= \{\varepsilon \mid q_i \in F\} \cup \{aw' \mid a \in \Sigma, w' \in \Sigma^*, \hat{\delta}(\delta(q_i, a), w') \in F\} \end{aligned}$$

Die nicht-leeren Wörter hängen von den Sprachen der Folgezustände ab:

$$\begin{aligned} \{aw' \mid a \in \Sigma, w' \in \Sigma^*, \hat{\delta}(\delta(q_i, a), w') \in F\} &= \bigcup_{a \in \Sigma} \{a\}\{w' \mid a \in \Sigma, w' \in \Sigma^*, \hat{\delta}(\delta(q_i, a), w') \in F\} \\ &= \bigcup_{a \in \Sigma} \{a\}(L_j \text{ wobei } q_j = \delta(q_i, a)) \end{aligned}$$

Anstatt die Vereinigung über die Transitionen a und Folgezustandssprachen L_j zu bilden, lassen sich die nicht-leeren Worte von L_i auch als Vereinigung über aller Zustände mit entsprechend gewählten *Koeffizienten* formulieren; die Zustände die keine Folgezustände sind, habe den Koeffizienten $\emptyset$.

$$\bigcup_{a \in \Sigma} \{a\}(L_j \text{ wobei } q_j = \delta(q_i, a)) = \bigcup_{j=0}^{n} \underbrace{\{a \mid a \in \Sigma, \delta(q_i, a) = q_j\}}_{A_{ij} \not\ni \varepsilon} L_j$$

Man beachte das die Koeffizienten A_{ij} nie das leere Wort enthalten. Nach diesen Überlegungen ergibt sich also für die Sprachen L_i das lineare Gleichungssystem

$$L_i = \{\varepsilon \mid q_i \in F\} \cup \bigcup_{j=0}^{n} \underbrace{\{a \mid a \in \Sigma, \delta(q_i, a) = q_j\}}_{A_{ij} \not\ni \varepsilon} L_j$$

Diese Gleichungen lassen sich analog als Gleichungen von regulären Ausdrücken r_i formulieren:

$$r_i = N(q_i) + \sum_{j=0}^{n} R_{ij} r_j$$

wobei $[\![r_i]\!] = L_i$ und $R_{ij} = \sum\{a \mid a \in \Sigma, \delta(q_i, a) = q_j\}$ mit $\varepsilon \notin [\![R_{ij}]\!]$ und

$$N(q_i) = \begin{cases} \mathbf{1} & q_i \in F \\ \mathbf{0} & q_i \notin F \end{cases}$$

Dieses Gleichungssystem lässt sich, wie lineare Gleichungssysteme in der Arithmetik, mit dem *Substituitionsverfahren* lösen („auflösen nach einer Variablen und einsetzen"). Dazu werden Ardens Lemma und weitere Rechenregeln für Sprachen, wie Distributivität, verwendet. Wir beginnen mit Gleichung r_n:

$$\begin{aligned} r_n &= N(q_n) + \sum_{j=0}^{n} R_{nj} r_j \\ &= \underbrace{N(q_n) + \left(\sum_{j=0}^{n-1} R_{nj} r_j \right)}_{r_B} + \underbrace{R_{nn}}_{r_A} r_n \end{aligned}$$

Wie oben angedeutet ist nach dem Herausziehen des $n$ten Summenglieds Ardens Lemma anwendbar (merke, $\varepsilon \notin [\![R_{nn}]\!]$), und wir setzen:

$$r_n := R_{nn}^* \left(N(q_n) + \sum_{j=0}^{n-1} R_{nj} r_j \right)$$

Dieses Ergebnis in $r_0, \ldots, r_{n-1}$ eingesetzt ergibt:

$$r_i = N(q_i) + \left(\sum_{j=0}^{n-1} R_{nj} r_j \right) + R_{in} R_{nn}^* \left(N(q_n) + \sum_{j=0}^{n-1} R_{nj} r_j \right)$$

(Ausmultiplizieren von $R_{in} R_{nn}^*$)

$$= N(q_i) + \left(\sum_{j=0}^{n-1} R_{nj} r_j \right) + R_{in} R_{nn}^* N(q_n) + \sum_{j=0}^{n-1} R_{in} R_{nn}^* R_{nj} r_j$$

(Zusammenlegen der Summen und Ausklammern von r_j)

$$= N(q_i) + R_{in} R_{nn}^* N(q_n) + \sum_{j=0}^{n-1} (R_{nj} + R_{in} R_{nn}^* R_{nj}) r_j$$

Nach diesen Umformungen ergeben sich ε-freie Koeffizienten $R_{nj} + R_{in} R_{nn}^* R_{nj}$ für r_j und wir mit dem Auflösen von $n-1$ analog zu n fortfahren. Am Ende erhalten wir einen regulären Ausdruck als Lösung für r_0. Per Konstruktion haben wir immer noch $[\![r_0]\!] = L_0 = L$. □

Bsp.: Ein Beispiel für die Konvertierung eines DEA in einen reg. Ausdruck.

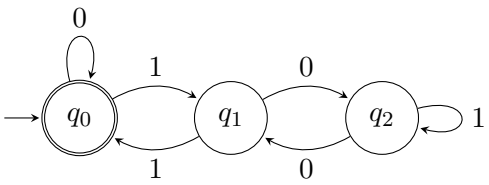


Abb. 10: DEA „modulo 3“

lineares Gleichungssystem mit 3 Unbekannten.

$$\begin{aligned} L_0 &= \mathbf{1} + 0 \cdot L_0 + 1 \cdot L_1 \\ L_1 &= 1 \cdot L_0 + 0 \cdot L_2 \\ L_2 &= \underbrace{0 \cdot L_1}_{B} + \underbrace{1}_{A} \cdot L_2 \end{aligned}$$

Ardens Lemma auf q_2:

$$L_2 = 1^* \cdot 0 \cdot L_1$$

Einsetzen in q_1

$$L_1 = \underbrace{1 \cdot L_0}_{B} + \underbrace{0 \cdot 1^* \cdot 0}_{A} \cdot L_1$$

Ardens Lemma auf q_1:

$$L_1 = (01^*0)^* \cdot 1 \cdot L_0$$

Einsetzen:

$$\begin{aligned} L_0 &= \mathbf{1} + 0 \cdot L_0 + 1 \cdot (01^*0)^* \cdot 1 \cdot L_0 \\ &= \mathbf{1} + (0 + 1 \cdot (01^*0)^* \cdot 1) \cdot L_0 \end{aligned}$$

Ardens Lemma auf q_0:

$$L_0 = (0 + 1 \cdot (01^*0)^* \cdot 1)^*$$

Entwurf für Vorlesung: 11.11.16

2.7 Entscheidungsprobleme

Ein Problem ist *entscheidbar* wenn es sich durch eine binäre Antwort (ja/nein) lösen lässt und es einen Algorithmus gibt, der für alle Instanzen des Problem die korrekte Antwort liefert.

Satz 2.12: Das Wortproblem ist für reguläre Sprachen entscheidbar.

D.h. Falls L reg. Sprache und $w \in \Sigma^*$, dann ex. Algorithmus, der entscheidet, ob $w \in L$.

BEWEIS: L sei durch DEA gegeben.
Berechnung von $\hat{\delta}(q_0, w)$ entspricht Durchlauf durch Graph des DEA + Test ob erreichter Zustand $\in F$ in Zeit $O(n)$, $n = |w|$. □

Satz 2.13: Das *Leerheitsproblem* ist für reg. Sprachen entscheidbar. Falls L reg. Sprache, dann existiert ein Algorithmus, der entscheidet, ob $L = \varnothing$.

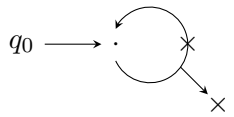
BEWEIS: Sei $\mathcal{A}$ DEA für L.
Setze Tiefensuche auf den Graphen von $\mathcal{A}$ an. Start bei q_0.
Falls die Suche einem akzeptierenden Zustand findet: Nein.
Ansonsten: Ja: $L = \varnothing$
Zeit: $O(|\Sigma||Q|)$ □

Satz 2.14: Das Endlichkeitsproblem für reg. Sprachen ist entscheidbar.

BEWEIS: Falls L durch $r \in RE(\Sigma)$ gegeben.
r enthält keinen $^* \Rightarrow [\![r]\!]$ endlich.
Zeit: $O(|r|)$

[Reicht nicht, liefert nur eine Richtung]

Falls L durch DEA $\mathcal{A}$ gegeben.



Oder: mit Pumping Lemma.
Sei L regulär und n die Konstante aus dem PL.
L unendlich $\Leftrightarrow \exists w \in L : n \leq |w| < 2n$

"$\Leftarrow$": w erfüllt Voraussetzung des PL, also $w = uvx$ mit $|uv| \leq n$ und $|v| \geq 1$.
Nach PL: $\forall i \in \mathbb{N}$, $uv^i x \in L$, also L unendlich.

"$\Rightarrow$" *Angenommen* L unendlich, aber $\forall w \in L : |w| < n$ oder $|w| \geq 2n$
Sei $w \in L$ minimal gewählt, so dass $|w| \geq 2n$.

w erfüllt Voraussetzung vom PL, also $w = xyz$ mit $|xy| \leq n$ und $|y| \geq 1$
also $\forall i \in \mathbb{N} : xy^i z \in L$ insbes. $i = 0 : xz \in L$ mit $|xz| < |w|$.

Zwei Möglichkeiten:

(a) $|xz| \geq 2n$ ↯ Minimalität von w

(b) $|xz| < 2n$

$$\begin{aligned} &|xz| + |y| = |w| \geq 2n \text{ mit } 1 \leq |y| \leq n \\ &\Rightarrow |xz| = |w| - |y| \geq 2n - n = n \\ &\Rightarrow |xz| \geq n \wedge |xz| < 2n \quad \lightning \text{zur Annahme} \end{aligned}$$

Also $\exists w \in L$ mit $n \leq |w| < 2n$ □

Satz 2.15: Das *Schnittproblem* ist für REG entscheidbar.
D.h. L_1, L_2 reguläre Sprachen. Ist $L_1 \cap L_2 = \varnothing$?

Beweis: Nach Satz 2.15 ist $L_1 \cap L_2$ regulär. $L_1 \cap L_2 = \varnothing$ entscheidbar nach Satz 2.13. □

Satz 2.16: Das *Äquivalenzproblem* ist für REG entscheidbar.
D.h. gegeben DEAs für L_1 und L_2, $\mathcal{A}_1$ und $\mathcal{A}_2$

$$L_1 = L(\mathcal{A}_1) = L(\mathcal{A}_2) = L_2 \; ? \qquad \boxed{\text{Inklusionsproblem}}$$

Beweis:

$$\begin{aligned} L_1 \cap \overline{L}_2 = \varnothing \quad &\Leftrightarrow \quad L_1 \subseteq L_2 \\ (L_1 \cap \overline{L}_2) \cup (L_2 \cap \overline{L}_1) = \varnothing \quad &\Leftrightarrow \quad L_1 = L_2 \end{aligned}$$ □

Satz 2.17: Äquivalenzproblem $\Leftrightarrow$ Inklusionsproblem (für REG)

Beweis:
- $=$ entspricht $\subseteq \wedge \supseteq$
- $L_1 \subseteq L_2$ genau dann, wenn $L_1 \cup L_2 = L_2$; REG ist abgeschlossen unter Vereinigung

□

Anwendungsbeispiel für reguläre Sprachen.

N – liest vom Netz
R – liest lokalen Speicher (ggf. vertrauliche Info)
W – postet auf FB

Programm:

$p = N|R|W|$ if $*$ then p_1 else p_2
| while $*$ do p

while $*$ do N;
R;
if $*$ then N else W

$\Big\}\ N^*R \cdot (N + W)$

Sicherheitspolitik: nach Lesen von lokalem Speicher kein Posten auf FB $\overline{\Sigma^* R \Sigma^* W \Sigma^*}$

Programm erfüllt Sicherheitspolitik nicht, denn

$$\begin{array}{c} N^* \cdot R(N+W) \not\subseteq \overline{\Sigma^* R \Sigma^* W \Sigma^*} \\ \cup\!| \\ N^* RW \end{array}$$

3 Grammatiken und kontextfreie Sprachen

Vorlesung: 18.11.16

Im folgenden Kapitel wechseln wir den Standpunkt von Spracherkennung auf *Sprachererzeugung*. Das Werkzeug sind hierbei sogenannte *Phasenstrukturgrammatiken*, oder kurz, *Grammatiken*. Bei Grammatiken gibt es neben dem Alphabet weitere Symbole, sogenannte *Nichtterminale* oder *Variablen*, und ein Regelsystem mit dem Worte, die Nichtterminale enthalten, geändert werden können.

Def. 3.1: Eine *Grammatik* ist ein 4-Tupel (N, Σ, P, S)

- N ist eine endliche Menge von *Nichtterminalsymbolen* (Variablen).
- Σ ist ein Alphabet von *Terminalsymbolen*.
- $P \subset (N \cup \Sigma)^* N (N \cup \Sigma)^* \times (N \cup \Sigma)^*$ ist endliche Menge von Regeln, sogenannte *Produktionen*.
- $S \in N$ ist das *Startsymbol*.

$\oplus$

Bsp. 3.1: $\mathcal{G} = (N, \Sigma, P, E)$ mit

$$\begin{aligned} \Sigma &= \{\mathtt{a}, (,), *, +\} \\ N &= \{E\} \\ P &= \{E \to a, \\ &\qquad E \to (E * E), \\ &\qquad E \to (E + E)\} \end{aligned}$$

G erzeugt geklammerte arithmetische Ausdrücke über der Konstante $\mathtt{a}$, wie zum Beispiel das Wort $(\mathtt{a} * (\mathtt{a} + \mathtt{a}))$. Eine Grammatik erzeugt Worte durch *Ableitungen*, die wir im Folgenden definieren.

Def. 3.2 (Ableitungsrelation, Ableitung, Sprache einer Grammatik)**:** Sei $\mathcal{G} = (N, \Sigma, P, S)$ eine Grammatik.

Die *Ableitungsrelation* zu $\mathcal{G}$ ist

$$\cdot \Longrightarrow_{\mathcal{G}} \cdot \subseteq (N \cup \Sigma)^* \times (N \cup \Sigma)^*$$

mit $\alpha \Longrightarrow_{\mathcal{G}} \beta$ gdw

- $\alpha = \gamma_1 \alpha' \gamma_2$,
- $\beta = \gamma_1 \beta' \gamma_2$ und

- $\alpha' \to \beta' \in P$

Eine Folge $\alpha = \alpha_0, \ldots, \alpha_n = \beta$ heißt *Ableitung von β aus α in n Schritten*, geschrieben $\alpha \overset{n}{\Longrightarrow}_{\mathcal{G}} \beta$, gdw $\alpha_i \Longrightarrow_{\mathcal{G}} \alpha_{i+1}$ für $0 \leq i < n$. Jedes solche α_i heißt *Satzform von $\mathcal{G}$*.

Die *Ableitung von β aus α*, geschrieben $\alpha \overset{*}{\Longrightarrow}_{\mathcal{G}} \beta$, existiert gdw ein $n \in \mathbb{N}$ existiert, so dass $\alpha \overset{n}{\Longrightarrow}_{\mathcal{G}} \beta$. Damit ist „$\overset{*}{\Longrightarrow}_{\mathcal{G}}$" die reflexive, transitive Hülle von „$\Longrightarrow_{\mathcal{G}}$".

Die *Sprache, die von $\mathcal{G}$ erzeugt wird* ist definiert als:

$$L(\mathcal{G}) = \{w \in \Sigma^* \mid S \overset{*}{\Longrightarrow}_{\mathcal{G}} w\}$$

⊕

Entwurf für Vorlesung: 18.11.16

Bsp. 3.2:

$$\begin{aligned} E &\Rightarrow a \\ E &\Rightarrow (E * E) \Rightarrow (a * E) \\ &\Rightarrow (a * (E + E)) \Rightarrow \ldots \Rightarrow (a * (a + a)) \end{aligned}$$

Bsp. 3.3: $\mathcal{G} = (N, \Sigma, P, S)$ mit

$$\begin{aligned} N &= \{S, B, C\} \\ \Sigma &= \{a, b, c\} \\ 8P &= \{S \to aSBC,\ S \to aBC,\ CB \to BC,\ aB \to ab, \\ &\quad bB \to bb, bC \to bc, cC \to cc\} \qquad \text{Startsymbol } S \\ L(\mathcal{G}) &= \{a^n b^n c^n \mid n \geq 1\} \end{aligned}$$

$S \Rightarrow aBC \Rightarrow abC \Rightarrow abc \qquad S \Rightarrow aSBC \Rightarrow aaSBCBC$

$S \Rightarrow aSBC \overset{n-2}{\Longrightarrow} n \Rightarrow a^{n-1}S(BC)^{n-1} \Rightarrow a^n(BC)^n = a^nBCBC\ldots \Rightarrow a^nBBCCBC$

$\Rightarrow a^nB^nC^n \overset{*}{\Rightarrow} a^nb^nc^n$

Die Chomsky-Hierarchie teilt die Grammatiken in vier Typen unterschiedlicher Mächtigkeit ein.

Def. 3.3 (Chomsky Hierarchie)**:**

- Jede Grammatik ist eine *Typ-0 Grammatik*.

- Eine Grammatik ist *Typ-1* oder *kontextsensitiv*, falls alle Regeln expansiv sind, d.h., für alle Regeln $\alpha \to \beta \in P$ ist $|\alpha| \leq |\beta|$. Ausnahme: falls S nicht in einer rechten Regelseite auftritt, dann ist $S \to \varepsilon$ erlaubt.
- Eine Grammatik heißt *Typ-2* oder *kontextfrei*, falls alle Regeln die Form $A \to \alpha$ mit $A \in N$ und $\alpha \in (N \cup \Sigma)^*$ haben.
- Eine Grammatik heißt *Typ-3* oder *regulär*, falls alle Regeln die Form

$$\begin{array}{ll} \text{Form} & A \to w \quad w \in \Sigma^* \\ \text{oder} & A \to aB \quad a \in \Sigma,\ B \in N \end{array}$$

Eine Sprache heißt Typ-i Sprache, falls $\exists$ Typ-i Grammatik für sie. $\oplus$

Beobachtung: Jede Typ-$i+1$ Sprache ist auch Typ-i Sprache.

Jede Typ-3 Grammatik ist Typ-2 Grammatik.
Jede Typ-2 Grammatik kann in äquivalente ε-freie Typ-2 Grammatik transformiert werden. $\to$ Typ-1 Grammatik. (Vgl. Elimination von ε-Produktionen)
Jede Typ-1 Grammatik ist auch Typ-0 Grammatik.

Ziel: Hierarchie-Satz (Chomsky)
Sei $\mathcal{L}_i$ die Menge der Typ-i Sprachen.
Es gilt $\mathcal{L}_3 \subsetneq \mathcal{L}_2 \subsetneq \mathcal{L}_1 \subsetneq \mathcal{L}_0$

Entwurf für Vorlesung: 25.11.16

Satz 3.1: L ist regulär $\Leftrightarrow$ L ist Type-3 Sprache

Beweis: „$\Rightarrow$“ Sei $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ DFA für L.
Konstruiere Grammatik $\mathcal{G} = (N, \Sigma, P, S)$

$$\begin{array}{ll} N = Q & S = q_0 \\ q \in Q\ \forall a : \delta(q,a) = q' & \curvearrowright\ q \to aq' \in P \\ q \in F & q \to \varepsilon \in P \end{array}$$

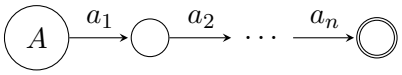
Zeige noch $L(\mathcal{G}) = L(\mathcal{A})$

„$\Leftarrow$“ Sei $\mathcal{G} = (N, \Sigma, P, S)$ Typ-3 Grammatik Def. $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ nondeterministic

finite automaton (NFA)

$$\begin{aligned} Q &= N \cup \dots \\ q_0 &= S \\ A \to aB \in P &\qquad \delta(A,a) \ni B \\ A \to a, \dots a_n &\qquad n = 0 \Rightarrow A \in F \end{aligned}$$

$n > 0$: n weitere Zustände erforderlich, letzter Endzustand



Zeige noch $L(\mathcal{G}) = L(\mathcal{A})$ □

Lemma 3.2: $\mathcal{L}_3 \subsetneq \mathcal{L}_2$

Beweis: Betrachte $L = \{a^n b^n \mid n \in \mathbb{N}\}$
Bekannt, dass L nicht regulär. Aber es gibt eine Typ-2 Grammatik für L:

$$\mathcal{G} = (\{S\}, \{a,b\}, \{S \to \varepsilon, S \to aSb\}, S)$$

Korrektheitsbeweis $L = L(\mathcal{G})$ ist dem Leser zur Übung selbst überlassen. □

Vorlesung: 30.11.16

3.1 Kontextfreie Sprachen

Hier sind einige Beispiele kontextfreier Sprachen und Grammatiken:

- Arithmetische Ausdrücke: $(\{E\}, \{a, +, *, (,)\}, P, E)$ mit

$$\begin{aligned} P = \{&E \to a, \\ &E \to (E + E), \\ &E \to (E * E)\} \end{aligned}$$

- Syntax von Programmiersprachen

<Stmt> → <Var> = <Exp>
| <Stmt> ; <Stmt>
| if (<Exp>) <Stmt>
else <Stmt>
| while (<Exp>) <Stmt>

Hier: Nichtterminal≙ Wort in spitzen Klammern <Stmt>; Terminalsymbol — alles andere.

