Einführung in die Theoretische Informatik
Zusammenfassung

Ali, Mihir, Noah

June 2, 2022

Contents

1 Formale Sprachen

1.1 Grundbegriffe

- Alphabet Σ (endliche Menge) z.B. $\{1, 0\}$
- Wort/String über Σ ist eine endliche Folge von Zeichen aus Σ
- $|w|$ länge des Wortes w
- Leeres Wort ϵ
- uv konkatenation der Wörter u und w
- Ist w ein Wort so ist $w^0 = \epsilon$ und $w^{n+1} = ww^n$
- Σ^* Menge aller Wörter über Σ
- (formale) Sprache $L \subseteq \Sigma^*$

1.1.1 Operationen auf Sprachen

Seien $A, B \subseteq \Sigma^*$

- Konkatenation:

$AB = \{uv | u \in A \wedge v \in B\}$

- Konkatenation mit sich selbst:

$A^n = \{w_1...w_n | w_1, ..., w_n \in A\} = A...A$

- $A^* = \{w_1...w_n | n \geq 0 \wedge w_1, ..., w_n \in A\} = \bigcup_{n \in \mathbb{N}} A^n$
- $A^+ = AA^* = \bigcup_{n \geq 1} A^n$

1. Sonderfälle:
   - $\forall A : \epsilon \in A^*$
   - $\emptyset^* = \{\epsilon\}$
   - $\emptyset A = \emptyset$
   - $\{\epsilon\} A = A$
   - $A^* A^* = A^* = (A^*)^*$

1.1.2 Grammatiken

4-Tupel $G = (V, \Sigma, P, S)$

- V ist endliche Menge von Nichtterminalzeichen
- Σ ist endliche Menga von Terminalzeichen (= Alphabet)
- $P \subseteq (V \cup \Sigma)^* \times (V \cup \Sigma)^*$ ist Menge von Produktionen
- $S \in V$ ist das Startsymbol

Die Sprache von G ist die Menge aller Wörter, die von G erzeugt werden. Sie wird mit $L(G)$ bezeichnet. Also jedes Wort, dass die Grammatik erzeugt muss in der Sprache erhalten sein und jedes Wort in der Sprache muss von der Grammatik erzeugt werden.

1. Reflexve transitive Hülle

   - $\alpha \rightarrow_G^0 \alpha$
   - $\alpha \rightarrow_G^{n+1} \gamma : \exists \beta . \alpha \rightarrow_G^n \rightarrow_G \gamma$
   - $\alpha \rightarrow_G^* \beta : \exists n . \alpha \rightarrow_G^n \beta$
   - $\alpha \rightarrow_G^+ \beta : \exists n > 0 . \alpha \rightarrow_G^n \beta$

1.1.3 Chomsky Hierarchie

Eine Grammatik G ist vom

- Typ 0 immer
- Typ 1 falls fpr jede Produktion $\alpha \rightarrow \beta$ außer $S \rightarrow \epsilon$ gilt $|\alpha| \leq |\beta|$
- Typ 2 Falls G vom typ 1 ist und für jede Produktion $\alpha\beta$ gilt $\alpha \in V$
- Typ 3 falls G vom Typ 2 ist und für jede Produktion $\alpha \rightarrow \beta$ außer $S \rightarrow \epsilon$ gilt $\beta \in \Sigma \cup \Sigma V$

1. Grmmatiken und Sprachklassen:

Typ 3	Rechtslineare Grammatiken	Reguläre Sprachen
Typ 2	Kontextfreie Grammatik	Kontextfreie Sprachen
Typ 1	Kontextsensitive Grammatik	Kontextsens. Sprachen
Typ 0	Phrasenstrukturgrammatik	Rekursiv aufzählbare Sprachen

2. Satz 2.13 $L(Typ3) \subset L(Typ2) \subset L(Typ1) \subset L(Typ0)$

1.1.4 Wortproblem

Gegeben: eine Grammatik G, ein Wort $w \in \Sigma^*$ Frage: Ist das Wort in w enthalten ($w \in L(G)$)?

2 Reguläre Sprachen

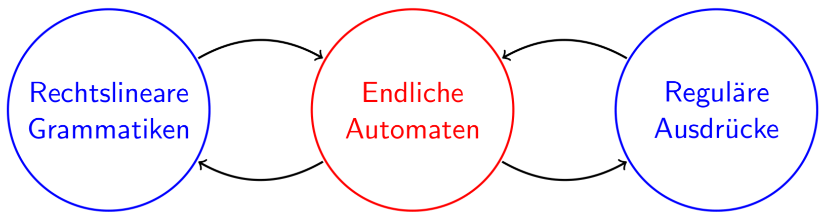

Figure 1: Reguläre Sprachen Schema

2.1 Deterministische endliche Automaten

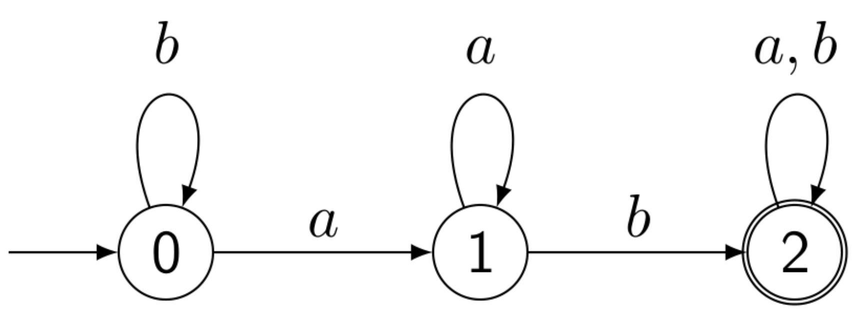

Figure 2: Beispiel Automat

- Beispiel:
  - Eingabewort *baba* $\rightarrow$ Zustandsfolge 0,0,1,2,2
- "Bei dieser Grammatik muss mindestens nach einem a ein b kommen"
- Die Sprache des DFA ist die Menge aller Wörter über $\{a, b\}$, die ab enthalten

Erkannte Sprache: Menge der Wörter, die vom Startzustand in einen Endzustand führen. Recognizer, die nur einmal das Wort durchläuft und in linearer Zeit es akzeptiert oder ablehnt.

2.1.1 Definition

Ein deterministischer endlicher Automat $M = (Q, \Sigma, \delta, q_0, F)$ besteht aus

- endliche Menge von Zuständen Q
- endlichem Eingabealphabet Σ
- einer totalen Übergangsfunktion $\delta : Q \times \Sigma \to Q$
- eienm Startzustand $q_0 \in Q$
- einer Menge $F \subset Q$ von Endzuständen

1. Akzeptierte Sprachen (Definition 3.2) Von M akzeptierte Sprache $L(M) := \{w \in \Sigma^* | \hat{\delta}(q_0, w) \in F\}$ wobei
   $\hat{\delta} : Q \times \Sigma^* \to Q$ induktiv definiert ist:
   $\hat{\delta}(q, \epsilon) = q$
   $\hat{\delta}(q, aw) = \hat{\delta}(\delta(q, a), w)$, für $a \in \Sigma, w \in \Sigma^*$
   ($\hat{\delta}(q, w)$ bezeichnet den Zustand, den man aus q mit w erreicht.)
   Eine Sprache ist regulär **gdw** sie von einem DFA akzeptiert wird.
2. Beispiel Automat der Sprache akzeptiert Induktiv beweisen pro Zustand.

2.2 Von rechtslinearen Grammatiken zu DFA

- Für jede rechtslineare Grammatik G gibt es einen DFA M mit $L(M) = L(G)$
- Für jeden DFA M gibt es eine rechtslineare Grammatik G mit $L(G) = L(M)$

2.2.1 Nichtdeterministischer endlicher Automat

Ein deterministischer endlicher Automat $M = (Q, \Sigma, \delta, q_0, F)$ besteht aus

- Q, Σ, q_0, F sind wie DFA

- $\delta : Q \times \Sigma \to P(Q)$
  $P(Q)$ = Menge aller Teilmengen von $Q = 2^Q$
  Alternative: Relation $\delta \subseteq Q \times \Sigma \times Q$

$\bar{\delta}(S, a) := \bigcup_{q \in S} \delta(q, a)$
Es folgt: $\hat{\bar{\delta}} : P(Q) \times \Sigma^* \to P(Q)$

1. Intuition: $\hat{\bar{\delta}}(S, w)$ ist Menge aller Zustände, die sich von einem Zustand in S aus w erreichen lassen.