- Palindrome über $\{a, b\}$
$$S \to aSa \mid bSb \mid a \mid b \mid \varepsilon$$
- Sprache der Worte über $\{0,1\}^*$, die gleich viele Nullen wie Einsen haben:
$$L = \{w \in \Sigma^* \mid \#_0(w) = \#_1(w)\}$$
Die Funktion $\#_a(w)$ berechnet hierbei die Anzahl der Vorkommen von $a \in \{0,1\}$ in w. Eine Grammatik für L ist $\mathcal{G} = (\{S\}, \{0,1\}, P, S)$ mit
$$\begin{aligned} P = \{&S \to 1S0S \\ &S \to 0S1S \\ &S \to \varepsilon\} \end{aligned}$$
Dass $L(\mathcal{G}) \subseteq L$ lässt sich per Induktion über die Länge der Ableitung von $S \overset{*}{\Longrightarrow}_{\mathcal{G}} w$ zeigen. Der Beweis wird als Übung dem Leser überlassen.

Wir zeigen das $L \subseteq L(\mathcal{G})$, also dass wenn $w \in L$, dann $w \in L(\mathcal{G})$ per Induktion über die Länge von w. Hierzu definieren wir die noch die Hilfsfunktion $d : \Sigma^* \to \mathbb{N}$ als
$$\begin{aligned} d(\varepsilon) &= 0 \\ d(1w) &= d(w) + 1 \\ d(0w) &= d(w) - 1 \end{aligned}$$
Per Induktion über $w \in \Sigma^*$ lässt sich zeigen, dass $L = \{w \in \Sigma^* \mid d(w) = 0\}$ und $d(v \cdot w) = d(v) + d(w)$.

IA $|w| = n = 0$, $w = \varepsilon$. Es gilt $\varepsilon \in L$, da $\#_0(\varepsilon) = \#_1(\varepsilon) = 0$.

IV $\forall n' < n : \forall w' \in \Sigma^* : falls |w| = n'$ und $w \in L$ dann $w \in L(\mathcal{G})$

IS $|w| = n > 1$, $w = aw'$, $a \in \{0,1\}$.

Betrachte $a = 0$ (der Fall für $a = 1$ funktioniert analog).

Da $d(w) = d(0w') = d(w') - 1 = 0$ ist $d(w') = 1$.

Es stellt sich heraus, dass $w' = w_1 1 w_2$, $d(w_1) = 0$ und $d(w_2) = 0$.

Sei $w' = a_1 \dots a_n$. Betrachte die Folge $d_0, \dots, d_n$ mit $d_0 = 0$ und $d_i = d(a_1 \dots a_i)$ für $1 \leq i < n$. Wähle $0 \leq i < n$ maximal, so dass für alle $0 \leq j \leq i$: $d_j < 1$.

Da i maximal ist folgt $d_{i+1} \geq 1$ und da $d_{i+1} - d_i \leq 1$ folgt $d_i = 0$, $d_{i+1} = 1$ und $a_i = 1$. Setze also $w_1 = a_1 \dots a_i$.

Es gilt also $w' = a_1 \dots a_i a_{i+1} w_2 = w_1 w_2$ mit $d(w_1) = 0$.

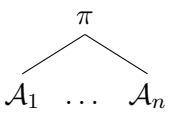
Da $d(v \cdot w) = d(v) + d(w)$ und $d(w') = d(w_1 1 w_2) = 1$ folgt $d(w_2) = 0$.

Da $|w_1| < n$ und $|w_2| < n$, folgt per IV, dass $S \overset{*}{\Longrightarrow}_{\mathcal{G}} w_1$ und $S \overset{*}{\Longrightarrow}_{\mathcal{G}} w_2$.

Es folgt mit den Produktionsregeln $S \Longrightarrow_{\mathcal{G}} 0S1S \overset{*}{\Longrightarrow}_{\mathcal{G}} 0w_1 1S \overset{*}{\Longrightarrow}_{\mathcal{G}} 0w_1 1w_2$.

Def. 3.4: Sei $\mathcal{G} = (N, \Sigma, P, S)$ eine kontextfreie Grammatik. Definiere die Menge der *Ableitungsbäume von $\mathcal{G}$ beginnend mit $A \in N$*, Abl(G, A), als Menge von markierten, geordneten Bäumen induktiv durch:

Falls $\pi = A \to w_0 A_1 w_1 \dots A_n w_n \in P$ mit $A_i \in N$, $w_i \in \Sigma^*$, $0 \le i \le n$ und $\mathcal{A}_i \in \mathrm{Abl}(G, A_i)$ dann ist



$\in \mathrm{Abl}(G, A)$

Abgekürzt: $\pi(\mathcal{A}_1, \dots, \mathcal{A}_n)$

Das *abgeleitete Wort zu einem* $\mathcal{A} \in \mathrm{Abl}(G, A)$, $Y(\mathcal{A})$ („yield" von $\mathcal{A}$), ist definiert durch

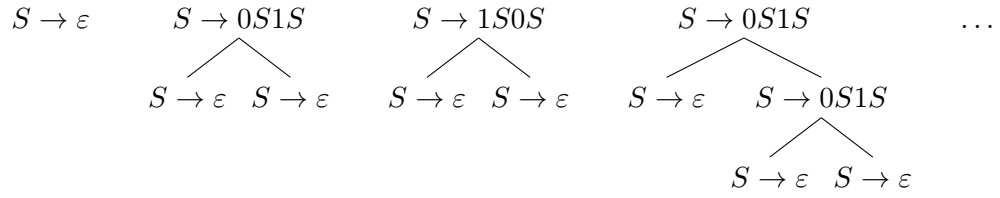
$$Y\left(\begin{array}{c}\pi\\ \mathcal{A}_1 \quad \dots \quad \mathcal{A}_n\end{array}\right) = w_o Y(\mathcal{A}_1) w_1 \dots Y(\mathcal{A}_n) w_n$$

wobei $\pi = A \to w_0 A_1 w_1 \dots A_n w_n \in P$ ⊕

Bsp. 3.4: Einige Ableitungsbäume für die Grammatik $\mathcal{G} = (\{S\}, \{0,1\}, P, S)$ mit

$$\begin{aligned} P = \{&S \to 1S0S \\ &S \to 0S1S \\ &S \to \varepsilon\} \end{aligned}$$

$\mathrm{Abl}(\mathcal{G}, S) =$



Lemma 3.3: Sei $\mathcal{G} = (N, \Sigma, P, S)$ eine kontextfreie Grammatik.

$$w \in L(\mathcal{G}) \text{ gdw } \exists \mathcal{A} \in \mathrm{Abl}(G, S) \text{ mit } Y(\mathcal{A}) = w$$

Beweis („links nach rechts"): Zu zeigen:

$$\forall n \in \mathbb{N} : \forall w \in \Sigma^* : \forall A \in N : A \overset{n}{\Longrightarrow} w \text{ dann } \exists \mathcal{A} \in \mathrm{Abl}(\mathcal{G}, A) \text{ mit } Y(\mathcal{A}) = w$$

Per Induktion über die Länge der Ableitung von $A \overset{n}{\Longrightarrow} w \in \Sigma^*$:

IA $n = 0$ (nichts zu tun, da $A \overset{n}{\Longrightarrow} \varepsilon$ unmöglich)

IS Sei $n > 0$: $A \overset{n}{\Longrightarrow}$ hat die Form

$$A \Longrightarrow w_0 A_1 w_1 \dots A_n w_n \overset{n-1}{\Longrightarrow} w$$

Also ist $w = w_0 v_1 w_1 \dots v_n w_n$ mit $A_i \overset{n_i}{\Longrightarrow} v_i$ für $1 \leq i \leq n$ und $n_i < n$.[†] Nach IV ergibt sich

$$\exists \mathcal{A}_i \in \mathrm{Abl}(G, A_i) \text{ mit } Y(\mathcal{A}_i) = v_i$$

Dann ist $\mathcal{A} = \pi(\mathcal{A}_1, \dots, \mathcal{A}_n)$ mit $\pi = A \to w_0 A_1 w_1 \dots A_n w_n \in \mathrm{Abl}(\mathcal{G}, A)$ und $Y(\mathcal{A}) = w_0 Y(\mathcal{A}_1) w_1 \dots Y(\mathcal{A}_n) w_n = w_0 v_1 w_1 \dots v_n w_n$.

□

Beweis („rechts nach links"): Zu zeigen:

$$\forall n \in \mathbb{N} : \forall \mathcal{A} \in \mathrm{Abl}(\mathcal{G}, A) : \text{falls } Y(\mathcal{A}) = w \text{ dann } A \overset{*}{\Longrightarrow} w$$

IS [††] Falls für $n \geq 0$

- $\mathcal{A} = \pi(\mathcal{A}_1, \dots, \mathcal{A}_n)$,
- $\pi = w_0 A_1 w_1 \dots A_n w_n$,
- $\mathcal{A}_i \in \mathrm{Abl}(\mathcal{G}, A_i)$,
- $Y(\mathcal{A}) = w_0 Y(\mathcal{A}_1) w_1 \dots Y(\mathcal{A}_n) w_n$

dann gilt per IV:[†††]

$$\begin{aligned} A &\Longrightarrow w_0 A_1 \dots A_n w_n \\ &\overset{*}{\Longrightarrow} w_0 Y(\mathcal{A}_1) \dots A_n w_n \\ &\dots \\ &\overset{*}{\Longrightarrow} w_0 Y(\mathcal{A}_1) \dots Y(\mathcal{A}_n) w_n \\ &= w \end{aligned}$$

[†] Dieser Beweisschritt ist nicht ganz präzise... die Ableitungen der v_i sind nicht unbedingt Teil der gesamten Ableitung $A \overset{n}{\Longrightarrow} w$. Man kann aber beweisen dass eine alternative Ableitung existieren muss, bei der dies der Fall ist.

[††] Der Induktionsanfang ist bei Ableitungsbäumen ein Spezialfall des Induktionsschritts (Bäume ohne Kinder bzw. Regeln, die nur Terminale ableiten) und wird daher nicht gesondert aufgeführt

[†††] Strengenommen müsste die Folgerung, die hier mit „..." abgekürzt ist noch per Induktion über Ableitungen gezeigt werden

□

Def. 3.5:

- Sei CFG die Menge der kontextfreien Grammatiken. Eine Grammatik $\mathcal{G} \in$ CFG heißt *eindeutig*, falls es für jedes Wort $\in L(\mathcal{G})$ genau einen Ableitungsbaum gibt.
- Eine kontextfreie Sprache heißt *eindeutig*, falls eine eindeutige Menge der kontextfreien Grammatiken (CFG) für sie existiert.

⊕

Bsp. 3.5:

$L = \{a^i b^j c^k \mid i = j \text{ oder } j = k\}$
$S \to AC \mid DB$
$A \to aAb \mid \varepsilon$ $\qquad$ $D \to aD \mid \varepsilon$
$C \to cC \mid \varepsilon$ $\qquad$ $B \to bBc \mid \varepsilon$ [für L gibt es keine eindeutige CFG]

Werte der Form $a^n b^n c^n$ haben zwei Ableitungen $\curvearrowright$ Grammatik ist nicht eindeutig.

Vorlesung: 2.12.16

3.2 Die Chomsky Normalform für kontextfreie Sprachen

Notwendig für das Pumping Lemma kontextfreier Sprachen und für einen effizienten Algorithmus für das Wortproblem.

Def. 3.6: Eine CFG heißt *separiert*, wenn jede Produktion entweder die Form

- $A \to A_1 \dots A_n \qquad A_i \in N, n \geq 0$, oder
- $A \to a \quad a \in \Sigma$

hat. ⊕

Lemma 3.4 (Sep)**:** Zu jeder CFG gibt es eine äquivalente separierte CFG.

Beweis: Sei $\mathcal{G} = (N, \Sigma, P, S)$ eine CFG. Konstruiere $\mathcal{G}' = (N', \Sigma, P', S)$ mit

- $N' = N \uplus \{Y_a \mid a \in \Sigma\}$ („$\uplus$" steht für „disjunkte Vereinigung")
- $P' = \{Y_a \to a \mid a \in \Sigma\} \cup \{A \to \beta[a \to Y_a] \mid A \to \beta \in P\}$. Die Schreibweise $\beta[a \to Y_a]$ bedeutet hier, dass alle $a \in \Sigma$, die in β vorkommen, durch Y_a ersetzt werden.

Offenbar gilt $L(\mathcal{G}) = L(\mathcal{G}')$ und $\mathcal{G}'$ ist separiert (ohne Beweis). Der Aufwand für SEP beträgt $O(|\mathcal{G}|)$, wobei die Größe einer Grammatik definiert ist durch

$$|G| = \sum_{A \in N} \sum_{A \to \alpha \in P} |A\alpha|$$

Das ist genau die Anzahl an Zeichen aus Σ, die man benötigt, um die Produktionen der Grammatik aufzuschreiben. □

Lemma 3.5 (BIN)**:** Zu jeder CFG gibt es eine äquivalente CFG, bei der für alle Produktionen $A \to \alpha$ gilt, dass $|\alpha| \leq 2$.

BEWEIS: Ersetze jede Produktion der Form

$$A \to X_1 X_2 \dots X_n$$

mit $X_i \in N \cup \Sigma$, $1 \leq i \leq n$, durch die Regeln

$$\begin{aligned} A &\to X_1 \langle X_2 \dots X_n \rangle \\ \langle X_2 \dots X_n \rangle &\to X_2 \langle X_3 \dots X_n \rangle \\ &\vdots \\ \langle X_{n-1} X_n \rangle &\to X_{n-1} X_n \end{aligned}$$

Dabei sind $\langle X_2 \dots X_n \rangle$, …, $\langle X_{n-1} \dots X_n \rangle$ neue Nichterminalsymbole.

Der Aufwand von BIN liegt in $O(|G|)$. □

Def. 3.7: Sie $\mathcal{G} = (N, \Sigma, P, S)$ eine CFG. Definiere die Menge Nullable$(\mathcal{G}) \subseteq N$ als

$$\text{Nullable}(\mathcal{G}) = \{A \in N \mid A \overset{*}{\Longrightarrow} \varepsilon\}.$$

Es handelt sich um die Menge an Nichtterminalen, aus denen das leere Wort abgeleitet werden kann. ⊕

Satz 3.6: Es gibt einen Algorithmus, der Nullable$(\mathcal{G})$ in $O(|\mathcal{G}|^2)$ ausrechnet.

BEWEIS: Definiere M_i als die Menge der Nichtterminale, aus denen sich ε mit einem Ableitungsbaum der Höhe $< i$ ableiten lässt:†

$$\begin{aligned} M_0 &= \emptyset \\ M_{i+1} &= M_i \cup \{A \mid A \to \alpha \in P \text{ und } \alpha \in M_i^*\} \end{aligned}$$

Es gilt für alle $i \in \mathbb{N}$, dass $M_i \subseteq N$. Da $|N|$ endlich ist, existiert ein $m \in \mathbb{N}$, so dass

$$M_m = M_{m+1} = \bigcup_{i \in \mathbb{N}} M_i$$

Das bedeutet, $\bigcup_{i \in \mathbb{N}} M_i$ lässt sich iterativ berechnen (s.u.).

Wir zeigen nun, dass Nullable$(\mathcal{G}) = \bigcup_{i \in \mathbb{N}} M_i$.

† Im Folgenden ist mit $*$ bei M_i^* und M^* der Kleene-Stern gemeint

„⊇" Es ist zu zeigen, dass $\forall i \in \mathbb{N} : M_i \subseteq \text{Nullable}(\mathcal{G})$

Per Induktion über i:

IA $i = 0$, $M_0 = \emptyset \subseteq \text{Nullable}(\mathcal{G})$

IS $i \Rightarrow i + 1$

Wenn $A \in M_{i+1}$, dann ist

- entweder $A \in M_i \subseteq \text{Nullable}(\mathcal{G})$ nach IV oder
- es existiert $A \to A_1 \dots A_n \in P$ mit $A_j \in M_i$ für $1 \leq j \leq n$. Per IV existieren Ableitungen

$$A_j \overset{*}{\Longrightarrow} \varepsilon$$

denen kann die obige Produktion vorangestellt werden, sodass auch eine Ableitung von A nach ε existert.

$$A \Longrightarrow A_1 \dots A_n \overset{*}{\Longrightarrow} \underbrace{\varepsilon \dots \varepsilon}_{n \text{ mal}} = \varepsilon$$

Also gilt $A \in \text{Nullable}(\mathcal{G})$

„⊆" Wenn $A \in \text{Nullable}(\mathcal{G})$, dann existiert $m \in \mathbb{N}$, so dass $A \overset{*}{\Longrightarrow} \varepsilon$ mit einem Ableitungsbaum der Höhe m. Wir zeigen per Induktion über m, dass $A \in M_{m+1} \subseteq \bigcup_{i \in \mathbb{N}} M_i$.

IA $m = 0$. Die Ableitung ist $A \Longrightarrow \varepsilon$, sodass $A \in M_1$.

IS $m > 0$. Die Wurzel des Ableitungsbaum muss mit $A \to A_1 \dots A_n$ $(n > 0)$ markiert sein und die Kinder sind jeweils Ableitungsbäume für ε in $Abl(\mathcal{G}, A_i)$ der Höhe $m_i \leq m - 1$, wobei $1 \leq i \leq n$.

Es gilt nun per IV, dass $A_i \in M_{m_i} \subseteq M_{m-1}$. Somit ist $A_i \in M_{m-1}$ und damit, per Definition, $A \in M_m$.

□

Aus dem Beweis ergibt sich der folgende Algorithmus zur Berechnung von $\bigcup_{i \in \mathbb{N}} M_i$.

```
M = {}
done = false
while (not done):
    done = true
    foreach A → a ∈ P:
        if (A ∉ M ∧ a ∈ M*):
            M = M ∪ A
            done = false
return M
```

Korollar 3.7: Es gibt einen Algorithmus, der zu einer CFG $\mathcal{G}$ berechnet ob $\varepsilon \in L(\mathcal{G})$. ⊕

BEWEIS: Prüfe ob $S \in \text{Nullable}(\mathcal{G})$ □

Lemma 3.8 (DEL)**:** Zu jeder CFG $\mathcal{G} = (N, \Sigma, P, S)$ gibt es eine äquivalente CFG $\mathcal{G}' = (N', \Sigma, P', S')$, bei der die einzige ε-Regel $S' \to \varepsilon$ ist (falls $\varepsilon \in L(\mathcal{G})$) und bei der S' auf keiner rechten Seite einer Produktion vorkommt.

BEWEIS:

1. Erweitere $\mathcal{G}$ um ein neues Startsymbol S' mit $S' \to S$ als neue Produktion. Dieser Schritt stellt sicher, dass S' auf keiner rechten Regelseite vorkommt.
2. Wende erst SEP, dann BIN an. Nun hat jede rechte Regelseite die Form $a \in \Sigma$ oder ε oder A oder BC.
3. Für alle Regeln $A \to BC \in P$:
   - Falls $B \in \text{Nullable}(\mathcal{G})$ füge $A \to C$ hinzu.
   - Falls $C \in \text{Nullable}(\mathcal{G})$ füge $A \to B$ hinzu.
4. Falls $S \in \text{Nullable}(\mathcal{G})$ füge $S' \to \varepsilon$ hinzu
5. Entferne alle Produktionen $A \to \varepsilon$ für $A \neq S'$.

□

Def. 3.8: Sei $\mathcal{G} = (N, \Sigma, P, S)$ eine Grammatik. Eine *Kettenregel* von $\mathcal{G}$ ist eine Produktion in P der Form $A \to B$, wobei $A, B \in N$. ⊕

Vorlesung: 7.12.16

Lemma 3.9 (UNIT)**:** Zu jeder CFG $\mathcal{G} = (N, \Sigma, P, S)$ gibt es eine äquivalente CFG $\mathcal{G}' = (N, \Sigma, P', S)$ ohne Kettenregeln.

BEWEIS: Setze zu Anfang $P' = P$ und eliminiere alle Kettenregeln mit folgendem Algorithmus:

1. Betrachte den gerichteten Graphen G mit Knoten N und Kanten $\{(A, B) \mid A, B \in N \text{ und } A \to B \in P'\}$.
2. Suche, z.B. mittels Tiefensuche, einen Zyklus in G . Wenn kein Zyklus gefunden wurde, weiter mit Schritt 4.
3. Wurde der Zyklus $A_1 \to A_2 \to \ldots \to A_k \to A_1$ gefunden, dann ersetze in P' alle Vorkommen von A_j mit $j > 1$ durch A_1 (auf linker *und* rechter Regelseite). Entferne dann alle Regeln der Form $A \to A$. Fahre fort mit Schritt 1.
4. Der Graph G ist ein gerichteter, azyklischer Graph (DAG, *Directed Acyclic Graph*). Sortiere die Knoten von G topologisch als $A_1, \ldots, A_n$, so dass A_n auf keiner linken Seite einer Kettenregel vorkommt.