2. Von nichtdeterminitsichen Automaten N akzeptierte Sprache $L(N) := \{w \in \Sigma^* | \hat{\bar{\delta}}(\{q_0\}, w) \cap F \neq \emptyset\}$

2.2.2 Satz 3.9

Für jede rechtslineare Grammatik G gibt es einen NFA M mit $L(G) = L(M)$

2.2.3 Satz 3.13

Für jeden DFA M gibt es eine rechtslineare Grammatik G mit $L(M) = L(G)$

2.3 3.3 NFAs mit ϵ -Übergängen

Grammatiken von Programmiersprachen enthalten viele Produktionen der Gestalt $A \to B$.

Ein NFA mit ϵ-Übergängen (auch ϵ -NFA) ist ein NFA mit einem speziellen Symbol $\epsilon \notin \Sigma$ und mit $\delta : Q \times (\Sigma \cup \epsilon) \to P(Q)$. Ein ϵ übergang darf ausgef¨uhrt werden, ohne dass ein

2.3.1 Lemma 3.16

Für jeden ϵ -NFA N gibt es einen NFA N' mit $L(N) = L(N')$.

2.4 3.4 Regex

- $\emptyset$ ist ein regex
- ϵ ist ein regex
- Für jedes $a \in \Sigma$ ist a ein regulärer Audruck
- Wenn α und β regex dann auch

1. $\alpha\beta$
2. $\alpha|\beta$
3. α^*

- Sonst NIX!

2.4.1 Definition 3.20

Zu einem regulären Ausdruck γ ist die zugehörige Sprache $L(\gamma)$ rekursiv definiert:

- $L(\emptyset) = \emptyset$
- $L(\epsilon) = \{\epsilon\}$
- $L(a) = a$
- $L(\alpha\beta) = L(\alpha)L(\beta))$
- $L(\alpha|\beta) = L(\alpha) \cup L(\beta)$
- $L(\alpha^*) = L(\alpha)^*$

2.4.2 Satz 3.23 (Kleene 1956)

Eine Sprache $L \subseteq \Sigma^*$ ist genau dann durch einen regulären Ausdruck darstellbar, wenn sie regulär ist.
$R_{ij}^{k+1} = R_{ij}^k \cup R_{i(k+1)}^k (R_{(k+1)(k+1)}^k)^* R_{(k+1)j}^k$ *in regex* $\cup = |$
R_{ij}^{k+1} = alle Wörter die in R_{ij}^k sind plus alle Wörter die mindestens einmal q_{k+1} besuchen Somit gilt $L(M) = L(\alpha_{1i_1}^n|...|\alpha_{1i_r}^n)$, wobei $F = \{i_1, ..., i_r\}$

2.4.3 Wie teuer sind unsere Konversionen?

$$\begin{tikzcd}
\mathsf{DFA} \arrow[dr] & \mathsf{NFA} \arrow[l] \arrow[d] & \epsilon\text{-}\mathsf{NFA} \arrow[l] \\
& \mathsf{RE} \arrow[ur] &
\end{tikzcd}$$

- RE $\rightarrow \epsilon$ -NFA: RE der Länge n, $O(n)$ Zustände

- ϵ -NFA $\rightarrow$ NFA: Q
- NFA $\rightarrow$ DFA: $O(2^n)$
- FA $\rightarrow$ RE: $O(n4^n)$

2.5 Abschlusseigenschaften regulärer Sprachen

2.5.1 Satz 3.24

Seien $R, R_1, R_2 \subseteq \Sigma^*$ reguläre Sprachen. Dann sind auch

- $R_1 R_2$
- $R_1 \cup R_2$
- R^*
- $\bar{R}(:= \Sigma^* \backslash R)$
- $R_1 \cup R_2$
- $R_1 \backslash R_2$

reguläre Sprachen

1. Produkt-Konstruktion Für den Schnitt ist die De-Morgan regel zu teuer also kann man auch eine Produkt Konstruktion ohne Umweg über De-Morgen benutzen.

   Das funktioniert über Parallelismus also beide DFAs laufen synchron parallel (kreuzprodukt der Zustandsräume).