5. **for** $j = m \dots 1$

   for each $A_j \to A_k \in P'$

   (wegen topologischer Sortierung gilt $k > j$)

   entferne $A_j \to A_k$ aus P'

   füge $A_j \to BC$ zu P' hinzu, falls $A_k \to BC \in P', \quad B, C \in N.$

   füge $A_j \to \mathtt{a}$ zu P' hinzu, falls $A_k \to \mathtt{a} \in P', \quad \mathtt{a} \in \Sigma.$

   Die äußere Schleife hat die Invariante, dass am Ende jedes Durchlaufs für $m \geq i \geq j$ keine Kettenregel $A_i \to \dots$ in P' existiert. Beim Verlassen der Schleife ist $j = 1$ und es existieren überhaupt keine Kettenregeln mehr.

□

Def. 3.9: Eine CFG $\mathcal{G} = (N, \Sigma, P, S)$ ist in Chomsky Normalform (CNF), falls jede Produktion die Form $A \to a$, $A \to BC$, oder $S \to \varepsilon$ hat, wobei $A, B, C \in N$, $a \in \Sigma$. Falls $S \to \varepsilon \in P$, dann darf S auf keiner rechten Seite einer Produktion vorkommen. ⊕

Beweis: Wende der Reihe nach Sep, Bin, Del und Unit an.† □

Beobachtung: Ist $\mathcal{G} = (N, \Sigma, P, S)$ in CNF, dann lassen sich für die Ableitungsbäume $\mathcal{A} \in \mathrm{Abl}(\mathcal{G}, S)$ folgende Eigenschaften feststellen:

1. $\mathcal{A}$ ist ein Binärbaum.
2. Falls $Y(\mathcal{A}) = w$ dann ist die Anzahl der Blätter des Baumes $|w|$.

Bemerkung: Sei $|\mathcal{G}| = \sum_{A \to \alpha \in P} (|\alpha| + 1)$ die Größe einer CFG.
Die Transformation nach CNF benötigt Zeit $O(|\mathcal{G}|^2)$. Die Größe der CNF-Grammatik ist $O(|\mathcal{G}|^2)$.

3.3 Das Pumping Lemma für kontextfreie Sprachen

Satz 3.10 (Pumping lemma für CFL, uvwxy Lemma)**:** Sei $L \in CFL$. Dann $\exists n > 0$, so dass $\forall z \in L$ mit $|z| \geq n$ $\exists u, v, w, x, y$ so dass $z = uvwxy$ mit

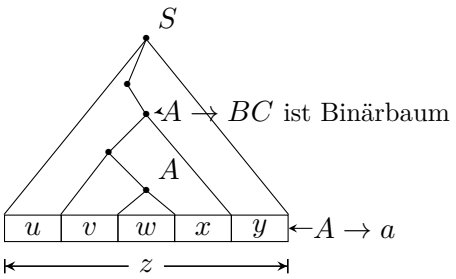
- $|vwx| \leq n$
- $|vx| \geq 1$
- $\forall i \in \mathbb{N} : uv^i wx^i y \in L$

†Es genügt sogar nur Del und dann Unit anzuwenden, da Del schon Sep und Bin ausführt.

BEWEIS: Sei $\mathcal{G} = (N, \Sigma, P, S)$ in CNF mit $L(\mathcal{G}) = L$. Wähle die Pumping-Konstante $n = 2^{|N|}$.

Betrachte den Ableitungsbaum $\mathcal{A} \in \text{Abl}(\mathcal{G}, S)$ von z mit $|z| \geq n$. [†] $\mathcal{A}$ ist ein Binärbaum mit $|z| \geq n = 2^{|N|}$ Blättern. In einem solchen Binärbaum existiert ein Pfad ζ der Länge $\geq |N|$. Auf ζ liegen $\geq |N| + 1$ Nichtterminale (NT), es muss also mindestens ein Nichtterminal A mehrmals vorkommen.

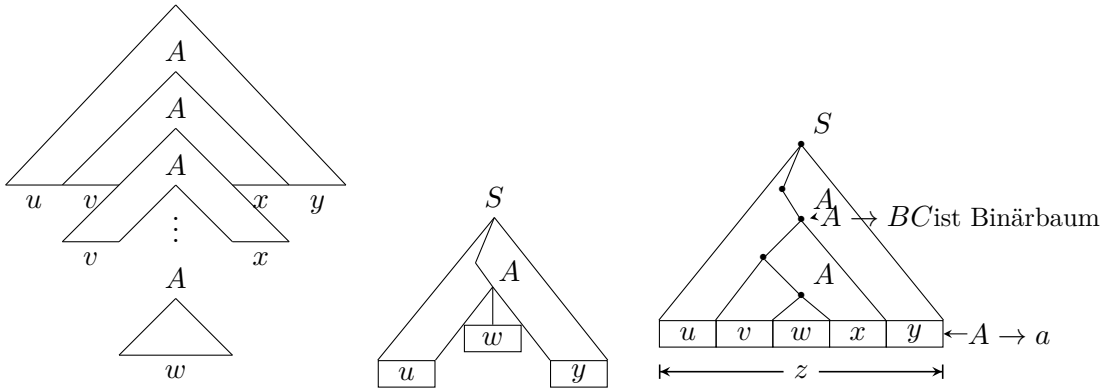
Folge ζ vom Blatt Richtung Wurzel und bis sich das ein A das erste Mal wiederholt. Das geschieht nach $\leq |N|$ Schritten. Nun teile $z = uvwxy$ wie hier skizziert:



Nun gilt

- $|vx| \geq 1$, da ζ entweder durch B oder durch C läuft. Nehme also an, ζ verläuft durch B. Somit muss $C \overset{*}{\Longrightarrow} x$. In CNF ist $C \overset{*}{\Longrightarrow} \varepsilon$ nicht möglich und somit ist $|x| \geq 1$. Der Fall, das ζ durch C verläuft, ist analog.
- $|vwx| \leq 2^{|N|} = n$ TODO

[†] Intuition: Da $\mathcal{G}$ in CNF, ist Ableitungsbaum $\mathcal{A}$ ein Binärbaum. Mit $|N|$ verschiedenen Nichtterminalsymbolen kann man also maximal Worte der Länge $2^{|N|}$ ableiten, wenn man keine Ableitung doppelt nutzen möchte. Leitet man ein längeres Wort ab, so muss man mind. eine Ableitung doppelt nutzen. Dann kann man sie aber auch gleich i-fach nutzen (v^i und x^i), und ist immernoch in der Sprache.



(a) $uv^iwx^iy \in L$ **(b)** $\curvearrowright uwy \in L$ **(c)** $\exists$ Pfad mit Länge $\geq k$

Abb. 11: Schema zu Satz 3.10

□

Lemma 3.11: $\mathcal{L}_2 \subsetneq \mathcal{L}_1$

Beweis: Sei $L = \{L = \{a^n b^n c^n \mid n \geq 1\}$.
L ist nicht kontextfrei. Verwende PL. Angenommen L sei kontextfrei. Sei dann n die Konstante aus dem PL.

Wähle $z = a^n b^n c^n$ und somit $|z| = 3n \geq n$. Nach PL ist $z = uvwxy$ mit $|vx| \geq 1$ und $|vwx| \leq n$.

Durch $|vwx| \leq n$ ergeben sich folgende Möglichkeiten:

- $vwx = a^j$: Für $i = 0$ ist $vwx = w$ mit $|w| < j$. Anzahl der a stimmt nicht mehr mit n überein.
- $vwx = a^k b^j$:
  - Falls $v = a^{k'}$, $x = b^{j'}$: Für $i = 0$ stimmt die Anzahl der a nicht mehr mit n überein.
  - Falls v ein Gemisch aus a und b enthält. Für $i \geq 2$ würde eine Folge a^n durch eine b-Folge unterbrochen.
  - Falls x ein Gemisch aus a und b enthält. Für $i \geq 2$ würde eine Folge a^n durch eine b-Folge unterbrochen.
- $vwx = b^j$: analog Fall a^j
- $vwx = b^k c^j$: analog Fall $a^k b^j$
- $vwx = c^j$: analog Fall a^j

In jeder der Möglichkeiten lässt sich durch Pumpen ein Wort $w \notin L$ finden. Daher kann L nicht kontextfrei sein. □

Bsp. 3.6: Die Sprache

$$L = \{ww \mid w \in \{a, b\}^*\}$$

ist kontextsensitiv aber nicht kontextfrei.

Sei n die Konstante aus dem PL

Betrachte $z = a^n b^n a^n b^n \in L$ mit $|z| = 4n \geq n$. Nach PL ist $z = uvwxy$ mit $|vx| \geq 1$ und $|vwx| \leq n$.

Es ergeben sich folgende Möglichkeiten:

- $vwx = a^j$, $j \leq n$. Für $i = 0$ ist $vwx = w$ mit $|w| < j$. Anzahl der a stimmt nicht mehr mit n überein.
- $vwx = a^k b^j$, $k + j \leq n$.
  - Falls $v = a^{k'}$, $x = b^{j'}$: Für $i = 0$ stimmt die Anzahl der a nicht mehr mit n überein.
  - Falls v ein Gemisch aus a und b enthält. Für $i > 2$ würde eine Folge a^n durch eine b-Folge unterbrochen.
  - Falls x ein Gemisch aus a und b enthält. Für $i > 2$ würde eine Folge a^n durch eine b-Folge unterbrochen.
- $vwx = b^j$ analog Fall a^j
- $vwx = b^k a^j$ analog Fall $a^k b^j$

Entwurf für Vorlesung: 9.12.16

3.4 Entscheidungsprobleme für kontextfreie Sprachen

Satz 3.12: Das Wortproblem „$w \in L$?“ ist für Menge der kontextfreien Sprachen (CFL) entscheidbar. Falls $|w| = n$, benötigt der Algorithmus $O(n^3)$ Schritte und $O(n^2)$ Platz.

Beweis: Algorithmus Cocke, Younger, Kasami (CYK). □

Bsp. 3.7: $L = \{a^n b^n c^m \mid n, m \geq 1\}$ mit Grammatik (CFG)

$$\begin{aligned} S &\to XY \\ X &\to ab \mid aXb \\ Y &\to c \mid cY \end{aligned}$$

In CNF: $S \to XY$
$X \to AB \mid AZ$
$Z \to XB$
$Y \to CY|c$
$A \to a, B \to b, C \to c$

BEWEIS (CYK-Algorithmus): Sei $\mathcal{G}$ ein CFG in CNF und sei $w = a_1 \dots a_n \in \Sigma^*, |w| = n$. Der CYK-Algorithmus bereichnet bei Eingabe von $\mathcal{G}$ und w, ob $w \in L(\mathcal{G})$.

Idee dabei: Berechne eine $(n \times n)$ Matrix M mit Einträgen in $\mathcal{P}(N)$ mit folgender Spezifikation:

$M_{ij} = \{A \mid A \overset{*}{\Rightarrow} a_i \dots a_j\}$ $\qquad 1 \le i \le j \le n$ nur diese $M_{ij} \neq \varnothing$

falls $i = j$:

$M_{ii} = \{A \mid A \overset{*}{\Rightarrow} a_i\}$
$= \{A \mid A \to a_i\}$

falls $1 \le i < j \le n$:

$M_{ij} = \{A \mid A \overset{*}{\Rightarrow} a_i \dots a_j\}$
$= \{A \mid A \Rightarrow BC \overset{*}{\Rightarrow} a_i \dots a_j\}$
$= \{A \mid A \to BC \wedge BC \overset{*}{\Rightarrow} a_i \dots a_j\}$
$= \{A \mid A \to BC \wedge \exists k : i \le k < j \quad B \overset{*}{\Rightarrow} a_i \dots a_k \wedge C \overset{*}{\Rightarrow} a_{k+1} \dots a_j\}$

Damit: $w \in L$ genau dann, wenn $S \Rightarrow^* w$ genau dann, wenn $S \in M_{1,n}$.

Beispiel der Matrix für $w = aaabbbcc$. Zellen, die mit „•" markiert sind, sind leer. Die Reihenfolge, in der die nicht-diagonalen, nicht leeren Zellen, eingetragen wurden ist mit kleinen Zahlen. Beispielsweise wurde $M_{35} = \{Z\}$ mit Zahl „3" *nach* $M_{78} = \{Y\}$ mit Zahl „1" eingetragen. Da $M_{1n} = \{S\}$ gilt $w \in L$.

$w =$	a	a	a	b	b	b	c	c
$M =$	A	•	•	•	•	X 6	S 7	S 8
		A	•	•	X 4	Z 5	•	•
			A	X 2	Z 3	•	•	•
				B	•	•	•	•
					B	•	•	•
						B	•	•
							C, Y	Y 1
								C, Y

□

CYK($\mathcal{G}, a_1, \ldots, a_n$)
M $n \times n$ Matrix mit $M_{ij} = \varnothing$

```
for i=1 .. n do   // O(|G|) · O(n)
  M_ii = {A | A → a_i}
for i=n−1 .. 1 do
  for j=i+1 .. n do
    for k=i .. j−1 do   // O(|G|) · O(n^3)
      M_ij = M_ij ∪ {A | A → BC, B ∈ M_ik, C ∈ M_{k+1,j}}
return S ∈ M_1n
```

Satz 3.13: Das Leerheitsproblem ist für CFL entscheidbar.

Beweis: Sei $\mathcal{G} = (N, \Sigma, P, S)$ eine CFG für L. Gefragt ist, ob $L(\mathcal{G}) = \{w \in \Sigma^* \mid S \overset{*}{\Rightarrow} w\} \overset{?}{=} \varnothing$. Betrachte dazu die Menge M der nützlichen Nichtterminalsymbole, aus denen ein Terminalwort herleitbar ist.

$$
\begin{aligned}
M &= \{A \mid A \overset{*}{\Rightarrow} w, w \in \Sigma^*\} \\
M_0 &= \{A \mid A \to w \in P, w \in \Sigma^*\} \\
M_{i+1} &= M_i \cup \{A \mid A \to \alpha \in P, \alpha \in (\Sigma \cup M_i)^*\} \\
\exists n : M_n = M_{n+1} &\overset{!}{=} M \\
L = \varnothing &\Leftrightarrow S \notin M
\end{aligned}
$$

□

Offenbar ist $M_0 \subseteq M$ eine Approximation von M, $M \supseteq M_{i+1} \supseteq M_i$ verbessert M_i und das jeweils nächste M_i ist in Linearzeit berechenbar. Da $M \subseteq N$ endlich ist, muss es ein n geben, sodass $M_n = M_{n+1}$. Für dieses n kann man zeigen, dass $M = M_n$.

Bem.: M ist die Menge der nützlichen Nichterminale. Nicht nützliche Nichtterminale können entfernt werden, ohne dass $L(\mathcal{G})$ sich ändert.

Satz 3.14: Das Endlichkeitsproblem für CFL ist entscheidbar.

Beweis: Mit PL analog zum Endlichkeitsproblem für reguläre Sprachen (Prüfe, ob $\exists w \in L$ mit $n < |w| \leq 2n$). □

3.5 Abschlusseingenschaften für kontextfreie Sprachen

Satz 3.15: Die Menge CFL ist abgeschlossen unter $\cup, \cdot, ^*$, jedoch *nicht* unter $\cap, \overline{\ }$.

Beweis:

$$\begin{aligned} \mathcal{G}_i &:= (N_i, \Sigma, P_i, S_i) \quad i = 1, 2 \\ \text{„}\cup\text{“} : N &= N_1 \dot\cup N_2 \dot\cup \{S\} \\ P &= \{S \to S_1, S \to S_2\} \cup P_1 \cup P_2 \\ \text{„}\cdot\text{“} :: N &= N_1 \dot\cup N_2 \dot\cup \{S\} \\ P &= \{S \to S_1 S_2\} \cup P_1 \cup P_2 \\ \text{„}^*\text{“} : N &= N_1 \dot\cup \{S\} \\ P &= \{S \to \varepsilon, S \to S_1 S\} \dot\cup P_1 \end{aligned}$$

- für $n \geq 1$

$$\underbrace{\{a^n b^n c^n\}}_{\notin \text{CFL}} = \underbrace{\{a^n b^n c^m \mid n, m \geq 1\}}_{\in \text{CFL}} \cap \underbrace{\{a^m b^n c^n \mid m, n \geq 1\}}_{\in \text{CFL}}$$

  Also CFL nicht abgeschlossen unter $\cap$.

- Angenommen CFL wäre abgeschlossen unter $\overline{}$
  Falls $L_1, L_2 \in \text{CFL}$, dann ist $\overline{L}_1, \overline{L}_2 \in \text{CFL}$ nach Annahme.
  $\curvearrowright \overline{L}_1 \cup \overline{L}_2 \in \text{CFL}$ wegen Teil “$\cup$“.
  $\curvearrowright \overline{\overline{L}_1 \cup \overline{L}_2} = L_1 \cap L_2 \in \text{CFL}$ $\lightning$ zu Teil “$\cap$“. □

Vorlesung: 14.12.16

Satz 3.16: Die Menge CFL ist abgeschlossen unter „$\cap$ mit regulären Sprachen REG”. Das heißt, für alle $L \in \text{CFL}$ und $R \in \text{REG}$ gilt $L \cap R \in \text{CFL}$.

Beweis: Sei $\mathcal{G} = (N, \Sigma, P, S)$ kontextfreie Grammatik in CNF mit $L(\mathcal{G}) = L$.

Sei $M = (Q, \Sigma, q, \delta, q_0, F)$ NEA mit $L(M) = R$.

Definiere $\mathcal{G}' = (N', \Sigma, P', S)$ durch

$$\begin{aligned} N' &= Q \times N \times Q \cup \{S\} \\ P' = &\{(p, A, q) \to a \mid A \to a \in P \text{ und } \delta(p, a) \ni q\} \\ &\cup \{(p, A, q) \to (p, B, q')(q', C, q) \mid A \to BC \in P \text{ und } p, q, q' \in Q\} \\ &\cup \{S \to (q_0, S, q) \mid q \in F\} \end{aligned}$$

Durch die $S \to \ldots$ Regeln erzeugt $\mathcal{G}'$ offensichtlich genau die Worte, die durch die Nichtterminale (q_0, S, q) für $q \in F$ erzeugt werden. Es bleibt zu zeigen, dass ein Nichtterminal genau die Worte aus L erzeugt die gleichzeitig einen (akzeptierenden) Lauf $q_0 \ldots q$ von M erlauben. Wir zeigen eine Verallgemeinerung: für alle $p, q \in Q$ und $A \in N$ gilt

$$\begin{aligned} &(p, A, q) \overset{*}{\Longrightarrow}_{\mathcal{G}'} w \\ \text{gdw } &A \overset{*}{\Longrightarrow}_{\mathcal{G}} w \text{ und es existiert ein Lauf } p \ldots q \text{ von } M \text{ auf } w \end{aligned}$$

- Richtung „links nach rechts": (siehe Übung)
- Richtung „rechts nach links":

  Per Induktion über den Ableitungsbaum $\mathcal{A} = \pi(\mathcal{A}_1, \ldots, \mathcal{A}_n) \in \mathrm{Abl}(\mathcal{G}, A)$ mit $Y(\mathcal{A}) = w$.

  IV $\forall 1 \leq i \leq n$: wenn $w_i = Y(\mathcal{A}_i)$ mit $\mathcal{A}_i \in \mathrm{Abl}(\mathcal{G}, A_i)$ und es existiert ein Lauf $p_i \ldots q_i$ von M auf w_i, dann $(p_i, A_i, q_i) \overset{*}{\Longrightarrow}_{\mathcal{G}'} w_i$.

  IS

  - $\pi = A \to a$, $a \in \Sigma$. Es gilt $w = a$ und pq ist Lauf auf a. Folglich ist $\delta(p, a) \ni q$. Damit ist $(p, A, q) \to a \in P'$, woraus direkt folgt, dass $(p, A, q) \Longrightarrow_{\mathcal{G}'} a$.
  - $\pi = A \to BC, \mathcal{A} = \pi(\mathcal{A}_1, \mathcal{A}_2), Y(\mathcal{A}_1) = w_1, Y(\mathcal{A}_2) = w_2, \mathcal{A}_1 \in \mathrm{Abl}(\mathcal{G}, B)$ und $\mathcal{A}_2 \in \mathrm{Abl}(\mathcal{G}, C)$.

    Es gilt $w = w_1 w_2$ und $\zeta = p \ldots q$ ist Lauf auf $w_1 w_2$. Es existiert also q' so dass $\zeta = p \ldots q' \ldots q$ und $p \ldots q'$ ist Lauf auf w_1 und $q' \ldots q$ ist Lauf auf w_2.

    Es folgt per IV, dass

    $$(p, B, q') \overset{*}{\Longrightarrow}_{\mathcal{G}'} Y(\mathcal{A}_1) = w_1 \text{ und } (q', C, q) \overset{*}{\Longrightarrow}_{\mathcal{G}'} Y(\mathcal{A}_2) = w_2.$$

    Nun ist $(p, A, q) \to (p, B, q')(q', C, q) \in P'$ per Konstruktion, also ist folgende Ableitung möglich:

    $$(p, A, q) \Longrightarrow_{\mathcal{G}'} (p, B, q')(q', C, q) \overset{*}{\Longrightarrow}_{\mathcal{G}'} w_1(q', C, q) \overset{*}{\Longrightarrow}_{\mathcal{G}'} w_1 w_2 = w$$

□

Satz 3.17: Sei $L \in \mathrm{CFL}$ und $R \in \mathrm{REG}$. Es ist entscheidbar, ob $L \subseteq R$.

Beweis: Es gilt $L \subseteq R$ gdw $L \cap \overline{R} = \emptyset$.

Da $R \in \mathrm{REG}$ ist durch die Abschlusseigenschaften regulärer Sprachen auch $\overline{R} \in \mathrm{REG}$. Nach Satz 3.15 ist $L \cap \overline{R} \in \mathrm{CFL}$. Nach Satz 3.13 ist $L \cap \overline{R} = \emptyset$ daher entscheidbar □

4 Kellerautomaten pushdown automaton (PDA)

Kellerautomat ≈ Endlicher Automat + Kellerspeicher von unbeschränkter Größe (Stack, push down)

Neu:

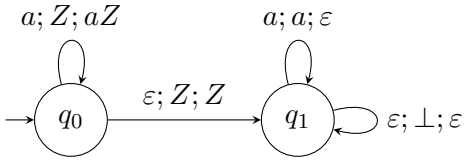
- bei jedem Schritt darf der PDA das oberste Kellersymbol inspizieren und durch beliebiges Kellerwort ersetzen (den neuen Kellerpräfix).
- der PDA darf auf dem Keller rechnen, ohne in der Eingabe weiter zu lesen. (ε-Transition oder Spontantransition).

Bsp. 4.1:

$$\Sigma = \{0,1\} \qquad \text{Eingabealphabet}$$
$$\Gamma = \{0,1,\bot\} \qquad \text{Kelleralphabet}$$
$$Q = \{q_0, q_1\} \qquad \bot \mathrel{\hat{=}} \text{Kellerbodensymbol}$$
$$\delta(q_0, a, Z) = \{(q_0, aZ)\} \qquad a \in \{0,1\}, Z \in \Gamma \tag{2}$$
$$\delta(q_0, \varepsilon, Z) = \{(q_1, Z)\} \tag{3}$$
$$\delta(q_1, a, a) = \{(q_1, \varepsilon)\} \tag{4}$$
$$\delta(q_1, \varepsilon, \bot) = \{(q_1, \varepsilon)\} \tag{5}$$

Die Übergangsfunktion δ bildet den aktuellem Zustand, das aktuelle Eingabesymbol und das aktuell oberste Kellersymbol auf Paare von Folgezustand und neuem Kellerpräfix ab. In diesem Beispiel ist die zurückgegebene Paarmenge in allen Fällen einelementig. Durch die ε Transitionen ist der PDA aber trotzdem nichtdeterministisch.

Bei der graphischen Darstellung werden die Transitionen mit Tripeln $a; Z; \gamma$ beschriftet, wobei $a \in \Sigma$ das Eingabesymbol, $Z \in \Gamma$ das oberste Kellersymbol und $\gamma \in \Gamma^*$ der neue Kellerpräfix ist:



wobei hier $a \in \Sigma$ und $Z \in \Gamma$.

Der Automat beginnt im Startzustand q_o mit einem Kellerspeicher, der nur das Kellerbodensymbol Z_0 enthält. Er akzeptiert ein Wort, wenn er alle Eingabesymbole gelesen und den Keller komplett leeren konnte. Anders als bei EAs gibt es *keine* finalen Zustände.

Die erkannte Sprache ist hier $L = \{ww^R \mid w \in \{0,1\}^*\}$.

Vorlesung: 16.12.16

Def. 4.1: Ein nichtdeterministischer Kellerautomat (NPDA) ist Tupel $(Q, \Sigma, \Gamma, q_0, Z_0, \delta)$

- Q endliche Zustandsmenge
- Σ endliches Eingabealphabet
- Γ endliches Kelleralphabet
- $q_0 \in Q$ Startzustand
- $Z_0 \in \Gamma$ Kellerbodensymbol
- $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to \mathcal{P}(Q \times \Gamma^*)$ ⊕

Im weiteren sei $M = (\dots)$ ein NPDA.

Def. 4.2: Die Menge der *Konfigurationen* von M ist $\mathrm{Konf}(M) = Q \times \Sigma^* \times \Gamma^*$
Die *Schrittrelation* von M

$$\vdash \subseteq \mathrm{Konf}(M) \times \mathrm{Konf}(M)$$

ist definiert durch

$$\begin{aligned} (q, aw, Z\gamma) &\vdash (q', w, \beta\gamma) && \text{falls } \delta(q, a, z) \ni (q', \beta) \\ (q, w, Z\gamma) &\vdash (q', w, \beta\gamma) && \text{falls } \delta(q, \varepsilon, z) \ni (q', \beta) \end{aligned}$$

Wir schreiben $(q, w, \gamma) \vdash^n (q', w', \gamma')$ wenn M in $n \in \mathbb{N}$ Schritten von Konfiguration (q, w, γ) nach Konfiguration (q', w', γ') gelangt.

Wir schreiben $\vdash^*$ für die reflexive, transitive Hülle von $\vdash$. Falls $(q, w, \gamma) \vdash^* (q', w', \gamma')$ existiert also $n \in \mathbb{N}$, so dass $(q, w, \gamma) \vdash^n (q', w', \gamma')$.

Die von M *erkannte Sprache* ist

$$L(M) = \{w \in \Sigma^* \mid (q_0, w, Z_0) \vdash^* (q', \varepsilon, \varepsilon)\}$$ ⊕

Bsp.: Die folgenden Schritte von M aus Beispiel 4.1 zeigen, dass $w = 0110 \in L(M)$:

$$\begin{aligned} &(q_0, 0110, \bot) \\ \vdash\ &(q_0, 110, 0\bot) && (\text{„pushen” des Eingabesymbols 0}) \\ \vdash\ &(q_0, 10, 10\bot) && (\text{„pushen” des Eingabesymbols 1}) \\ \vdash\ &(q_1, 10, 10\bot) && (\varepsilon\text{-Übergang von } q_0 \text{ nach } q_1) \\ \vdash\ &(q_1, 0, 0\bot) && (\text{„poppen” des Eingabesymbols 1}) \\ \vdash\ &(q_1, \varepsilon, \bot) && (\text{„poppen” des Eingabesymbols 0}) \\ \vdash\ &(q_1, \varepsilon, \varepsilon) && (\varepsilon\text{-Übergang zum Entfernen von } \bot) \end{aligned}$$

Satz 4.1:

$$L \in \text{CFL gdw } L = L(M) \text{ für einen NPDA } M$$

Vorlesung: 21.12.16

Beweis:

- CFG zu NPDA: Sei $\mathcal{G} = (N, \Sigma, P, S)$ Grammatik für L in CNF.

  Definiere NPDA M durch

  - $Q = \{q_0\}$
  - $\Gamma = \Sigma \uplus N$
  - $Z_0 = S$
  - $\delta(q_0, a, a) = \{(q_0, \varepsilon)\}$ für $a \in \Sigma$
  - $\delta(q_0, \varepsilon, A) = \{(q_0, \alpha)\}$ für $A \to \alpha \in P$

  Wir zeigen nun, dass $L(M) = L(\mathcal{G})$.

  Zum Führen des Beweises benötigen wir folgende Beobachtung, die für alle NPDAs $M = (\ldots)$, $q, q' \in Q$, $w \in \Sigma^*$, $Z \in \Gamma$ gilt:

  $$\text{Wenn } (q, w, Z) \vdash^* (q', \varepsilon, \varepsilon) \text{ dann } \forall v \in \Sigma^*, \gamma \in \Gamma^* : (q, wv, Z\gamma) \vdash^* (q', v, \gamma)$$

  Der Beweis ist per Induktion über die Länge der Ableitung (nicht gezeigt).

  - Wir zeigen
    $$\text{wenn } S \overset{*}{\Longrightarrow} w \text{ dann } (q_0, w, S) \vdash^* (q_0, \varepsilon, \varepsilon)$$
    Wir beweisen dazu die stärkere Aussage
    $$\forall A \in N : \text{ wenn } A \overset{*}{\Longrightarrow} w \text{ dann } (q_0, w, A) \vdash^* (q_0, \varepsilon, \varepsilon)$$
    per Induktion über den Ableitungsbaum $\mathcal{A} = \pi(\mathcal{A}_1, \ldots, \mathcal{A}_n) \in \text{Abl}(\mathcal{G}, A)$ mit $Y(\mathcal{A}) = w$.

    IV Für alle $n' \leq n$ und $1 \leq i \leq n'$, $A_i \in N$, $\mathcal{A}_i \in \text{Abl}(\mathcal{G}, A_i)$ gilt:
    $$(q_0, Y(\mathcal{A}_i), A_i) \vdash^* (q_0, \varepsilon, \varepsilon) \qquad (\star)$$

    IS Unterscheide die Form der Produktionen der CNF-Grammatik $\mathcal{G}$:

    * $\pi = S \to \varepsilon$. Per Konstruktion gilt $(q_0, \varepsilon) \in \delta(q_0, \varepsilon, S)$ und somit $(q_0, \varepsilon, S) \vdash (q_0, \varepsilon, \varepsilon)$.

* $\pi = A \to a$, $a \in \Sigma^*$. Per Konstruktion gilt $(q_0, a) \in \delta(q_0, \varepsilon, A)$ und $(q_0, \varepsilon) \in \delta(q_0, \varepsilon)$.

  Somit gilt $(q_0, a, A) \vdash (q_0, a, a) \vdash (q_0, \varepsilon, \varepsilon)$.

* $\pi = A \to BC$, $B, C \in N$, $\mathcal{A} = \pi(\mathcal{A}_1, \mathcal{A}_2)$, $\mathcal{A}_1 \in \mathrm{Abl}(\mathcal{G}, B)$, $\mathcal{A}_1 \in \mathrm{Abl}(\mathcal{G}, B)$, $w = Y(\mathcal{A}_1)Y(\mathcal{A}_2)$.

  Per Konstruktion gilt $(q_0, BC) \in \delta(q_0, \varepsilon, A)$.

  Ferner gilt per IV, dass $(q_0, Y(\mathcal{A}_1), B) \vdash^* (q_0, \varepsilon, \varepsilon)$ und $(q_0, Y(\mathcal{A}_2), C) \vdash^* (q_0, \varepsilon, \varepsilon)$.

  Es folgt mit Beobachtung $(\star)$, dass $(q_0, w, A) = (q_0, Y(\mathcal{A}_1)Y(\mathcal{A}_2), A) \vdash (q_0, Y(\mathcal{A}_1)Y(\mathcal{A}_2), BC) \vdash^* (q_0, Y(\mathcal{A}_2), C) \vdash^* (q_0, \varepsilon, \varepsilon)$.

– Wir zeigen

$$\text{wenn } (q_0, w, S) \vdash^* (q_0, \varepsilon, \varepsilon) \text{ dann } S \overset{*}{\Longrightarrow} w$$

Wir beweisen dazu die stärkere Aussage

$$\forall n \in \mathbb{N} : \forall w \in \Sigma^*, \alpha \in \Gamma^* : \text{ wenn } (q_0, w, \alpha) \vdash^n (q_0, \varepsilon, \varepsilon) \text{ dann } \alpha \overset{*}{\Longrightarrow} w$$

per Induktion über die Anzahl der Berechnungsschritte n.

IA $n = 0$. $w = \varepsilon$, $\alpha = \varepsilon$: es gilt $\varepsilon \Longrightarrow \varepsilon$.

IV Für alle $n' < n$ und $w \in \Sigma^*, \alpha \in \Gamma^*$ gilt

$$\text{wenn } (q_0, w, \alpha) \vdash^{n'} (q_0, \varepsilon, \varepsilon) \text{ dann } \alpha \overset{*}{\Longrightarrow} w$$

IS $n > 0$, $\alpha = Z\alpha'$,

$(q_0, w, Z\alpha') \vdash (q_0, w', \beta\alpha') \vdash^{n-1} (q_0, \varepsilon, \varepsilon)$, $\delta(q_0, x, Z) \ni (q_0, \beta)$

Es gibt zwei Fälle für Z:

* $Z = a$, $a \in \Sigma$.

  Es folgt $\beta = \varepsilon$ und $w = aw'$.

  Per IV gilt $\alpha' \overset{*}{\Longrightarrow} w'$ und somit auch $\alpha = a\alpha' \overset{*}{\Longrightarrow} aw' = w$.

* $Z = A$, $A \to \beta \in P$.

  Es folgt $w = w'$.

  Per IV gilt $\beta\alpha' \overset{*}{\Longrightarrow} w'$ und somit auch $A\alpha \Longrightarrow \beta\alpha' \overset{*}{\Longrightarrow} w' = w$.

- NPDA zu CFG:

  Zunächst zeigen wir, dass es genügt NPDAs zu betrachten, die bei jeder Transition Wörter der maximalen Länge 2 auf den Keller schreiben:

Lemma 4.2: Zu jedem NPDA gibt es einen äquivalenten NPDA, so dass falls $\delta(q, x, Z) \ni (q', \gamma) \quad x \in \Sigma \cup \{\varepsilon\}$ dann ist $|\gamma| \leq 2$

BEWEIS: Sei $(q', \gamma) \in \delta(q, x, Z)$ mit $\gamma = Z_n \dots Z_1$ für $n > 2$:

- neue Zustände $q_2 \dots q_{n-1}$
- Ersetze (q', γ) durch $(q_2, Z_2 Z_1)$
- Definiere $\delta(q_i, \varepsilon, Z_i) = \{(q_{i+1}, Z_{i+1} Z_i)\}$, für $2 \leq i < n-1$
- Definiere $\delta(q_{n-1}, \varepsilon, Z_{n-1}) = \{(q', Z_n Z_{n-1})\}$

Wiederhole bis alle Transitionen die gewünschte Form haben. □

Sei $M = (Q, \Sigma, \Gamma, q_0, Z_0, \delta)$ nun ein NPDA wobei $|\gamma| \leq 2$ für alle $(q', \gamma) \in \delta(q, x, Z), q \in Q, x \in \Sigma \cup \{\varepsilon\}, Z \in \Gamma$.

Definiere $\mathcal{G} = (N, \Sigma, S, P)$ mit

- $N = Q \times \Gamma \times Q \cup \{S\}$
- $P = \{(q, Z, q') \to x \mid \delta(q, x, Z) \ni (q', \varepsilon), x \in \Sigma \cup \{\varepsilon\}\}$
  $\cup \{(q, Z, q') \to x(q'', Z', q') \mid \delta(q, x, Z) \ni (q'', Z'), q'' \in Q, x \in \Sigma \cup \{\varepsilon\}\}$
  $\cup \{(q, Z, q') \to x(q_1, Z_1, q_2)(q_2, Z_2, q') \mid \delta(q, x, Z) \ni (q'', Z_1 Z_2),$
  $q_1 \in Q, q_2 \in Q, x \in \Sigma \cup \{\varepsilon\}\}$

Es bleibt zu zeigen, dass $L(\mathcal{G}) = L(M)$.

- Wir zeigen, dass

  $$\text{wenn } (q, Z, q') \stackrel{*}{\Longrightarrow} w \text{ dann } (q, w, Z) \vdash^* (q', \varepsilon, \varepsilon)$$

  per Induktion über den Ableitungsbaum $\mathcal{A} = \pi(\mathcal{A}_1, \dots, \mathcal{A}_n) \in \text{Abl}(\mathcal{G}, (q, Z, q'))$ mit $Y(\mathcal{A}) = w$.

  IV Für $1 \leq i \leq n$ und $Z_i \in \Gamma$, $q_i, q_i' \in Q$, $\mathcal{A}_i \in \text{Abl}(\mathcal{G}, (q_i, Z_i, q_i'))$ gilt

  $$(q_i, Y(\mathcal{A}_i), Z_i) \vdash^* (q_i', \varepsilon, \varepsilon)$$

  IS Es gibt 3 Fälle für π:

  * $(q, Z, q') \to x$, $\delta(q, x, Z) \ni (q', \varepsilon), x \in \Sigma \cup \{\varepsilon\}$.
    Es folgt $w = x$ und damit: $(q, x, Z) \vdash (q', \varepsilon, \varepsilon)$.
  * $(q, Z, q') \to x(q'', Z', q')$, $\delta(q, x, Z) \ni (q'', Z'), q'' \in Q, x \in \Sigma \cup \{\varepsilon\}$
    Es folgt $w = xY(\mathcal{A}_1)$, $\mathcal{A} = \pi(\mathcal{A}_1)$, $\mathcal{A}_1 \in \text{Abl}(\mathcal{G}, (q'', Z', q'))$.
    Es folgt per IV, dass $(q'', Y(\mathcal{A}_1), (q'', Z', q')) \vdash^* (q_0, \varepsilon, \varepsilon)$ und damit auch $(q, xY(\mathcal{A}_1), (q, Z, q')) \vdash (q'', Y(\mathcal{A}_1), (q'', Z', q')) \vdash^* (q_0, \varepsilon, \varepsilon)$.

* $(q, Z, q') \to x(q_1, Z_1, q_2)(q_2, Z_2, q'), \delta(q, x, Z) \ni (q_1, Z_1 Z_2), q_1 \in Q, q_2 \in Q, x \in \Sigma \cup \{\varepsilon\}$

  Es folgt $w = xY(\mathcal{A}_1)Y(\mathcal{A}_2),\ \mathcal{A} = \pi(\mathcal{A}_1, \mathcal{A}_2),\ \mathcal{A}_1 \in \mathrm{Abl}(\mathcal{G}, (q_1, Z_1, q_2)),$ $\mathcal{A}_2 \in \mathrm{Abl}(\mathcal{G}, (q_2, Z_2, q'))$.

  Es folgt per IV, dass

  $$(q_1, Y(\mathcal{A}_1), Z_1)) \vdash^* (q_2, \varepsilon, \varepsilon)$$

  mit Beobachtung $(\star)$ auch

  $$(q_1, Y(\mathcal{A}_1)Y(\mathcal{A}_2), Z_1 Z_2) \vdash^* (q_2, Y(\mathcal{A}_2), Z_2)$$

  Es folgt ferner per IV, dass

  $$(q_2, Y(\mathcal{A}_2), Z_2) \vdash^* (q', \varepsilon, \varepsilon)$$

  Somit gilt

  $$(q, xY(\mathcal{A}_1)Y(\mathcal{A}_2), Z) \vdash (q_1, Y(\mathcal{A}_1)Y(\mathcal{A}_2), Z_1 Z_2)) \vdash^* (q_2, Y(\mathcal{A}_2), Z_2) \vdash^* (q', \varepsilon, \varepsilon)$$

– Wir zeigen für alle $n \in \mathbb{N} : \forall m \in \mathbb{N},\ Z_1, \ldots, Z_m \in \Gamma,\ q, q' \in Q$, dass wenn

$$(q, w, Z_1 \ldots Z_m) \vdash^n (q', \varepsilon, \varepsilon)$$

dann existieren $q_1, \ldots, q_{m+1} \in Q$, so dass

$$(q_1, Z_1, q_2)(q_2, Z_2, q_3) \ldots (q_m, Z_m, q_{m+1}) \overset{*}{\Longrightarrow} w$$

mit $q_1 = q$ und $q_{m+1} = q'$ per Induktion über n.

IA $n = 0$. $w = \varepsilon$, $m = 0$.

Es folgt $\varepsilon \overset{*}{\Longrightarrow} w$.

IV Für alle $n' < n,\ m \in \mathbb{N},\ Z_1, \ldots, Z_m \in \Gamma,\ q, q' \in Q$ gilt, dass wenn

$$(q, w, Z_1 \ldots Z_m) \vdash^{n'} (q', \varepsilon, \varepsilon)$$

dann existieren $q_1, \ldots, q_{m+1} \in Q$, so dass

$$(q_1, Z_1, q_2)(q_2, Z_2, q_3) \ldots (q_m, Z_m, q_{m+1}) \overset{*}{\Longrightarrow} w$$

IS $n > 0$. $(q, w, Z_1 \ldots Z_m) \vdash (q'', w', \gamma Z_2 \ldots Z_m) \vdash^{n-1} (q', \varepsilon, \varepsilon)$. $w = xw'$, $x \in \{\varepsilon\} \cup \Sigma$.

* $\gamma = \varepsilon, \quad (q, Z_1, q'') \to x \in P.$

  Per IV gilt: es existieren $q_2, \ldots, q_{m+1} \in Q$, so dass

  $$(q_2, Z_2, q_3) \ldots (q_m, Z_m, q_{m+1}) \overset{*}{\Longrightarrow} w' \text{ und } q'' = q_2 \text{ und } q' = q_{m+1}.$$

  Somit gilt auch

  $$(q, Z_1, q'')(q_2, Z_2, q_3) \ldots (q_m, Z_m, q_{m+1}) \Longrightarrow x \overset{*}{\Longrightarrow} xw'$$

* ... (andere Fälle ähnlich)

□

Def. 4.3: Ein deterministischer Kellerautomat (DPDA) ist ein Tupel $(\underbrace{Q, \Sigma, \Gamma, q_0, Z_0}_{\text{wie gehabt}}, \delta, F)$

- $F \subseteq Q$ akzeptierende Zustände
- $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to \mathcal{P}(Q \times \Gamma^*)$ wobei für alle $q \in Q, a \in \Sigma, Z \in \Gamma$ gelten muss, dass $|\delta(q, a, Z)| + |\delta(q, \varepsilon, Z)| \leq 1$
- Die Schrittrelation „$\vdash$" ist definiert wie bei NPDAs.
- $L(M) = \{w \in \Sigma^* \mid (q_0, w, Z_0) \vdash^* (q', \varepsilon, \gamma) \wedge q' \in F\}$ ⊕

Lemma 4.3: Zu jedem DPDA M gibt es einen äquivalenten DPDA M', der jede Eingabe bis zum Ende liest.