2.5.2 Satz 3.24 Abschlusseigenschaften regulärer Sprachen

Seien $R.R_1, R_2 \subseteq \Sigma^*$ reguläre Sprachen. Dann sind auch $R_1 R_2, R_1 \cup R_2, R^k, \bar{R}(:= \Sigma^* \backslash R), R_1 \cap R_2, R_1 \backslash R_2$ auch reguläre Sprachen

2.5.3 Satz 3.25

Sind $M_1 = (Q_1, \Sigma, \delta_1, s_1, F_1)$ und $M_2 = (Q_2, \Sigma, \delta_2, s_s, F_2)$ DFAs, dann ist der **Produkt-Automat**

$$M := (Q_1 \times Q_2, \Sigma, \delta, (s_1, s_2), F_1 \times F_2)$$
$$\delta((q_1, q_2), a) := (\delta_1(q_1, a), \delta_2(q_2, a))$$

2.6 Rechnen mit Regulären Ausdrücken

2.6.1 Definition 3.26

Zwei reguläre Ausdrücke sind **äquivalent gdw** sie die gleiche Sprache darstellen:
$\alpha \equiv \beta :\Leftrightarrow L(\alpha) = L(\beta)$

(by the way $\equiv$ steht für Bedeutungsäquivalenz und $=$ für syntaktische gleichheit)

2.6.2 Lemma 3.27

- $\emptyset|\alpha \equiv \alpha|\emptyset \equiv \alpha$
- $\emptyset\alpha \equiv \alpha\emptyset \equiv \emptyset$
- $\epsilon\alpha \equiv \alpha\epsilon \equiv \alpha$
- $\emptyset^* \equiv \epsilon$
- $\epsilon^* \equiv \epsilon$

2.6.3 Lemma 2.8

- Assozitiviät
- Kommutativität
- Distributivität
  - $\alpha(\beta|\gamma) \equiv \alpha\beta|\gamma$
  - $(\alpha|\beta)\gamma \equiv \alpha\gamma|\beta\gamma$
- Idempotenz: $\alpha|\alpha \equiv \alpha$

2.7 Pumping Lemma

Wie zeigt man, dass eine Sprache nicht regulär ist?

2.7.1 Satz 3.32 (Pumping Lemma für Reguläre Sprachen)

Sei $R \subseteq \Sigma^*$ regulär. Dann gibt es ein $n > 0$, so dass sich jedes $z \in R$ mit $|z| \geq n$ so in $z = uvw$ zerlegen lässt, dass

- $v \neq \epsilon$,

- $|uv| \leq n$
- $\forall i \geq 0 . uv^i w \in R.$

Es gibt nicht-reguläre Sprachen, für die das Pumping-Lemma gilt! $\Rightarrow$ Pumping-Lemma hinreichend aber nicht notwendig um Nicht-Regularität zu zeigen.

regulär $\subset$ Pumping-Lemma gilt $\subset$ alle Sprachen

2.8 Entscheidungsverfahren

Eingabe: Ein oder mehrere Objekte, die Reguläre Sprachen beschreiben (DFA, NFA, RE Typ3 Gram, ...) **Frage:** Haben die Sprachen die Eigenschaft X? Ein (Entscheidungs-)Problem ist entscheidbar, wenn es einen Algorithmus gibt, der bei jeder Eingabe in endlicher Zeit die richtige Antwort auf die Frage feststellt.

Welche Entscheidungsprobleme sind für rechtslineare Grammatiken entscheidbar und wie hängt die Laufzeit mit der Beschreibung zusammen.

2.8.1 Definition 3.37

Sei D ein DFA, NFA, RE, rechtslineare Grammatik ...

- **Wortproblem**: Gegeben w und D: gilt $w \in L(D)$
- **Leerheitsproblem**: Gegeben D: gilt $\emptyset = L(D)$
- **Endlichkeitsproblem**: Gegeben D: isz $L(D)$ endlich
- **Äquivalenzproblem**: Gegeben D_1, D_2, gilt $L(D_1) = L(D_2)$

2.9 Automaten und Gleichungssysteme

Wir werden jetzt aus einem Automat ein Gleichungssystem machen um daraus einen RE zu machen.