Entwurf für Vorlesung: 23.12.16

Beweis: Sei $M = (Q, \Sigma, \Gamma, q_0, Z_0, \delta, F)$. Zwei Möglichkeiten, warum M nicht die gesamte Eingabe verarbeiten: Die Transitionsrelation ist nicht total [†] oder der Automat bleibt bei leerem Keller stecken.

Abhilfe: Führe einen Senkzustand ein, auf dem M' weiterrechnet, wenn M nicht mehr weiterrechnen kann. Definiere dazu $M' = (Q', \Sigma, \Gamma', q_0', Z_0', \delta', F')$ mit

$Q' = Q \cup \{q_0', q_s\}$ neue Zustände: q_0' neuer Startzustand und q_s Senkzustand
$F' = F$
$\Gamma' = \Gamma \cup \{Z_0'\}$ neues Kellerbodensymbol

und Transitionsfunktion δ' gegeben durch

$$\delta'(q_0', \varepsilon, Z_0') = \{(q_0, Z_0 Z_0')\}$$

[†] d.h. es gibt zwei Konfigurationen a, b so dass $a \not\vdash^* b$

- Transition totalisieren: Für alle $q \in Q$, $Z \in \Gamma$

$$\delta'(q, \varepsilon, Z) = \begin{cases} \delta(q, \varepsilon, Z) & \text{falls } \neq \emptyset \vee \exists a \in \Sigma, \delta(q, a, Z) \neq \emptyset \\ \{(q_s, Z)\} & \text{sonst} \end{cases}$$

$$\delta'(q, a, Z) = \begin{cases} \delta(q, a, Z) & \text{falls } \neq \emptyset \vee \delta(q, \varepsilon, Z) \neq \emptyset \\ \{(q_s, Z)\} & \text{sonst} \end{cases}$$

  Intuition: Wenn $\delta(q, a, Z) = \emptyset$ und ein ϵ-Übergang ebenfalls ausscheidet, kann das Wort von M nicht weiter abgearbeitet werden. M' geht nun in Senkzustand q_0 und arbeitet dort weiter. Analog für $\delta(q, \epsilon, Z)$.

- Kellerunterlauf: M hat Z_0 abgeräumt, so dass Z_0' sichtbar geworden ist. Füge eine ϵ-Transition in Senkzustand q_s hinzu, indem man für alle $q \in Q$ definiere:

$$\delta'(q, \varepsilon, Z_0') = \{(q_s, Z_0')\}$$

  In q_s kann der Automat nun für alle $a \in \Sigma$, $Z \in \Gamma'$ abarbeiten:

$$\delta'(q_s, a, Z) = \{(q_s, Z)\}$$

  Bemerke, dass $q_0 \notin F'$, da ein Kellerunterlauf das Akzeptieren des eingelesen Wortes ausschliesst.

Es verbleibt zu zeigen, dass $\forall w \in \Sigma^* \; \exists q \in Q' \; \gamma \in \Gamma'^*$ so dass $(q_0', w, Z_0') \vdash^* (q, \varepsilon, \gamma)$. (Induktion über w) Weiter ist zu zeigen, dass $L(M) = L(M')$. □

Satz 4.4: Die deterministischen CFL sind unter Komplement abgeschlossen.

Beweis: Sei $L = L(M)$ für DPDA M. Nach Lemma 4.3 liest M die komplette Eingabe.

Konstruiere DPDA M', s.d. $L(M') = \overline{L}$. Definiere dazu $Q' = q \times \{0, 1, 2\}$. Bedeutung des zusätzlichen „Flags" $\in \{0, 1, 2\}$:

- 0: seit Lesen des letzten Symbols $\epsilon \vee a \in \Sigma$ wurde kein akzeptierender Zustand durchlaufen
- 1: seit Lesen des letzten Symbols wurde mindestens ein akzeptierender Zustand durchlaufen
- 2: markiert einen akzeptierenden Zustand in M'.

$F' = F \times \{2\}$

Definiere Hilfsfunktion $\mathit{final} : Q \to \{0, 1\}$ durch

$$\mathit{final}(q) = \begin{cases} 0 & q \notin F \\ 1 & q \in F \end{cases}$$

$$q_0' = (q_0, \mathit{final}(q_0))$$

Für alle $q \in Q$, $Z \in \Gamma$.

Falls $\delta(q, \varepsilon, Z) = \{(q', \gamma)\}$, dann

$$\delta'((q,0), \varepsilon, Z) = ((q', \mathit{final}(q')), \gamma)$$
$$\delta'((q,1), \varepsilon, Z) = ((q', 1), \gamma)$$

Falls $\delta(q, a, Z) = \{(q', \gamma)\}$, dann

$$\delta'((q,0), \varepsilon, Z) = \{((q,2), Z)\}$$
$$\delta'((q,2), a, Z) = ((q', \mathit{final}(q')), \gamma)$$
$$\delta'((q,1), a, Z) = ((q', \mathit{final}(q')), \gamma)$$

□

Satz 4.5: Die deterministischen CFL sind **nicht** unter Vereinigung und Durchschnitt abgeschlossen.

Beweis: Betrachte

$$L_1 = \{a^n b^n c^m \mid n, m \geq 1\}$$
$$L_2 = \{a^m b^n c^n \mid n, m \geq 1\}$$

Sowohl L_1 als auch L_2 sind DCFL, aber $L_1 \cap L_2 = \{a^n b^n c^n \mid n \geq 1\}$ ist nicht kontextfrei.

DCFL ist nicht abgeschlossen unter Vereinigung. Angenommen doch: Seien U, V DCFL. Dann sind auch $\overline{U}$ und $\overline{V}$ DCFL. Bei Abschluss unter Vereinigung wäre $\overline{U} \cup \overline{V}$ eine DCFL und somit auch $\overline{\overline{U} \cup \overline{V}} = U \cap V$, ein Widerspruch gegen den ersten Teil. □

Satz 4.6: DCFL ist abgeschlossen unter Schnitt mit REG.

Beweis: Sei L DCFL und R regulär. Konstruiere das Produkt aus einem DPDA für L und einem DFA für R. Offenbar ist das Ergebnis ein DPDA, der $L \cap R$ erkennt. †

$L = L(M_1)$ mit $M_1 = (Q_1, \Sigma, \Gamma, q_{01}, Z_0, \delta_1, F_1)$ DPDA

$R = L(M_2)$ mit $M_2 = (Q_2, \Sigma, q_{02}, \delta_2, F_2)$ DFA

Konstruiere $M' = (Q, \Sigma, \Gamma, q_0, Z_0, \delta, F)$ mit

- $Q = Q_1 \times Q_2$

† Intution des Beweis: Konstruiere Produktautomaten, der akzeptiert gdw. der DFA und NDPA akzeptieren. Idee: Definiere Tupel (q_1, q_2), DPDA arbeitet auf q_1, DFA arbeitet auf q_2. Beispielsweise werden ϵ-Übergänge des DPDA nur auf der „DPDA-Seite" d.h. auf q_1 abgearbeitet, ohne Beeinflussung von q_2.

- $q_0 = (q_{01}, q_{02})$
- $F = F_1 \times F_2$
- Falls $\delta_1(q_1, \varepsilon, Z) = \{(q_1', \gamma)\}$, dann gilt für alle $q_2 \in Q_2$:

$$\delta((q_1, q_2), \varepsilon, Z) = \{((q_1', q_2), \gamma)\}. \text{ (Ansonsten leer)}$$

- Falls $\delta_1(q_1, a, Z) = \{(q_1', \gamma)\}$, dann gilt für alle $q_2 \in Q_2$:

$$\delta((q_1, q_2), a, Z) = \{((q_1', \delta_2(q_2, a)), \gamma)\} \text{ (Ansonsten leer)}$$

Zu zeigen ist noch $L(M') = L(M_1) \cap L(M_2)$. □

Satz 4.7: Sei L DCFL und R regulär. Es ist entscheidbar, ob $R = L$, $R \subseteq L$ und $L = \Sigma^*$.

Beweis: Es gilt $R \subseteq L$ gdw. $R \cap \overline{L} = \emptyset$.

Weiter ist $R = L$ gdw. $R \subseteq L$ und $L \subseteq R$. Für den zweiten Teil betrachte $L \cap \overline{R}$.

Für kontextfreie Sprachen ist $L \neq \emptyset$ entscheidbar, also betrachte $L = \Sigma^*$ gdw. $\overline{L} = \emptyset$. □

Satz 4.8: DPDA Äquivalenzproblem Seien L_1, L_2 DCFL. Dann ist $L_1 = L_2$ entscheidbar.

Beweis: Siehe Senizergues (2000) und Stirling (2001). □

Wir betrachten zum Ende des Kapitels noch eine praktische Fragestellung: Wie sieht man einer CFL an, dass sie von einem DPDA erkennbar ist?

Sei $\mathcal{G} = (N, \Sigma, P, S)$. Wir können nach Satz 4.1 einen PDA für $\mathcal{G}$ konstruieren, mit

$$\delta(q, a, a) = \{(q, \varepsilon)\} \quad a \in \Sigma$$
$$\delta(q, \varepsilon, A) = \{(q, \beta) \mid A \to \beta \in P\}$$

Die Transitionen für Eingabezeichen $a \in \Sigma$ sind deterministisch. Die ε-Transitionen sind es nicht unbedingt.

Die Idee ist nun, den Automaten mit einem Symbol Lookahead † erweitern. Dieses Symbol wird jeweils im Zustand des Automaten gespeichert. Der PDA für $L(\mathcal{G})$ mit Lookahead hat nun folgende Komponenten:

†Das Problem ist, dass ein deterministischer PDA nicht „alle mögl. Regeln gleichzeitig ausprobieren" kann, so wie ein NDPA. Möchte der Automat ein Eingabesymbol a matchen und kann aktuell mehrere Produktionen anwenden, müsste er in die Zukunft sehen können, um diejenige zu wählen, die das gewünschte a an erster Position erzeugt. Diesen „Blick in die Zukunft" gewähren die $first(A)$ Mengen: Sie geben für jedes Nichtterminal A an, welche Terminale $a \in \Sigma$ als Präfix von $yield(A)$ auftreten können.

- $Q = \Sigma \cup \{\varepsilon, \$\}$
- $q_0 = [\varepsilon]$ (zu Beginn ist der Lookahead leer)
- $F = [\$]$ (das Symbol \$ markiert das Ende der Eingabe)

und die Transitionsfunktion

$$\begin{aligned}
\delta([\varepsilon], a, Z) &= \{([a], Z)\} && \text{lade Lookahead}\\
\delta([a], \varepsilon, a) &= \{([\varepsilon], \varepsilon)\} && \text{match}\\
\delta([a], \varepsilon, A) &= \{([a], \beta) \mid A \to \beta \in P, a \in \mathit{first}(\beta)\} && \text{select}
\end{aligned}$$

Der Automat startet in der Konfiguration $([\varepsilon], w\$, S\$)$.

Beispiel: Sei ein CFG gegeben mit den Produktionen $S \to (S) \mid a$. Für die Eingabe (a) ergibt sich folgende Abarbeitung:

$$\begin{aligned}
&([\varepsilon], (a), S\$) && \text{Lade Lookahead ,('}\\
\Rightarrow&([(], (a), S\$) && \text{Select: Da ,(' } \in first(\ (a)\ ), \text{ wende } S \to (a) \text{ an}\\
\Rightarrow&([(], (a), (S)\$) && \text{Match ,('}\\
\Rightarrow&([\varepsilon], a), S)\$)\\
\Rightarrow&...
\end{aligned}$$

Def. 4.4: Sei $\mathcal{G} = (N, \Sigma, P, S)$ CFG und $\beta \in (N \cup \Sigma)^*$

$$\mathit{first}(\beta) = \{a \in \Sigma \mid \exists w \in \Sigma^*, \beta \Rightarrow^* aw\} \cup \{\varepsilon \mid \beta \Rightarrow^* \varepsilon\}$$

$\oplus$

Spezifikation von $\mathit{first}(\beta)$

$$\begin{aligned}
\mathit{first}(\varepsilon) &= \{\varepsilon\}\\
\mathit{first}(a\beta) &= \{a\}\\
\mathit{first}(A\beta) &= \begin{cases} \mathit{first}(A) & \varepsilon \notin \mathit{first}(A)\\ \mathit{first}(A) \setminus \{\varepsilon\} \cup \mathit{first}(\beta) & A \Rightarrow^* \varepsilon \end{cases}\\
\mathit{first}(A) &= \bigcup \{\mathit{first}(\beta) \mid A \to \beta \in P\}
\end{aligned}$$

Algorithmus zur Berechnung von $\mathit{first}(A)$ für alle $A \in N$

Sei $FI[A] \subseteq N$ ein Feld indiziert mit Nichtterminalsymbolen.

For each $A \in N$: $FI[A] \leftarrow \emptyset$

Repeat

For each $A \in N$: $FI'[A] \leftarrow FI[A]$

For each $A \to \beta \in P$

$FI[A] \leftarrow FI[A] \cup \mathit{first}_{FI}(\beta)$

Until $\forall A \in N$: $FI'[A] = FI[A]$

Dabei ist

$$\mathit{first}_{FI}(\varepsilon) = \{\varepsilon\}$$
$$\mathit{first}_{FI}(a\beta) = \{a\}$$
$$\mathit{first}_{FI}(A\beta) = \begin{cases} FI[A] & \varepsilon \notin FI[A] \\ FI[A] \setminus \{\varepsilon\} \cup \mathit{first}_{FI}(\beta) & \varepsilon \in FI[A] \end{cases}$$

Beispiel: Betrachte eine Grammatik arithmetische Ausdrücke mit Startsymbol S und $N = \{E, T, F\}$, $\Sigma = \{a, -, *\}$ und Produktionen

$$E \to TE' \qquad E' \to -TE' \mid \varepsilon$$
$$T \to FT' \qquad T' \to *FT' \mid \varepsilon$$
$$F \to a$$

Tabelle der Werte von $FI[A]$ wobei Zeile i den Werten in FI nach dem i-ten Schleifendurchlauf entspricht.

FI	E	E'	T	T'	F
0	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
1	$\emptyset$	$\{-, \varepsilon\}$	$\emptyset$	$\{*, \varepsilon\}$	$, \{a\}$
2	$\emptyset$	$\{-, \varepsilon\}$	$\{a\}$	$\{*, \varepsilon\}$	$, \{a\}$
3	$\{a\}$	$\{-, \varepsilon\}$	$\{a\}$	$\{*, \varepsilon\}$	$, \{a\}$
4	$\{a\}$	$\{-, \varepsilon\}$	$\{a\}$	$\{*, \varepsilon\}$	$, \{a\}$

Ergebnis nach vier Durchläufen.

Anmerkung: first ist nicht die vollständige Lösung des Problems. Für den Lookahead müssen auch noch die Symbole betrachtet werden, die *nach* einem bestimmten Nichtterminal auftreten können. Mehr dazu in Vorlesung Compilerbau.

Entwurf für Vorlesung: 11.1.17

5 Turing und Church

1930er Jahre
Suche nach formalem Modell für maschinelle Berechenbarkeit

Alan Turing: (1912-1954) Turingmaschine 1936

Alonzo Church: Lambdakalkül 1936

Emil Post: Postband 1936

Kleene, Sturgis: partiell rekursive Funktionen

Chomsky: Typ-0-Grammatiken 1956

Alan Turing:
- Informatik, Logik
- Kryptographie (Enigma Entschlüsselung, Sprachverschlüsselung)
- KI (Turing-Test)

außerdem: Turing-Award

5.1 **Turingmaschine** (informell)

Ein primitives Rechenmodell:

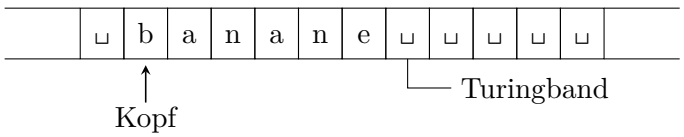


Abb. 12: Turingband
[q] = Zustand

Turingband
- unendliches Band
- Jedes Feld enthält ein Symbol aus einem Bandalphabet Γ.
- uninitialisiert: $\sqcup \in \Gamma$ (Blank) ist ein spezielles Symbol welches ein Feld als „leer" markiert

Kopf
- zeigt immer auf ein Feld
- nur am Kopf kann die TM ein Zeichen lesen und schreiben
- kann nach rechts /links bewegt werden

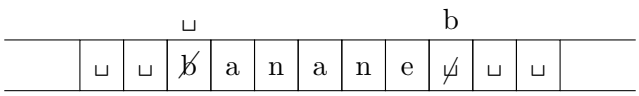
Zustand
- kann verändert werden
- kann gelesen werden
- es gibt nur endlich viele Zustände

Turingtabelle

q	a	q'	a'	d

$\sim$ Programm $\sim$ Transitionsfunktion
$\rightarrow$ Wenn TM in Zustand q und Kopf liest gerade Symbol $a \in \Gamma$ dann wechsle in Zustand q', schreibe a' (über altes a) und bewege den Kopf gemäß $d \in \{L, R, N\}$

Bsp.:



q_0	x	q_x	$\sqcup$	R	$x \neq \sqcup$
q_x	$\sqcup$	q_3	x	L	
q_x	y	q_x	y	R	$y \neq \sqcup$
q_3	y	q_3	y	L	$y \neq \sqcup$
q_3	$\sqcup$	q_4	$\sqcup$	R	

Abb. 13: Bsp.: Turingmaschine—Füge das erste Zeichen am Ende der Eingabe an

Was kann die TM ausrechnen?

1. Die TM kann eine Sprache $L \subseteq \Sigma^*$ erkennen.
   - Wörter müssen auf Band repräsentierbar sein $\Sigma \subseteq \Gamma \setminus \{\sqcup\}$

   Ein Wort w wird von einer TM erkannt, wenn

- zu Beginn steht nur w auf dem Band, alle anderen Zellen $= \sqcup$
- Kopf auf erstem Zeichen von w
- Zustand ist Startzustand q_0
- Abarbeitung der Turingtabelle (TT)
- Falls TM nicht terminiert: $w \notin L$
- Falls TM terminiert betrachte den errechneten Zustand q.
  Falls $q \in F$ (akzeptierender Zustand), dann $w \in L$, anderenfalls $w \notin L$

Bsp.:

$\Sigma = \{0, 1\}$

$L = \{w \in \Sigma^* \mid w \text{ ist Palindrom}\}$

$Q = \{q_0, q_1, q_r^0, q_r^1, q_r^{0'}, q_r^{1'}, q_l^0, q_l^1\} \quad F = \{q_1\}$

q_0	$\sqcup$	q_1	$\sqcup$	N	$q_1 \times q_1 \times N$
q_0	0	q_r^0	$\sqcup$	R	
q_0	1	q_r^1	$\sqcup$	R	
q_r^0	$\sqcup$	q_1	$\sqcup$	N	
q_r^0	0	$q_r^{0'}$	0	R	
q_r^0	1	$q_r^{0'}$	1	R	
$q_r^{0'}$	$\sqcup$	q_l^0	$\sqcup$	L	$q_l \to$ prüfe 0, fahre zum linken Rand und weiter mit q_0
$q_r^{0'}$	0	$q_r^{0'}$	0	R	} Rechtslauf
$q_r^{0'}$	1	$q_r^{0'}$	1	R	} Rechtslauf

Alternative 1:
TM hält bei jeder Eingabe an.

q_l^0	$\sqcup$	$---$	$---$	$---$	$\leftarrow$ Halt
q_l^0	0	q_l	$\sqcup$	L	
q_l^0	1	$---$	$---$	$---$	$\leftarrow$ Halt

Alternative 2:
TM hält nur bei Palindrom an.

q_l^0	$\sqcup$	q_l^0	1	N
q_l^0	0	q_l	$\sqcup$	L
q_l^0	1	q_l^0	$\sqcup$	N

2. Eine TM berechnet eine *partielle* Funktion $f : \Sigma^* \dashrightarrow \Sigma^*$
   Die Berechnung von $f(w),\ w \in \Sigma^*$
   - w auf leeres Band
   - Kopf auf erstes Zeichen, Standardzustand q_0
   - Abarbeitung der TT
   - Falls terminiert und Kopf steht auf dem ersten Symbol von $v \in \Sigma^*$
     Dann $f(w) = v$

Schreibe $A \longrightarrow B$ totale Funktion von A nach B
$A \dashrightarrow B$ partielle Funktion von A nach B

Bsp. 5.1: $\Sigma = \{0, 1\}$
Gesucht eine TM, die die Nachfolgerfunktion auf natürliche Zahlen in Binärdarstellung berechnet.
Annahme: niederwertigste Stelle der Zahl am Anfang der Eingabe.

$\xrightarrow{\text{Start}}$

q_0	$\sqcup$	q_2	1	L
q_0	0	q_1	1	L
q_1	1	q_0	0	R
q_1	$\sqcup$	q_1	$\sqcup$	N ←Halt
q_1	0	q_2	0	L } Linksmaschine
q_1	1	q_2	1	L } Linksmaschine

5.2 Formalisierung der TM

Def. 5.1: Eine TM ist ein 7-Tupel

$$\mathcal{A} = (Q, \Sigma, \Gamma, \delta, q_0, \sqcup, F)$$

- Q ist endliche Menge von Zuständen
- Σ ist endliches Alphabet
- $\Gamma \supsetneq \Sigma$ ist endliches Bandalphabet
- $\delta : Q \times \Gamma \longrightarrow \mathcal{P}(Q \times \Gamma \times \{R, L, N\})$
- $q_0 \in Q$ Startzustand
- $\sqcup \in \Gamma \setminus \Sigma$ das Blank
- $F \subseteq Q$ Menge der akzeptierenden Zustände

Die TM $\mathcal{A}$ heisst *deterministisch* (DTM), falls $\forall q \in Q, \forall a \in \Gamma, |\delta(q, a)| \leq 1$.
Ansonsten ist $\mathcal{A}$ *nichtdeterministisch* (NTM). ⊕

Im Folgenden sei $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, q_0, \sqcup, F)$ eine TM.

Def. 5.2: Eine Konfiguration einer TM ist ein Tupel

$$(v, q, w) \in \text{Konf}(\mathcal{A}) = \Gamma^* \times Q \times \Gamma^+$$

⊕

- v linke Bandhälfte,
- q Zustand,
- w rechte Bandhälfte,
- Kopfposition auf erstem Symbol von w

Abkürzend $vqw \in \mathrm{Konf}(\mathcal{A})$ steht für Band:

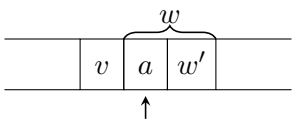


Abb. 14: vqw-Band

Forts.: Die *Startkonfiguration* bei Eingabe w ist: $q_0\, w\, \sqcup$.
Eine *Haltekonfiguration* hat die Form: $vqaw$, so dass $\delta(q,a) = \emptyset$

Def. 5.3: Die *Rechenschrittrelation*

$$\vdash \subseteq \mathrm{Konf}(\mathcal{A}) \times \mathrm{Konf}(A)$$

ist definiert durch

$$\begin{array}{lll}
1. & vqaw \vdash vq'a'w & \text{falls } \delta(q,a) = (q', a', N) \\
2. & vqaw \vdash va'q'w & \text{falls } \delta(q,a) = (q', a', R), w \neq \varepsilon \\
 & va'q'\sqcup & \text{---------- " ----------}, w = \varepsilon \\
3. & qaw \vdash q'\sqcup a'w & \delta(q,a) = (q', a', L) \\
 & vbqaw \vdash vq'ba'w & \text{------- " ------- } b \in \Gamma
\end{array}$$

$\vdash$ Einzelschritt, gesuchte Relation für endlich viele Schritte
$\vdash^* \subseteq \mathrm{Konf}(\mathcal{A}) \times \mathrm{Konf}(\mathcal{A})$ die reflexive transitive Hülle von $\vdash$ $\oplus$

$\vdash$ Relation auf $\mathrm{Konf}(\mathcal{A})\hat{=}$ Berechnungsschritt
$\vdash^*$ reflexiv, transitiver Abschluss $\hat{=}$ endl. viele Berechnungsschritte

Def. 5.4 (Die von TM $\mathcal{A}$ akzeptierte Sprache)**:**

$$\begin{aligned}
L(\mathcal{A}) = \{w \in \Sigma^* \mid\ & q_0 w \vdash^* uqv \\
& uqv \text{ Haltekonfiguration} \\
& q \in F\}
\end{aligned}$$

$\oplus$

Beachte: $w \notin L(\mathcal{A}) \lessgtr \begin{matrix} \mathcal{A} \text{ kann anhalten} \\ \mathcal{A} \text{ kann nicht terminieren} \end{matrix}$

Def. 5.5 (Die von TM $\mathcal{A}$ berechnete Funktion)**:**

$$
\begin{array}{lll}
 & f_{\mathcal{A}} : \Sigma^* - \to \Sigma^* & \\
 & f_{\mathcal{A}}(w) = v & \\
\text{falls} & q_0 w \vdash^* uqv' & \text{Haltekonf.} \\
\text{und} & v = \text{out}(v') &
\end{array}
$$

$$
\begin{array}{ll}
\text{out} : \Gamma^* \to \Sigma^* & \\
\text{out}(\varepsilon) \ = \varepsilon & \\
\text{out}(au) = a \cdot \text{out}(u) & a \in \Sigma \\
\text{out}(bu) = \varepsilon & b \in \Gamma \setminus \Sigma
\end{array}
$$

⊕

Beachte: Falls $q_0 w$ nicht terminiert, dann ist $f_{\mathcal{A}}(w)$ nicht definiert.

Eine TM $\mathcal{A}$ terminiert nicht bei Eingabe w, falls für alle $uq'v$, so dass $q_0 w \vdash^* uq'v$ $uq'v$ ist keine Haltekonfiguration.

5.3 Techniken zur TM Programmierung

- Endlicher Speicher
  Zum Abspeichern eines Elements aus endl. Menge A verwende

$$Q' = Q \times A$$

- Mehrspurmachinen

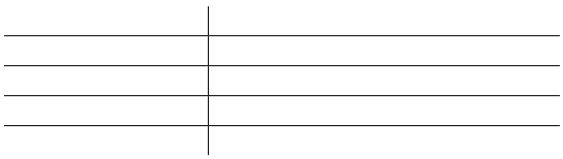


Abb. 15: Mehrspurmachine

Eine k-Spur TM kann gleichzeitig $k \geq 1$ Symbole $\leftarrow \Gamma$ unter dem Kopf lesen. Kann durch Standard TM simuliert werden:

$$\Gamma' = \Sigma \,\dot\cup\, \Gamma^k \text{ mit } \sqcup' = \sqcup^k$$

... vereinfacht die Programmierung

Bsp.: Schulalg. für binäre Addition, Multiplikation

- *Mehrbandmachinenen*
  Eine k-Band TM besitzt $k \geq 1$ Bänder und k Köpfe, die bei jedem Schritt lesen, schreiben und sich unabhängig voneinander bewegen.

$$\delta_K : Q \times \Gamma^k \to Q \times \Gamma^k \times \{R, L, N\}^k$$

- keine herkömmliche TM (für $k > 1$)
- kann durch $2k + 1$ Spur TM simuliert werden:

Spur			
1	——————— Band 1 ———————		
2		# Kopf für Band 1	
3	——————— Band 2 ———————		
4		# Kopf	
	⋮		
$2k$	...	# Kopf k	
$2k+1$	#		##
	linker Rand		rechter Rand

Satz 5.1: Eine k-Band TM kann durch eine 1-Band TM simuliert werden. $M = (Q \dots)$

Beweis: Zeige: ein Schritt der k-Band TM wird durch endlich viele Schritte auf einer 1-Band TM simuliert.

1. Schritt: Kodierung der Konfiguration der k-Band TM
   Definiere M' als TM mit $2k + 1$ Spuren und $\Gamma' = \Gamma \cup \{\#\}$
   - Die Spuren $1, 3, \dots, 2k-1$ enthalten das entspr. Band von M: Band $i \leftrightarrow$ Spur $2i - 1$
   - Die Spuren $2, 4, \dots, 2k$ sind leer bis auf eine Marke $\#$, die auf Spur $2i$ die Position des Kopfes auf Band i markiert
   - Spur $2k + 1$ enthält
     $\#$ Marke für linken Rand
     $\#\#$ Marke für rechten Rand
     Zwischen den beiden Marken befindet sich der bearbeitete Bereich des Bands. D.h. die TM arbeitet zwischen der linken und rechten Marke und schiebt die Marken bei Bedarf weiter.

2. Schritt: Herstellen der Start-Konfiguration.
   Annahme: Eingabe für M auf Band 1
   Jetzt Eingabe (für M') $w = a_1 \dots a_n$

   a) Kopiere w auf Spur 1

   b) Kopf setzen auf Spur $2, \dots, 2k$ an die Position des ersten Symbols von w

   c) auf Spur $2k+1$: #␣##

$2k+1$	#␣##
$2k$	#
$2k-1$	␣
$\vdots$	
4	#
3	␣
2	#
Spur 1	$a_1 a_2 \dots a_2$

   Springe nach $\mathrm{Sim}(q_0)$, der Zustand in M', an dem die Simulation des Zustands q aus M beginnt.

3. Simulation eines Rechnerschritts im Zustand $\mathrm{Sim}(q)$:
   Kopf auf linker Begrenzung, d.h. linker # auf Spur $2k+1$

   - Durchlauf bis rechter Rand, sammle dabei Symbole unter den Köpfen, speichern in endl. Zustand $\overrightarrow{\gamma} \in \Gamma^k$
   - Berechne $\delta(q, \overrightarrow{\gamma}) = (q', \overrightarrow{\gamma'}, \overrightarrow{d})$
     neuer Zustand, für jeden Kopf ein neues Symbol $\overrightarrow{\gamma'}$ und Richtung $\overrightarrow{d}$.
   - Rücklauf nach links, dabei Schreiben um $\overrightarrow{\gamma}'$ und Versetzen der Köpfe gemäß $\overrightarrow{d}$.

   Falls eine Kopfbewegung den Rand auf Spur $2k+1$ überschreitet, dann verschiebe Randmarke entsprechend.

   Beim Rücklauf: Test auf Haltekonfiguration der k-Band TM.
   Falls ja, dann Sprung in Haltekonf. von M'

   Weiter im Zustand $\mathrm{Sim}(q')$.

□

Korollar: Beim Erkunden eines Worts der Länge n benötige die k-Band Maschine M $T(n)$ Schritte und $S(n)$ Zellen auf den Bändern.

- M' benötigt $O(S(n))$ Zellen

- M' benötigt $O(S(n \cdot T(n)))$ Schritte $= O(T(n)^2)$

Weitere TM-Booster

- Unbeschränkt großer Speicher

  $\rightarrow$ für jede „Variable“ ein neues Band

- Datenstrukturen

  ↳ ensprechend kodieren.

$\oplus$

5.4 Das Gesetz von Church-Turing (Churchsche These)

Satz 5.2: Jede intuitiv berechenbare Funktion ist mit TM (in formalem Sinn) berechenbar.

„Intuitiv berechenbar“ $\equiv$ man kann Algorithmus hinschreiben

- endliche Beschreibung
- jeder Schritt effektiv durchführbar
- klare Vorschrift

Status wie Naturgesetz – nicht beweisbar

$\rightarrow$ allgemein anerkannt

$\rightarrow$ weitere Versuche Berechenbarkeit zu formulieren, äquivalent zu TMen erwiesen.

6 Berechenbarkeit

Entwurf für Vorlesung: 18.01.17

6.1 Typ-0 und Typ-1 Sprachen

Def. 6.1: Sei M TM.

- M *akzeptiert* $w \in \Sigma^*$, falls $q_0 w \vdash^* uq'v$ Haltekonfiguration und $q' \in F$
- M *akzeptiert* $L \subseteq \Sigma^*$, falls M akzeptiert $w \leftrightarrow w \in L$
- M *entscheidet* $L \subseteq \Sigma^*$, falls M akzeptiert L und M hält für jede Eingabe an.
- $L \subseteq \Sigma^*$ ist *semi-entscheidbar* (rekursiv aufzählbar), falls $\exists M$, die L akzeptiert.
- $L \subseteq \Sigma^*$ ist *entscheidbar* (rekursiv), falls $\exists M$, die L entscheidet.

$\oplus$

Def. 6.2: Laufzeit und Platzbedarf einer TM M :

$$\text{Laufzeit}: T_M(w) = \begin{cases} \text{Anzahl der Schritte einer kürzesten Berechnung, die zur Akz. von } w \text{ führt (falls } \exists) \\ 1, \text{ sonst} \end{cases}$$

$$\text{Platzbedarf}: S_M(w) = \begin{cases} \text{geringster Platzbedarf (Länge einer Konf.) einer akz. Berechnung von } w \text{ (falls } \exists) \\ 1, \text{ sonst} \end{cases}$$

Zeitbeschränkt mit $t(n)$: $\forall w \in \Sigma^* : |w| \leq n \Rightarrow T_M(w) \leq t(n)$,
platzbeschränkt analog. $\oplus$

Satz 6.1: Zu jeder NTM gibt es eine deterministische TM (DTM) M', so dass

- M' akzeptiert $L(M)$
- M' terminiert gdw. M terminiert
- Falls M zeit- und platzbeschränkt ist mit $t(n)$ bzw. $s(n)$ (n = Länge der Eingabe), dann ist M' zeitbeschränkt mit $2^{O(t(n))}$ und platzbeschränkt mit $O(s(n) \cdot t(n))$.

Beweis: Die Konfigurationen von M bilden einen Baum, dessen Kanten durch $\vdash$ gegeben sind. Er ist endlich verzweigt, hat aber ggf. unendlich lange Äste.

Definiere eine (Mehrband-)DTM, die den Konfigurationsbaum systematisch durchläuft und akzeptiert, sobald eine Haltekonfiguration erreicht ist, in der M akzeptiert.

Die DTM terminiert ebenfalls, wenn alle Blätter des Baumes besucht worden sind, ohne dass eine akzeptierende Konfiguration gefunden wurde.

Baumsuche mit Kontrollinformation und bereits besuchten Konf. auf ein Extraband.

- Tiefensuche? Nicht geeignet, sie könnte in unendlichen Ast laufen.
- Breitensuche? OK, aber Platzbedarf $O(2^{t(n)} \cdot s(n))$

- iterative deepending : Tiefensuche mit vorgegebener Schranke, bei erfolgloser Suche Neustart mit erhöhter Schranke. □

Nächstes Ziel: Charakterisierung von Typ-1 Sprachen.

Def. 6.3:

- DTAPE($s(n)$) : Menge der Sprachen, die von einer DTM in Platz $s(n)$ akzeptiert werden können.
- NTAPE($s(n)$) : Wie für DTAPE, aber mit Eine nichtdeterministische TM (NTM). ⊕

Bemerkung:

1. Für „$s(n) \leq n$“ betrachte 2-Band TM, bei denen die Eingabe read-only ist und nur das zweite Arbeitsband der Platzschranke unterliegt (so ist $s(n)$ sublinear möglich).
2. Jede Platzbeschränkung impliziert Laufzeitschranke.
   Angenommen Platzschranke $s(n)$
   $\curvearrowright$ TM hat nur endlich viele Konfigurationen

$$N := \underbrace{n}_{\substack{\uparrow \\ \text{Kopfpos. im} \\ \text{Eingabeband}}} \underbrace{|Q|}_{\substack{\uparrow \\ \text{mögliche Inhalte} \\ \text{des} \\ \text{Arbeitsbands}}} \cdot \underbrace{|\Gamma|^{s(n)}}_{\substack{\uparrow \\ \text{Kopfpos. auf} \\ \text{Arbeitsband}}} \cdot s(n) \in 2^{O(\log n + s(n))}$$

3. DTM mit Platzschranke : M entscheidet,
   falls sie akzeptiert, dann in weniger als N Schritten,
   falls nach N Schritten keine Termination erfolgt
   $\curvearrowright$ Endlosschleife – Abbruch
4. NTM : nutze den Nicht-Determinismus (ND) optimistisch aus :
   falls eine akzeptierende Berechnung existiert, dann muss es eine Berechnung ohne wiederholte Konfiguration geben.

Satz 6.2:

- $L \in \text{DTAPE}(n) \curvearrowright \exists$ DTM, die L in Zeit $2^{O(n)}$ entscheidet.
- $L \in \text{NTAPE}(n)$ analog.

Beweis: siehe oben. □

Bemerkung: Die Klasse NTAPE(n) heißt auch Linear Bounded Automaton (LBA).

Satz 6.3: $\mathcal{L}_1 = \mathrm{NTAPE}(n)$

Beweis:

„⇒“: Sei $G = (N, \Sigma, P, S)$ Typ-1 Grammatik für L.
Konstruiere NTM M mit $L = L(M),\ \Gamma = \Sigma \cup N \cup \{\sqcup\}$

1. M rät nicht deterministisch eine Position auf dem Band und eine Produktion $\alpha \to \beta$. Falls β gefunden wird, ersetze durch α, weiter bei 1.
2. Falls Bandinhalt $= S$ stop, akzeptiert.

Dieses Verfahren terminiert.

„⇐“: Gegeben: NTM M linear beschränkt.
Gesucht: Typ-1 Grammatik $\mathcal{G}$ mit $L(\mathcal{G}) = L(M)$
Idee:

$$a_1 \cdots a_n \longrightarrow \begin{pmatrix} a_1 \\ a_1 \end{pmatrix} \begin{pmatrix} a_2 \\ a_2 \end{pmatrix} \begin{pmatrix} (q,a) \\ a_3 \end{pmatrix} \begin{pmatrix} a_4 \\ a_4 \end{pmatrix} \begin{pmatrix} a_n \\ a_n \end{pmatrix} \qquad \begin{matrix} \text{Spur 1} \\ \text{Spur 2} \end{matrix}$$

ad[†] Spur 1: Alphabet $\Gamma \cup (Q \times \Gamma) = \triangle$

$$P' \begin{cases} (q,a) & \to (q',a') \quad q \in Q, a \in \Gamma & \delta(q,a) \ni (q',a',N) \\ (q,a)b & \to a'(q',b) \quad b \in \Gamma & \delta(q,a) \ni (q',a',R) \\ b(q,a) & \to (q',b)a' & \delta(q,a) \ni (q',a',L) \end{cases}$$

Def. $\widetilde{uqav} = u(q,a)v\ ,\ u,v \in \Gamma^*, a \in \Gamma$
Es gilt: $uqav \vdash^* k' \curvearrowright \widetilde{uqav} \overset{*}{\Rightarrow} \widetilde{k'}$ mit Produktion P'.

[†] ad ≈ zur

Def. $\mathcal{G}$ durch $N = \{S\} \dot\cup \triangle \times \Sigma$

mit $P =$

$$
\begin{array}{rll}
S \to \begin{pmatrix}(q_0, a)\\ a\end{pmatrix} & & \forall a \in \Sigma \\
S \to S \begin{pmatrix}a\\ a\end{pmatrix} & & \forall a \in \Sigma \\
\begin{pmatrix}\alpha\\ a\end{pmatrix} \to \begin{pmatrix}\beta\\ a\end{pmatrix} & & \forall \alpha \to \beta \in P' \\
& & \quad \alpha, \beta \in \triangle \\
\begin{pmatrix}\alpha_1\\ a_1\end{pmatrix}\begin{pmatrix}\alpha_2\\ a_2\end{pmatrix} \to \begin{pmatrix}\beta_1\\ a_1\end{pmatrix}\begin{pmatrix}\beta_2\\ a_2\end{pmatrix} & & \forall \alpha_1\alpha_2 \to \beta_1\beta_2 \in P' \\
& & \quad \alpha_i, \beta_i \in \triangle \\
\begin{pmatrix}x\\ a\end{pmatrix} \to a & & x \in \Gamma \\
& & a \in \Sigma \\
\begin{pmatrix}(q', x)\\ a\end{pmatrix} \to a & & x \in \Gamma, q' \in F, \delta(q', x) = \emptyset \\
& & a \in \Sigma
\end{array}
$$

$$
\begin{aligned}
S &\overset{*}{\Rightarrow} \begin{pmatrix}(q_0, a_1)\\ a_1\end{pmatrix}\begin{pmatrix}a_2\\ a_2\end{pmatrix}\dots\begin{pmatrix}a_n\\ a_n\end{pmatrix} \\
&\quad \text{TM } \dots \\
&\overset{*}{\Rightarrow} \begin{pmatrix}x_1\\ a_1\end{pmatrix}\dots\begin{pmatrix}(q', x_i)\\ a_i\end{pmatrix}\dots\begin{pmatrix}x_n\\ a_n\end{pmatrix} \\
&\overset{*}{\Rightarrow} a_1 \dots a_i \dots a_n
\end{aligned}
$$

Damit gesehen $L(\mathcal{G}) \subseteq L(M)$
Rückrichtung: selbst □

Satz 6.4: Die Typ-1 Sprachen sind abgeschlossen unter $\cup$, $\cap$, $\cdot$, * und Komplement.

Beweis: Zu $\cup$ und $\cap$ betrachte NTM.
Für $\cdot$ und * konstruiere Grammatik.
ad Komplement „2. LBA-Problem[†]“ bis 1987, dann gelöst durch Immerman und Szelepcsényi. □

Bemerkung: 1. LBA-Problem (1964): Ist NTAPE(n) = DTAPE(n)? Bisher ungelöst.