2.9.1 Ardens Lemma (Satz 3.47)

Sind A, B und X Sprachen mit $\epsilon \notin A$, so gilt $X = AX \cup B \Rightarrow X = A^*B$

2.9.2 Korollar 3.48

Sind α, β und X reguläre Ausdrücke mit $\epsilon \notin L(\alpha)$, so gilt $X \equiv \alpha X | \beta \Rightarrow X \equiv \alpha^* \beta$

2.9.3 Algorithmus um RE aus Automat zu machen

1. Wandle FA mit n Zuständen in ein System von n Gleichungen
2. Löse das System durch schrittweise Elimination von Variablen mit Hilfe von Ardens Lemma für REs (Korollar 3.48).
3. Ist k der Startzustand, so beschreibt $\mathrm{X_k}$ die vom Automaten akzeptierte Sprache.

2.10 Minimierung endlicher Automaten

TODO MIA

2.11 Äquivalenztest von DFAs

Zwei Automaten sind genau äquivalent wenn:

1. Gegeben DFAs $M1$ und $M2$, bilde disjunkte Vereiningung.
("Male M1 und M2 nebeneinander.")
2. Berechne Menge der äquivalenten Zustände.
3. L(M1) = L(M2) gdw die beiden Startzustände äquivalent sind

2.12 Äquivalenz von Zuständen

Zwei Zustände sind äquivalent wenn sie selbe Sprache akzeptieren.

2.13 Minimalität des Quotientenautomaten

Die Residualsprache von L bzgl $w \in \Sigma^*$ ist die Menge:
$L^w := \{z \in \Sigma^* | wz \in L\}$
$L' \subseteq \Sigma^*$ ist Residualsprache von L wenn es w gibt mit $L' = L^w$

2.13.1 Definition 3.55 (Äquivalenz von Wörtern bzgl. L)

(Intuition: Zwei Wörter sind äquivalent wenn sie die gleiche Residualsprache haben.)

zwei Wörter sind äquivalent gdw sie zu den gleichen Zuständen führen

2.13.2 Satz 3.56

Sei M ein DFA ohne unerreichbare Zustände. Der Quotientenautomat $M/\equiv$ ist ein minimaler DFA für $L(M)$.

2.14 Definition 3.57 (Kanonischer Minimalautomat)

$M_L := (R_L, \Sigma, \delta_L, L, F_L)$ mit $\delta_L(R, a) := R^a$ und $F_L := R \in RL|\varepsilon \in R$. δ_L ist wohldefiniert und $\hat{\delta}_L(R, w) = R^w$. Jeder Zustand R erkennt die Sprache R und somit $L(M_L) = L$.

2.14.1 Satz 3.58

Jeder minimaler DFA für eine reguläre Sprache L unterscheidet sich vom kanonischen Minimalautomaten $\mathrm{M_L}$ nur durch eine

2.15 Satz 3.59

Eine Sprache $L \subseteq \Sigma^*$ ist genau dann regulär, wenn sie endlich viele Residualsprachen hat.

3 Kontextfreie Sprachen

3.0.1 Syntaxbaum:

Die Blätter des Baums, von links nach rechts gelesen,

3.1 Definition 4.2

Eine kontextfreie Grammatik $G = (V, \Sigma, P, S)$ ist ein 4-Tupel:
V ist eine endlichen Menge, die Nichtterminalzeichen (oder Variablen),
Σ ist ein Alphabet, die Terminalzeichen, disjunkt von V , $P \subseteq V \times (V \cup \Sigma)*$ eine endlichen Menge, die Produktionen, und
$S \in V$ ist das Startsymbol.

3.2 Definition 4.4

Eine kontextfreie Grammatik G = (V, , P, S) induziert eine Ableitungsrelation →G auf W¨ortern ¨uber V : →G gdw es eine Regel A → in P gibt, und W¨orter 1, 2, so dass = 1A2 und = 12 Beispiel: a + T + a →G

3.3 Definition 4.5 (Reflexiv transitive Hülle)

TODO

3.4 Definition 4.6 (Kontextfreie Sprache)

TODO

3.5 Lemma 4.9 (Dekompositionslemma)

$\alpha_1\alpha_2 \rightarrow^n_G \beta$
$\Leftrightarrow$
$\exists\beta_1, \beta_2, n_1, n_2 . \beta = \beta_1\beta_2 \wedge n = n_1 + n_2 \wedge \alpha_i \rightarrow^{n_i}_G \beta_i (i = 1, 2)$

3.6 Definition 4.12 (Balancierte Klammerausdrücke)

3.6.1 Präfix

$u \preceq w \iff \exists v : uv = w$

3.6.2 Anzahl an Vorkommnissen

$\#_a(w) :=$ Anzahl an a's in w Seien $A(w) := \#_{[}(w) \quad B(w) := \#_{]}(w)$ $w \in \{\ [\ ,\ ]\ \}^*$ sei **balanciert** gdw