[†]LBA = Linear Bounded Automaton – 1964 Kuroda

Satz 6.5: Das Wortproblem für Typ-1 Sprachen ist entscheidbar.

Beweis:

$$L \in \mathcal{L}_1 \curvearrowright L \in \mathrm{NTAPE}(n)$$
$$\curvearrowright \text{ nach Satz 6.2: } L \text{ entscheidbar}$$

Nach Satz 6.1 sogar mit DTM. □

Die Rückrichtung „L entscheidbar. $\not\curvearrowleft L$ ist Typ-1 Sprache“ gilt nicht!

Satz 6.6: $\mathcal{L}_0 = \mathrm{NTM}$

Beweis: “$\Rightarrow$“ Kontruktion einer NTM M wie in Satz 6.3, aber ohne Platzbeschränkung.

“$\Leftarrow$“ Konstruktion analog zu Satz 6.3 + Startsymbol S'

$$S' \to \begin{pmatrix} \sqcup \\ \varepsilon \end{pmatrix} S' \begin{pmatrix} \sqcup \\ \varepsilon \end{pmatrix} \qquad \text{Schaffe Platz für Berechnung von } M$$
$$S' \to S$$

Erweitere N

$$= \{S', S\} \cup \triangle \times (\Sigma \cup \{\varepsilon\})$$

Neue Löschregeln:

$$\begin{pmatrix} x \\ \varepsilon \end{pmatrix} \to \varepsilon \qquad \forall x \in \Gamma$$

$\leftharpoondown$ die einzigen Regeln, die Typ-1 Bedingung verletzen. □

Satz 6.7: Die Typ-0 Sprachen sind unter $\cup$, $\cap$, $\cdot$, * abgeschlossen.

Beweis: Konstruiere NTM für $\cup$, $\cap$; Typ-0-Grammatiken für $\cdot$ und *. □

Bem.: Typ-0 Sprachen sind *nicht* unter Komplement abgeschlossen!

6.2 Universelle TM und das Halteproblem

Ziel: Universelle TM (eine TM, die TMs interpretiert) ist eine TM U, die zwei Eingaben nimmt:

1. Kodierung einer (anderen) TM M_0
2. Eingabe w für M_0

so dass

$$w \in L(M_0) \curvearrowright (M_0, w) \in L(U)$$
$$M_0 \text{ terminiert bei Eingabe } w \curvearrowright U \text{ terminiert bei Eingabe } (M_0, w)$$

Zur Vereinfachung:

$$\Sigma = \{0, 1\}$$
$$\Gamma = \{\sqcup, 0, 1\}$$

Die Kodierung von M_0

$$= (Q, \ldots, q_1, \delta, \ldots, \{q_2\})$$

mit $Q = \{q_1, q_2, \ldots, q_t\}$ ist im wesentlichen die Kodierung von δ. Dazu zwei Hilfsfunktionen:

$x \in \Gamma$	$f(x)$
0	1
1	2
$\sqcup$	3

D	$g(D)$
L	1
N	2
R	3

Kodiere δ Zeilenweise:

$$\delta(q_i, X) = (q_k, Y, D) \ (= \text{Zeile} z)$$

durch

$$\text{code}(z) = 0^i 10^{f(x)} 10^k 10^{f(y)} 10^{g(D)}$$

Wenn δ durch s Zeilen $z_1 \ldots z_s$ gegeben, dann definiere die *Gödelnummer* von M_0 durch

$$\ulcorner M_0 \urcorner = 111\text{code}(z_1)11\text{code}(z_2)11 \ldots 11\text{code}(z_s)111$$

Definiere U als 3-Band Maschine mit

B_1 : Eingabe + Arbeitsband (für M_0)

B_2 : $\ulcorner M_0 \urcorner$

B_3 : 0^k für Zustand q_k

1. Schritt: Transformierte Eingabe $\ulcorner M_0 \urcorner$

- Beginnt die Eingabe mit gültiger Gödelnummer?
- Gleichzeitig: Verschiebe $\ulcorner M_0 \urcorner$ auf B_2
- Wenn Ende von $\ulcorner M_0 \urcorner$ erreicht, schiebe 0 auf B_3.

Jetzt:

B_1 : w

$B_2 : \ulcorner M_0 \urcorner$

$B_3 : 0' \sim Z, q$

Satz 6.8: Es gibt eine universelle TM U mit $L(U) = \{\ulcorner M \urcorner w \mid w \in L(M)\}$

BEWEIS: Initialisierung:

$B_1 :$ w Eingabewort/Arbeitsband

$B_2 : \ulcorner M \urcorner$ Gödelnummer

$B_3 :$ 0 Zustand

Hauptschleife von U:

- Test auf Haltekonfiguration.
- Falls ja: Falls in qz : akzeptiert.
  sonst : nicht
- Ausführung des nächsten Schritts:
  Suche Zeile in $\ulcorner M \urcorner$ gemäß Zustand und aktuellem Symbol auf Arbeitsband. Ändere B_1 und B_3 gemäß δ.
  B_2, B_3: Kopf zurück zum Anfang.

□

Schreibe ab jetzt M_w für die Maschine mit Gödelnummer w. Falls w kein gültiger Code für eine TM, dann sei M_w eine beliebige fest TM M mit $L(M) = \varnothing$.

Def. 6.4: Das spezielle Halteproblem besteht aus allen Codes von Maschinen, die anhalten, falls sie auf den eigenen Code angesetzt werden.

$$K = \{w \in \{0,1\}^* \mid M_w \text{ angesetzt auf } w \text{ terminiert}\}$$

⊕

Satz 6.9: Das spezielle Halteproblem ist unentscheidbar.

BEWEIS: Angenommen M ist eine TM, die K entscheidet.

Konstruiere Maschine M', die zunächst M auf ihre Eingabe anwendet. Falls M akzeptiert, dann geht M' in eine Endlosschleife. Falls M nicht akzeptiert, dann hält M' an.

Sei $w' = \ulcorner M' \urcorner$ der Code von M' und setze M auf w' an. Es gilt:

M akzeptiert w'

gdw.(nach Def von K) M' angesetzt auf w' terminiert

gdw. M akzeptiert w' nicht.

Ein Widerspruch. ↯ □

Korollar 6.10: $\overline{K} = \{w \in \{0,1\}^* \mid M_w \text{ hält nicht bei Eingabe } w\}$, das Komplement von K, ist nicht entscheidbar. $\oplus$

BEWEIS: Angenommen $\overline{K}$ sei entscheidbar durch M. Dann entscheidet M' K. M' führt zuerst M aus und negiert das Ergebnis. $\lightning$ Satz 6.9 □

Lemma 6.11: K ist semi-entscheidbar.

BEWEIS: Die Maschine M kopiert die Eingabe w und f"uhrt die universelle TM f"ur M_w aus. Falls diese Simulation stoppt, geht M in einen akzeptierenden Zustand und terminiert. □

Satz 6.12: Falls L semi-entscheidbar und $\overline{L}$ semi-entscheidbar, dann ist L entscheidbar.

BEWEIS: Sei M die TM für L, $\overline{M}$ die TM für $\overline{L}$.
Führe M und $\overline{M}$ „parallel“ mit der gleichen Eingabe aus.
Falls M akzeptiert ⇒ Ja
Falls $\overline{M}$ akzeptiert ⇒ Nein.
Eine der Maschinen muss anhalten, wegen Voraussetzung. □

K nicht entscheidbar
$\overline{K}$ nicht entscheidbar
K semi-entscheidbar (Typ-0)
$\overline{K}$ nicht-semientscheidbar (keine Typ-0)

Korollar 6.13: $\mathcal{L}_0 \supsetneq \mathcal{L}_1$ $\oplus$

BEWEIS: K ist unentscheidbar (also $\notin \mathcal{L}_1$), aber semi-entscheidbar (also $\in \mathcal{L}_0$). □

Fragen:

1. Ist $\mathcal{L}_1$ = Menge der entscheidbaren Sprachen?
   - Nein:
     Konstruiere eine Kodierung von Typ-1 Grammatiken als Worte $w \in \{0,1\}^*$. Die Grammatik zum Wort w sei G_w; falls w kein sinnvoller Kode ist, setze $G_w = (\{S\}, \{0,1\}, \{\}, S)$ die leere Grammatik.
     Die *Diagonalsprache* $D = \{w \in \{\mathbf{0},1\}^* \mid w \notin L(G_w)\}$ ist entscheidbar, weil das Wortproblem für Typ-1 Sprachen entscheidbar, aber es $\nexists w$, sodass $L(G_w) = D$. Beweis durch Widerspruch.

	w_1 w_2 w_3 $\cdots$
G_1	
G_2	
G_3	
$\vdots$	

2. Ist $\mathcal{L}_0$ = Menge aller Sprachen?
   - Nein: $\overline{K} \notin \mathcal{L}_0$

Def. 6.5 (Reduktion)**:**
Seien $U, V \subseteq \Sigma^*$ Sprachen.
U ist auf V reduzierbar ($U \preceq V$), falls eine totale berechenbare Funktion $f : \Sigma^* \to \Sigma^*$ existiert, so dass $\forall x \in \Sigma^* : x \in U \iff f(x) \in V$. $\oplus$

Lemma 6.14: Falls $U \preceq V$ und V (semi-)entscheidbar, dann ist auch U (semi-)entscheidbar.

Beweis: Wenn M ein (Semi-)Entscheidungsverfahren für V ist, dann konstruiere M' wie folgt

- wende erst f auf die Eingabe x an (f ist berechenbare Funktion gemäß Reduktion und kann daher programmiert werden)
- führe M auf dem Ergebnis $f(x)$ aus

$\curvearrowright M'$ ist (Semi-)Entscheidungsverfahren für U. (Weil f total ist, terminiert der Code f"ur f immer und daher ändert das Terminationsverhalten nicht.) $\square$

Anwendung: $U \preceq V$ und U unentscheidbar $\curvearrowright V$ unentscheidbar.

Def. 6.6: Das *Halteproblem* ist definiert durch

$$H = \{\ulcorner M \urcorner \# w \mid M \text{ hält bei Eingabe } w \text{ an}\} \qquad \oplus$$

Satz 6.15: H ist unentscheidbar.

Beweis: Die Funktion $f(w) = w\#w$ ist total berechenbar und liefert eine Reduktion $K \preceq H$.

Denn: $w \in K$ gdw. M_w hält bei Eingabe w an gdw. $w\#w \in H$. $\square$

Satz 6.16: H ist semi-entscheidbar.

Beweis: Modifiziere U, sodass sie jede Eingabe akzeptiert, bei der sie anhält. $\square$

Def. 6.7: Das Halteproblem auf leerem Band $H_\varepsilon = \{\ulcorner M \urcorner \mid M \text{ terminiert auf leeren Band}\}$ $\oplus$

Satz 6.17: H_ε ist unentscheidbar.

Beweis: Konstruiere eine Reduktion $H \preceq H_\varepsilon$ mit Hilfe der Funktion $f(w\#x) = w'$, wobei w' der Code einer TM ist, die

- zuerst x aufs leere Band schreibt und dann
- M_w auf diese Eingabe anwendet.

Offenbar gilt $w\#x \in H$ gdw. $f(w\#x) \in H_\varepsilon$. $\square$

Nun betrachten wir TMs vom Blickwinkel der von ihnen berechneten (partiellen) Funktionen. Sei R die Menge der von TMs berechneten Funktionen.

Satz 6.18 (Satz von Rice)**:**
Sei R die Menge aller partiellen TM-berechenbaren Funktionen und $\varnothing \neq S \subsetneq R$ eine nichttriviale (nicht-leere, echte) Teilmenge davon.
Dann ist $L(S) = \{\ulcorner M \urcorner \mid M \text{ berechnet Funkt. aus } S\}$ unentscheidbar.

Beweis: Angenommen M_S entscheidet $L(S)$.
Sei $\Omega \in R$ die überall undefinierte Funktion. Wir nehmen an, dass $\Omega \in S$ (anderenfalls betrachten wir $\overline{L(S)}$).

Da $R \setminus S \neq \varnothing$ gibt es eine berechenbare Funktion $f \in R \setminus S$ und f werde von TM M_f berechnet.

Definiere $M' = M'_{(M,f)}$ wie folgt: M' führt zunächst M (beliebige TM) auf leerer Eingabe aus. Falls M anhält, wendet M' dann M_f auf die tatsächliche Eingabe an.

Die von M' berechnete Funktion ist also $f_{M'} = \begin{cases} f & \text{falls } M \text{ auf leerem Band hält,} \\ \Omega & \text{sonst.} \end{cases}$

Definiere nun M'' wie folgt:

- Bei Eingabe $\ulcorner M \urcorner$ berechne die Gödelnummer von M'.
- Wende nun M_S auf $\ulcorner M' \urcorner$ an.

$$\begin{aligned} M_s \text{ akzeptiert } \ulcorner M' \urcorner &\iff M' \text{ berechnet Funktion in } S \\ &\iff M' \text{ berechnet } \Omega \ (f_{M'} \in \{\Omega, f\}, \Omega \in S, f \notin S\ ) \\ &\iff M \text{ hält } \textit{nicht} \text{ auf leerem Band an} \end{aligned}$$

Also entscheidet M'' H_ε. $\quad \lightning$ □

6.3 Eigenschaften von entscheidbaren und semi-entscheidbaren Sprachen

Satz 6.19: Seien L_1 und L_2 entscheidbar. Dann sind $\overline{L_1}$, $\overline{L_2}$, $L_1 \cup L_2$ und $L_1 \cap L_2$ entscheidbar.
Beweis: Übung oder selbst. □

Satz 6.20: Seien L_1 und L_2 semi-entscheidbar. Dann sind $L_1 \cup L_2$ und $L_1 \cap L_2$ semi-entscheidbar.

Beweis: vgl. Satz 6.7. □

Satz 6.12 (Wiederholung)**:** Falls L semi-entscheidbar und $\overline{L}$ semi-entscheidbar, dann ist L entscheidbar.

Satz 6.21: Die Menge der semi-entscheidbaren Sprachen ist *nicht* unter Komplement abgeschlossen.

Beweis: Laut Satz 6.9 und Korollar 6.10 sind das spezielle Halteproblem K und $\overline{K}$ nicht entscheidbar.

K ist semi-entscheidbar, aber nicht $\overline{K}$. □

6.4 Weitere unentscheidbare Probleme

Entwurf für Vorlesung: 1.2.17

Das Postsche Korrespondenzproblem (PCP)

Gegeben:
Endliche Folge von Wortpaaren $K = ((x_1, y_1), \ldots, (x_k, y_k))$ mit $x_i, y_i \in \Sigma^+$

Gesucht:
Indexfolge $i_1, \ldots, i_n \in \{1, \ldots, k\}$ $(n \geq 1)$, so dass $x_{i_1} \cdots x_{i_n} = y_{i_1} \cdots y_{i_n}$

Die Folge $i_1, \ldots, i_n$ (falls diese existiert) heißt *Lösung* des Korrespondenzproblems K.

Bsp.:

$$K = ((\underbrace{1, 101}_{x_1, y_1}), (\underbrace{10, 00}_{x_2, y_2}), (\underbrace{011, 11}_{x_3, y_3}))$$

besitzt die Lösung $(1, 3, 2, 3)$, denn

$$x_1 x_3 x_2 x_3 = \underbrace{1 \cdot 01}_{y_1} \underbrace{1 \cdot 1}_{y_3} \underbrace{0 \cdot 0}_{y_2} \underbrace{11}_{y_3} = y_1 y_3 y_2 y_3$$

Frage:

$x_1 = 001$	$x_2 = 01$	$x_3 = 01$	$x_2 = 10$
$y_1 = 0$	$y_2 = 011$	$y_3 = 101$	$y_2 = 001$

Besitzt dieses PCP eine Lösung? Ja, aber mit 66 Indizes [Schöning, S.124]

Bemerkung:
Offensichtlich ist das PCP semi-entscheidbar: Systematisches Ausprobieren von Indexfolgen findet Lösung nach endlicher Zeit, *sofern es eine gibt.*

Ziel: PCP ist unentscheidbar. Vorbereitung: Es interessiert uns ab hier nur, ob das Problem eine Lösung hat oder nicht.

Das modifizierte PCP (MPCP)
Gegeben: wie bei PCP

Gesucht: Lösung des CP mit $i_1 = 1$

Lemma 6.22: MPCP $\preceq$ PCP

Beweis: Betrachte MPCP $K = ((x_1, y_1), \dots (x_k, y_k))$ über Σ.
Sei $\Sigma' = \Sigma \uplus \{\#, \$\}$
Für ein Wort $w = a_1 \dots a_n \in \Sigma^+$ sei

$$\begin{aligned} \bar{w} &= \#a_1\#a_2\# \dots \#a_n\# \\ \grave{w} &= \#a_1\#a_2\# \dots \#a_n && \text{(am Ende kein \#)} \\ \acute{w} &= \quad a_1\#a_2\# \dots \#a_n\# && \text{(am Anfang kein \#)} \end{aligned}$$

Definiere nun

$$f(K) = (\underbrace{(\bar{x}_1, \grave{y}_1)}_{1}, \underbrace{(\acute{x}_1, \grave{y}_1)}_{2}, \underbrace{(\acute{x}_2, \grave{y}_2)}_{2+1}, \dots, \underbrace{(\acute{x}_k, \grave{y}_k)}_{k+1}, \underbrace{(\$, \#\$)}_{k+2})$$

eine totale berechenbare Funktion.

Zeige $K \in \text{MPCP} \iff f(K) \in \text{PCP}$:

„$\Longrightarrow$“: $1, i_2, \dots, i_n$ Lösung für K
$\curvearrowright$ $1, i_2 + 1, \dots, i_n + 1, k + 2$ Lösung für $f(K)$

„$\Longleftarrow$“:

1. Sei $i_1, \dots, i_n$ Lösung für $f(K)$, in der das Paar $k + 2$ höchstens einmal vorkommt.
   $\curvearrowright$ Durch die Stuktur der Worte gilt für Lösungen immer: $i_1 = 1$, $i_n = k + 2$ (ansonsten fehlt am Anfang oder am Ende das Symbol #)
   Es gilt ferner für $1 < j < n$
   - $i_j \neq 1$, da in der x-Konkatenation sonst # doppelt vorkommt, was in der y-Konkatenation nicht möglich ist.
   - $i_j \neq k + 2$, da $k + 2$ per Annahme nur einmal vorkommt.

   Also gilt für $1 < j < n$: $i_j \in \{2, \dots, k + 1\}$.
   $\curvearrowright$ $1, i_2 - 1, \dots, i_{n-1} - 1$ Lösung für K
2. Sei $i_1, \dots, i_n$ Lösung für $f(K)$, in der das Paar $k + 2$ mehrmals vorkommt. Dann gibt es auch eine Lösung $i_m, \dots, i_{m+l}$ mit $1 \leq m \leq m + l \leq n$ so dass $k + 2$ nur einmal vorkommt (ohne Beweis). Weiter bei 1.

□

Es reicht nun zu zeigen, dass MPCP unentscheidbar ist!

Lemma 6.23: H $\preceq$ MPCP

Beweis: TM $M = (Q, \Sigma, \Gamma, \delta, q_0, \sqcup, F)$ und Eingabewort $w \in \Sigma^*$.

Gesucht: totale berechenbare Funktion, die $(\ulcorner M \urcorner, w) \mapsto \underbrace{(x_1, y_1), \ldots, (x_k, y_k)}_{k}$, sodass

$\ulcorner M \urcorner w \in H$ gdw K eine Lösung als MPCP besitzt.

Idee: Definiere K so, dass die Berechnung von M simuliert wird.
Alphabet für $K : \triangle = \Gamma \cup Q \cup \{\#\}$
$(x_1, y_1) = (\#, \#q_0 w \#)$

1. Kopieren
$$(a, a) \quad , a \in \Gamma \cup \{\#\}$$

2. Transition ($a \in \Gamma$)
$$\begin{array}{rrl}
(qa, q'a') & \forall q, a : \delta(q, a) & \ni (q', a', N) \\
(qa, a'q') & \ldots & \ni (q', a', R) \\
(bqa, q'ba') & & \ni (q', a', L), b \in \Gamma \\
(q\#, q'a'\#) & \forall q : \delta(q, \sqcup) & \ni (q', a', N) \\
(q\#, a'q'\#) & & \ni (q', a', R) \\
(bq\#, q'ba'\#) & & \ni (q', a', L), b \in \Gamma \\
(\#qa, \#q'\sqcup a') & \forall q, a : \delta(q, a) & \ni (q', a', L) \\
(\#q\#, \#q'\sqcup a'\#) & \forall q : \delta(q, \sqcup) & \ni (q', a', L)
\end{array}$$

3. Löschen
$$(aqb, qb) \text{ für } a, b \in \Gamma \text{ und } \delta(q, b) = \emptyset$$
$$(qba, qb) \text{ für } a, b \in \Gamma \text{ und } \delta(q, b) = \emptyset$$

4. Abschluss
$$(qb\#\#, \#) \text{ für } b \in \Gamma \text{ und } \delta(q, b) = \emptyset (q\#\#, \#) \text{ für } \delta(q, \sqcup) = \emptyset$$

$\ulcorner M \urcorner w \in H$
$\iff$ Folge von Konf von M, $k_0 \ldots k_t$ mit $k_0 = q_0 w$ und $k_t = uq'bv$ mit $\delta(q', b) = \emptyset$ mit

$h_{i-1} \vdash k_i \quad \forall 1 \leq i \leq t$
$\Longleftrightarrow$ Die Instanz K von MPCP besitzt Lösung und ein Lösungswort der Form

$$\#k_0\#k_1\#\ldots\#h_t\#k_t^1\#k_t^2\#\ldots\#q'b\#\#$$

oder

$$\#k_0\#k_1\#\ldots\#h_t\#k_t^1\#k_t^2\#\ldots\#q'\#\#$$

wobei $k_t^0 = k_t$ und k_t^j durch Streichen eines Bandsymbols rechts oder links von q' bzw $q'b$ aus ihrem Vorgänger k_t^{j-1}entsteht.