1. $A(w) = B(w)$
2. $\forall u \preceq w : A(u) \geq B(u)$

3.6.3 4,13

Grammatik $S \rightarrow \epsilon \mid [S] \mid SS$ erzeugt genau die Menge der balancierten Wörter

3.7 4.15 Syntaxbaum

Ein Syntaxbaum für $G = \{V, \Sigma, P, S)\}$ so dass gilt:

- jedes Blatt mit einem Zeichen aus $\Sigma \cup \{\epsilon\}$ beschriftet ist
- jeder innere Knoten mit einem $A \in V$ beschriftet ist, falls Nachfolger: als $X_1, ..., X_n \in V \cup \Sigma \cup \{\epsilon\}$ beschriftet. Dann ist $A \rightarrow X_1, ..., X_n$ eine Produktion in P
- ein Blatt ϵ der einzige Nachfolger seines Vorgänger ist

3.8 4.17 Äquivalente Bedingungen

Für ein CFG & $w \in \Sigma^*$

- $A \to_G^* w$
- $w \in L_G(A)$ (gemäß der induktiven Definition)
- Es gibt ein Syntaxbaum mit Wurzel A dessen **Rand** das Wort w ist

3.9 4.18

Ein CFG heißt mehrdeutig $\iff$ es 2 **verschiedene** Syntaxbäume gibt die mit gleichem Rand
Ein CFL L heißt inhärent mehrdeutig $\iff$ jede CFG G mit $L(G) = L$ mehrdeutig ist

4 Chomsky-Normalform

4.1 4.21

Ein CFG G ist in Chomsky-Normalform $\iff$ alle Produktionen eine der Formen: $\quad A \to a \quad$ oder $\quad A \to BC \quad$ haben

4.2 4.22

Jede CFG G hat eine CFG G' in Chomsky-Normalform mit $L(G') = L(G) \setminus \{\epsilon\}$
Wenn man $\epsilon \in L(G')$ haben will: Füge am Ende $S' \to S, S \to \epsilon$ hinzu und setzte S' als Startsymbol

4.2.1 Bespiel 4.24

$S \to AB, \quad A \to aAA|B, \quad B \to bBS|\epsilon$
$S \to A, \quad A \to \epsilon, \quad B \to bS$
$S \to B, \quad A \to aA, \quad S \to \epsilon, \quad A \to a$
$B \to bB, \quad B \to b$
$A \to B$ ist eine **Kettenproduktion**

4.3 4.25

Aus jeder CFG G kann man ein CFG G' konstruieren was keine Kettenproduktionen enthält sodass gilt $L(G) = L(G')

4.4 Konstruktion einer Chomsky-Normalform

Eingabe: eine kontextfreie Grammatik $G = (V, \Sigma, P, S)$

1. Für jedes $a \in \Sigma$ das in einer Produktion mit Länge ≥ 2 vorkommt:
   - Füge ein neues Nichtterminal X_a zu V & ersetzt alle diese a's dadurch
   - Füge $X_a \rightarrow a$ zu P hinzu
2. Ersetze Produktion in der Form:

   $$A \rightarrow B_1 B_2 ... B_k \quad (k \geq 3)$$

   durch

   $$A \rightarrow B_1 C_2,\ C_2 \rightarrow B_2 C_3,\ ...,\ C_{k-1} \rightarrow B_{k-1} B_k$$

   wobei $C_2,\ ...,\ C_{k-1}$ neue Nichtterminale sind
3. Elimniere alle ϵ -Produktionen
4. Eliminiere alle Kettenproduktionen

4.5 4.27 Greibach-Normalform

Ein CFG ist in Greibach-Normalform falls alle Produktionen in der From sind:
$A \rightarrow aA_1...A_n$

4.6 Pumping Lemma für kontextfreie Sprachen

Für jede kontextfreie Sprache L gibt es ein $n \geq 1$, so dass sich jedes Wort $z \in L$ mit $|z| \geq n$ sich zerlegen lässt in:
$z = uvwxy$
mit

- $vx \neq \epsilon$
- $|vwx| ≤ n
- $\forall i \in \mathbb{N} : uv^i wx^i y \in L$