Intuition: "Die Konkatenation der x_is hinkt immer um eine Konfiguration der Konkatenation der y_is hinterher". □

Satz 6.24: PCP ist unentscheidbar.

Beweis: H $\leq$ MPCP und MPCP $\leq$ PCP □

Entwurf für Vorlesung: 3.2.17

Satz 6.25: Das Schnittproblem „$L(\mathcal{G}_1) \cap L(\mathcal{G}_2) \neq \varnothing$?" für CFL ist unentscheidbar.

Beweis: Durch Reduktion PCP $\leq$ Schnittproblem.
Sei $K = \{(x_i, y_i) \mid 1 \leq i \leq k\}$ Instanz von PCP über Σ.
Berechne aus K zwei CFG $\mathcal{G}_1$ und $\mathcal{G}_2$, so dass K eine Lösung hat $\Longleftrightarrow L(\mathcal{G}_1) \cap L(\mathcal{G}_2) \neq \varnothing$

$$\begin{aligned} \mathcal{G}_1 : S_1 &\to 1x_1 | \ldots | kx_k \\ &\quad\ |1S_1x_1| \ldots |kS_1x_k \\ \mathcal{G}_2 : S_2 &\to 1y_1 | \ldots | ky_k \\ &\quad\ |1S_2x_1| \ldots |kS_2y_k \end{aligned} \qquad \text{Alphabet} : \Sigma \cup \{1 \ldots k\}$$

$$\begin{aligned} & w \in L(\mathcal{G}_1) \cap L(\mathcal{G}_2) \\ \Longleftrightarrow\ & w = k_n \ldots k_1, xk_1 \ldots xk_n \\ & \quad = k_n \ldots k_1, yk_1 \ldots yk_n \\ \Longleftrightarrow\ & (k_1 \ldots k_n) \text{ ist Indexfolge zur Lösung von PCP } k \end{aligned}$$ □

Folgerung : Schnittproblem für Typ 1 und Typ 0 Sprachen ist ebenfalls unentscheidbar.

Korollar 6.26: Das Schnittproblem ist auch für deterministische CFL (DCFL) unentscheidbar. ⊕

Beweis: $L(G_1)$ ist auch DPDA erkennbar. □

Satz 6.27: Das Äquivalenzproblem für CFL ist unentscheidbar.

BEWEIS: Sei $A = \{\mathcal{G}_1, \mathcal{G}_2 \mid L(\mathcal{G}_1) = L(\mathcal{G}_2)\}$
Angenommen $\mathcal{G}_1, \mathcal{G}_2$ sind Typ 2 Grammatiken für DCFL.
Dann ist $(\mathcal{G}_1, \mathcal{G}_2) \in$ Schnittproblem.

$$\iff L(\mathcal{G}_1) \cap L(\mathcal{G}_2) = \varnothing$$
$$\iff L(\mathcal{G}_1) \subseteq \overline{L(\mathcal{G}_2)}$$

Da $\mathcal{G}_2$ eine deterministische CFG (DCFG) $\exists\, \mathcal{G}_2$ mit $L(\mathcal{G}_2') = \overline{L(\mathcal{G}_2)}$ (Abschluss unter Komplement).

$$\iff L(\mathcal{G}_1) \subseteq L(\mathcal{G}_2') \quad \rightsquigarrow \text{Inklusionsproblem} \qquad (*)$$
$$\iff L(\mathcal{G}_1) \cup L(\mathcal{G}_2') = L(\mathcal{G}_2')$$

Wegen Abschluss unter $\cup$: $\exists\, \mathcal{G}_3 \in$ CFG mit $L(\mathcal{G}_3) = L(\mathcal{G}_1) \cup L(\mathcal{G}_2')$

$$\iff L(\mathcal{G}_3) = L(\mathcal{G}_2')$$
$$\iff (\mathcal{G}_3, \mathcal{G}_2') \in A$$

$\curvearrowright$ Äquivalenzproblem ist unentscheidbar.
$(*) \rightarrow$ (Inklusionsproblem ist ebenfalls unentscheidbar.) □

Satz 6.28: Das Leerheitsproblem für Typ 1 Sprachen ist unentscheidbar.

BEWEIS: Reduktion auf Schnittproblem für CFL.
Sei $(\mathcal{G}_1, \mathcal{G}_2) \in$ Schnittproblem (Typ 1).
Insbesondere $\mathcal{G}_1, \mathcal{G}_2$ Typ 1 Grammatiken.
Typ 1 Sprachen sind unter $\cap$ abgeschlossen, also $\exists\, \mathcal{G}$ Typ 1 Gramatik mit $L(\mathcal{G}) = L(\mathcal{G}_1) \cap L(\mathcal{G}_2)$
Also „$L(\mathcal{G}) = \varnothing$“ unentscheidbar. □

7 Komplexitätstheorie

7.1 Komplexitätsklassen und P/NP

Def. 7.1: Sei $f : \mathbb{N} \to \mathbb{N}$ eine Funktion.
Die Klasse NTIME($f(n)$) besteht aus allen Sprachen, die von einer (Mehrkanal-)TM M in $T_M(w) \leq f(|w|)$ akzeptiert werden.
Dabei $T_M(w) = \begin{cases} \text{Anzahl der Schritte einer kürzesten akzeptierenden Berechnung von } M \text{ auf } w \\ 1 \quad \text{falls } \nexists \end{cases}$

$\oplus$

Def. 7.2: Ein Polynom ist eine Funktion $p : \mathbb{N} \to \mathbb{N}$ mit $\exists k \in \mathbb{N}\ a_0, \ldots, a_k \in \mathbb{N}$ und $p(n) = \sum_i^k a_i n^k$ $\oplus$

Def. 7.3: Die Klasse NP besteht aus allen Sprachen, die von NTM in polynomieller Zeit akzeptiert werden können.

$$NP = \cup_{p \text{ Polynom}} \text{NTIME}(p(n))$$

$\oplus$

Analog für deterministische TM:

Def. 7.4: Sei $f : \mathbb{N} \to \mathbb{N}$ Funktion
DTIME($f(n)$) = Klasse der Sprachen, die von DTM in $T_M(w) \leq f(|w|)$ Schritten akzeptiert wird.

$$P = \cup_{p \text{ Polynom}} DTIME(P(n)) \qquad \oplus$$

Offenbar $P \leq NP$. Seit 1970 weiß man nicht, ob $P = NP$ oder $P \neq NP$

Praktische Relevanz: Es existieren wichtige Probleme, die offensichtlich in NP liegen, aber die besten bekannten Algorithmen sind exponentiell.
Beispiel: Traveling Salesman ($O(2^n)$), Erfüllbarkeit der Aussagenlogik.

Struktur: Viele der NP-Probleme haben sich als gleichwertig erwiesen, in dem Sinn, dass eine P-Lösung für alle anderen liefert.
$\rightsquigarrow$ NP-Vollständigkeit.

Def. 7.5: Seien $A, B \subseteq \Sigma^*$ Sprachen. A ist polynominell reduzierbar auf B, $A \preceq_p B$, falls $\exists$ totale berechenbare Funktion $f : \Sigma^* \to \Sigma^*$, deren Konfigurationszeit durch ein Polynom beschränkt ist und $w \in A \iff f(w) \in B \quad \forall w \in \Sigma^*$ $\oplus$

Lemma 7.1: Falls $A \preceq_p B$ und $B \in P$ (NP) dann auch $A \in P$ (NP).

BEWEIS: $B \in P : \exists M$, die B in $p(n)$ Schritten akzeptiert.
$\exists M_f$, die die Reduktion $A \preceq_p B$ implementiert. Die Laufzeit von M_f sei durch q Polynom beschränkt.
Betrachte $M' =$"erst M_f, dann M auf dem Ergebnis" M' akzeptiert A.
$w \in A$ $M_f(w)$ liefert $f(w)$ in $\subseteq q(|w|)$ Schritten ohne $|f(m)| \subseteq q(|w|)$
$M(f(w))$ benötigt $\leq p(|f(w)|) \leq p(q(|w|))$ Schritte zum akzeptieren.
$\curvearrowright A \in \text{DTIME}(q(w) + p(q(w)) \subseteq P$ □

Lemma 7.2: $\preceq_p$ ist reflexiv und transitiv.

BEWEIS: Identität; ähnlich wie Beweis von Lemma 7.1. □

Entwurf für Vorlesung: 8.2.17

Def. 7.6:

- Eine Sprache A heißt NP-hart (NP-schwer), falls $\forall L \in NP : L \preceq_p A$.
- Eine Sprache A heißt NP-vollständig, wenn A NP-hart und $A \in NP$. ⊕

Bem.: Sobald eine NP-hartes Problem A bekannt ist, reicht es $A \preceq_p B$ zu finden, um zu teigen, dass B ebenfalls NP-hart ist.

Satz 7.3: Sei A NP-vollständig.

$$A \in P \iff P = NP$$

BEWEIS:
„$\Longleftarrow$" trivial.
„$\Longrightarrow$" $A \in P \subseteq NP \quad \forall L \in NP : L \preceq_p A$. Nach Lemma 7.1 : $L \in P$ □

Ein erstes NP-vollständiges Problem.

Def. 7.7: SAT, das Erfüllbarkeitsproblem der Aussagenlogik (AL) ist definiert durch

Eingaben: Formal F der Aussagenlogik.

Frage: Ist F erfüllbar, d.h. existiert eine Belegung β der Variablen mit $\{0, 1\}$, so dass $F[\beta] = 1$ ist.

$SAT = \{\text{code}(F) \mid F\text{ist erfüllbare Formel der AL}\}$ ⊕

Satz 7.4 (Cook)**:** SAT ist NP-vollständig.

BEWEIS:

1. $SAT \in NP$
   Rate nicht deterministisch eine Belegung β
   Werte $F[\beta]$ aus
   $\curvearrowright$ in NTIME(n), polynomiell
2. SAT ist NP-hart.
   Zeige: $\forall L \in NP : L \preceq_p SAT$
   $L \in NP : \exists p$ Polynom, NTM M mit $L = L(M)$ mit Zeitschranke $T_M(w) \leq p(|w|)$.

   Sei $w = x_1 \dots x_n \in \Sigma^*$ Eingabe für M.
   Definiere F, so dass F erfüllbar $\iff$ M akzeptiert w

   Sei $Q = Q(M)$ mit $\{q_1, \dots, q_k\} = Q$
   Sei $\Gamma = \Gamma(M)$ mit $\{a_i, \dots, a_l\} = \Gamma$
   Definiere folgende Variablen zur Ver. in F

   - state$(t, q) = 1$, genau dann wenn M nach T Schritten im Zustand q
   - pos$(t, i) = 1$, gdw. der Kopf von M steht nach t Schritten auf Position i.
     $t \in \{0, \dots P(n)\}$
     $i \in \{-p(n), \dots, 0, 1, \dots, p(n)\}$
   - tape$(t, i, a) = 1$, gdw. nach t Schritten befindet sich a an Position i auf dem Band.
     $t \in \{0, \dots, p(n)\}$
     $i \in \{-p(n), \dots, p(n)\}$
     $a \in \Gamma$ □

Lemma 7.5: Für jedes $k \in \mathbb{N}$ existiert eine Formel G, sodass $G(x_i, \dots, x_k) = 1$ gdw. $\exists j : x_j = 1$ und $\forall i \neq j : x_i = 0$. Es gilt $|G| \in O(k^2)$.

Beweis:

$$G(x_i, \dots, x_k) = \bigvee_{i=1}^{k} x_i \wedge \bigwedge_{i \neg j} \neg(x_i \wedge x_j)$$

$M = (Q, \Sigma, \Gamma, \delta, q_0, \sqcup, F)$ erkennt L in NTIME(p), p Polynom.

Ziel: Konstruiere aus M, w eine Formel F, so dass

F erfüllbar $\leftrightarrow$ M akzeptiert w

state(t, q) $t \in 0, \dots, p(n), q \in Q$

$\Leftrightarrow$ nach t Schritten ist M in Zeile q

pos(t, i) $\Leftrightarrow$ nach t Schritten ist Kopf amn Pos i , $-p(n) \leq i \leq p(n)$

tape(t, i, a) $\Leftrightarrow$ nach t Schritten enthält Band$[i] = q \in \Gamma$

$F = R \wedge A \wedge T_1$

1. Randbedingungen

$$\begin{aligned} R = & \bigwedge_t G(\text{state}(t, q1), \ldots, \text{state}(t, q_k)) \\ & \wedge \bigwedge_t G(\text{pos}(t, -p(n)), \ldots, \text{pos}(t, D), \ldots, \text{pos}(t, p(n))) \\ & \wedge \bigwedge_{t,i} G(\text{tape}(t, i, a_1), \ldots, \text{tape}(t, i, a_l)) \end{aligned}$$

2. Anfangskonfiguration

$$\begin{aligned} A = & \quad \text{state}(0, q_1) \wedge \text{pos}(0, 1) \\ & \wedge \text{tape}(0, 1, x_1) \wedge \cdots \wedge \text{tape}(0, n, x_n) \\ & \wedge \bigwedge_{-p(n) \leq i \leq p(n)} \text{tape}(0, i, \sqcup) \end{aligned}$$

3. Transitionsschritte

$$\begin{aligned} T_1 = & \bigwedge_{\substack{t \in 0, \ldots, p(n)-1, \\ i,n}} \text{state}(t, q) \wedge \text{pos}(t, i) \wedge \text{tape}(t, i, a) \\ & \rightarrow \text{state}(z+1, q') \wedge \text{pos}(t+1, i+d) \wedge \text{tape}(t+1, i, a') \\ & \delta(q, a) \ni (q', a', d) \\ & d \in \{-1, a, 1\} \\ T_2 = & \bigwedge_{\substack{t,i,q \\ t < p(n)}} \neg \text{pos}(t, i) \wedge \text{tape}(t, i, a) \rightarrow \text{tape}(t+1, i, a) \end{aligned}$$

4. Endkonfiguration

$$E = \bigvee_{q \in F} \text{state}(p(n), q)$$

$|F|$ ist polznomiell beschränkt in $|M, w|$, also $L \preceq_p$ SAT
$\curvearrowright$ SAT ist NP-vollständig.

□

7.2 Weitere *NP*-vollständige Probleme

Def. 7.8: $3SAT$ ist das Erfüllbarkeitsproblem der AL für Formeln in CNF mit höchstens drei Literalen pro Klausel ⊕

Satz 7.6: $3SAT$ ist NP-vollständig.

Beweis: $3SAT \in NP$ offensichtlich.
Reduktion $SAT \preceq_p 3SAT$. Sei F eine Formel der AL.
Definiere ϕ : Formel $\to \{0,1\}^* \to$ Formel, sodass $F' = \phi(F, \epsilon)$ erfüllbar ist, gdw. F erfüllbar.

$$\begin{aligned}
\phi(\text{true}, \pi) &= [y_\pi] \qquad , y_\pi \text{ neue Variable, nicht aus } F \\
\phi(\text{false}, \pi) &= [\overline{y_\pi}] \\
\phi(x_i, \pi) &= [y_\pi \leftrightarrow x_i] \\
\phi(F_0 \wedge F_1, \pi) &= \phi(F_0, \pi_0) \wedge \phi(F_1, \pi_1) \wedge [y_\pi \leftrightarrow y_{\pi_0} \wedge y_{\pi_1}] \\
\phi(F_0 \vee F_1, \pi) &= \phi(F_0, \pi_0) \wedge \phi(F_1, \pi_1) \wedge [y_\pi \leftrightarrow y_{\pi_0} \vee y_{\pi_1}] \\
\phi(F, \pi) &\text{ ist min. um einen linearen Faktor größer als } F \\
&\quad \text{ist Konj. von eingeklammerten Termen } [\dots]
\end{aligned}$$

- $y_\pi \leftrightarrow x_i = (\overline{y_\pi} \vee x_i) \wedge (y_\pi \vee \overline{x_i})$
- $y_\pi \leftrightarrow y_{\pi_0} \wedge y_{\pi_1} = (\overline{y_\pi} \vee y_{\pi_0} y_{\pi_1}) \wedge (y_\pi \vee \overline{y_{\pi_0} y_{\pi_1}}) = (\overline{y_\pi} \vee y_{\pi_0}) \wedge (\overline{y_\pi} \vee y_{\pi_1})$
- $y_\pi \leftrightarrow y_{\pi_0} \vee y_{\pi_1} = (y_\pi \vee \overline{y_{\pi_0}}) \wedge (y_\pi \vee \overline{y_{\pi_1}}) \wedge (\overline{y_\pi} \vee y_{\pi_0} \vee y_{\pi_1}) \wedge (y_\pi \vee \overline{y_{\pi_0}} \vee \overline{y_{\pi_1}})$

□

Def. 7.9 (CLIQUE)**:** Sei $\mathcal{G} = (V, E)$ ein ungerichteter Graph und $k \in \mathbb{N}$.
$(\mathcal{G}, k) \in$ CLIQUE, falls $\exists$ Clique der Größe k in $\mathcal{G}$.
Eine Clique $C \subseteq V$, so dass $\forall u \neq v \in C : \{u, v \in E\}$ ⊕

Satz 7.7: CLIQUE ist NP-vollständig.

Beweis: Durch Reduktion: $3SAT \preceq_p$ CLIQUE
Sei F eine Formel in 3CNF, erweitert, so dass jede Klausel 3 Literale

$$\begin{aligned}
(x \vee y) &\rightsquigarrow (x \vee y \vee x) \\
x &\rightsquigarrow (x \vee x \vee x)
\end{aligned}$$

Jetzt $F = \bigwedge_{i=1}^{m} (z_{i,1} \vee z_{i,2} \vee z_{i,3}) \qquad z_{i,j} \in \{x_i, \dots, x_n\} \cup \{\overline{x}_1, \dots, \overline{x}_n\}$

Definiere $\mathcal{G} = (V, E)$ und k wie folgt:

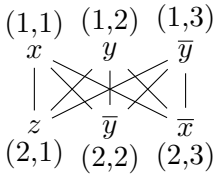
$$\begin{aligned}
V &= \{(i,j) \mid 1 \leq i \leq m, j \in \{1,2,3\}\} \\
E &= \{\{(i,j),(p,q)\} \mid i \neq p, z_{i,j} \neq \neg z_{p,q} \\
k &= m
\end{aligned}$$

F ist erfüllbar.
$\iff$ in jeder Klausel i muss mindestens ein Literal $= 1$ sein, unter Bedingung β.
$\iff \exists$ Folge $z_{1,j_2}, \ldots, z_{m,j_m}$ mit $z_{i,j_2}[\beta] = 1$
$\iff \exists$ Folge $z_{1,j_1}, \ldots, z_{m,j_m}$, sodass $\forall i \neq p : z_{i,j_i} \neq \neg z_{p,j_p}$
$\iff \exists$ Menge von Knoten $\{(1, j_1), \ldots, (m, j_m)\}$ die paarweise durch Kanten verbunden sind.
$\iff \exists$ Clique der Größe $k = m$ in $\mathcal{G}$ □

Bsp.:

$$F = (\underbrace{x \vee y \vee \overline{y}}_{1}) \wedge (\underbrace{z \vee \overline{y} \vee \overline{x}}_{2})$$

$\mathcal{G}$:



RL: Grafik überprüfen

Liste der Definitionen

Liste der Sätze

Abbildungsverzeichnis

Abkürzungsverzeichnis

AL Aussagenlogik
CFL Menge der kontextfreien Sprachen
CFG Menge der kontextfreien Grammatiken
CNF Chomsky Normalform
CP Korrespondenzproblem
CYK Cocke, Younger, Kasami
DAG gerichteter azyklischer Graph
DCFG deterministische CFG
DCFL deterministische CFL
DEA deterministischer endlicher Automat
DFA engl.: deterministic finite automaton
DPDA deterministischer Kellerautomat
DTM deterministische TM
EA endlicher Automat
LBA Linear Bounded Automaton
MPCP Das modifizierte PCP
ND Nicht-Determinismus
NEA nichtdeterministischer endlicher Automat
NFA engl.: nondeterministic finite automaton
NPDA nichtdeterministischer Kellerautomat
NT Nichtterminal
NTM Eine nichtdeterministische TM
PCP Das Postsche Korrespondenzproblem
PDA pushdown automaton (Kellerautomat)
PL Pumping Lemma
RE Menge der regulären Ausdrücke
REG Menge der regulären Sprachen
RM Registermaschine
TM Turing-Maschine
TT Turingtabelle

Anmerkungsverzeichnis