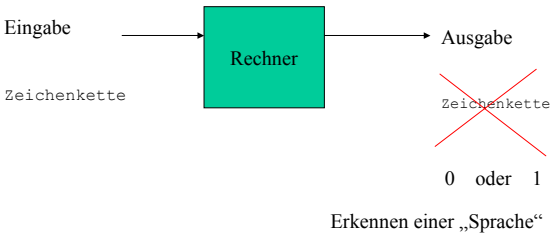
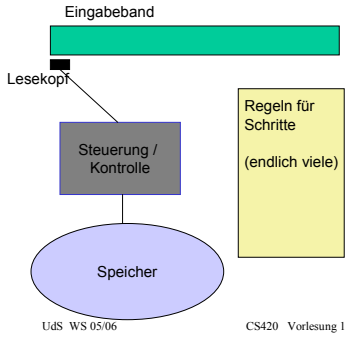
Einfache Rechensituation



Erkennen einer „Sprache“

Skizze einer abstrakten Rechenmaschine



Rechenschritt

Abhängig von
Symbol unter Lesekopf
derzeitigem Zustand
Speicherinhalt

gehe in neuen Zustand
bewege Lesekopf
ändere Speicherinhalt

durch anwendbare Regel vorgegeben

Konfiguration (Momentaufnahme der Maschine)

- Inhalt des Eingabebandes
- Position des Eingabekopfes
- Zustand
- Speicherinhalt

Startkonfiguration für Eingabe x: $start_x$
Endkonfigurationen

Automat akzeptiert Eingabe *x*, wenn er durch eine endliche Anzahl von Schritten aus $start_x$ eine Endkonfiguration erreicht.

Determinismus

Konfiguration (Momentaufnahme der Maschine)

- Inhalt des Eingabebandes
- Position des Eingabekopfes
- Zustand
- Speicherinhalt

Startkonfiguration für Eingabe x: $start_x$
Endkonfigurationen

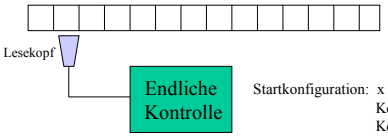
Automat akzeptiert Eingabe *x*, wenn er durch eine endliche Anzahl von Schritten aus $start_x$ eine Endkonfiguration erreichen kann

Nicht - **Determinismus**

3 Arten von Rechenmaschinen

- **kein Speicher** endlicher Automat
- **Speicher ist Band mit Zellen und Schreib/Lesekopf** Turing Maschine
- **Speicher ist Band mit Zellen und Schreib/Lesekopf, der nur an einem Ende des Bandes agiert** Keller Automat

Endlicher Automat



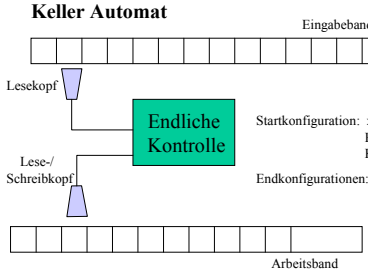
Startkonfiguration: $x \in \Sigma^*$ auf Band,
Kopf ganz links,
Kontrolle in Startzustand

Endkonfigurationen: Kopf ganz rechts
Kontrolle in einem Endzustand

Rechenschrittregeln: (endlich viele)

derzeitiger Zustand | gelesenes Symbol → neuer Zustand | Kopf nach rechts

Keller Automat



Startkonfiguration: $x \in \Sigma^*$ auf E-Band, nichts auf A-Band
E-Kopf ganz links,
Kontrolle in Startzustand

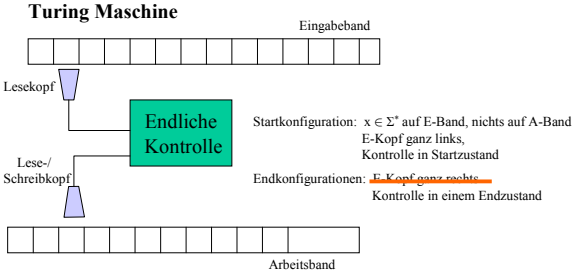
Endkonfigurationen: E-Kopf ganz rechts
Kontrolle in einem Endzustand

Rechenschrittregeln: (endlich viele)

derzeitiger Zustand | gelesenes E-Symbol | gelesenes A-Symbol → neuer Zustand | E-Kopf nach rechts | schreibe A-Symbol | bewege A-Kopf

Wenn A-Kopf nach links bewegt wird, hinterlässt er leere Zelle.

Turing Maschine



Startkonfiguration: $x \in \Sigma^*$ auf E-Band, nichts auf A-Band
E-Kopf ganz links,
Kontrolle in Startzustand

Endkonfigurationen: ~~E-Kopf ganz rechts~~
Kontrolle in einem Endzustand

Rechenschrittregeln: (endlich viele)

derzeitiger Zustand | gelesenes E-Symbol | gelesenes A-Symbol → neuer Zustand | E-Kopf nach rechts | schreibe A-Symbol | bewege A-Kopf

L DEA-Sprache : Es gibt einen deterministischen endlichen Automaten, der L akzeptiert.

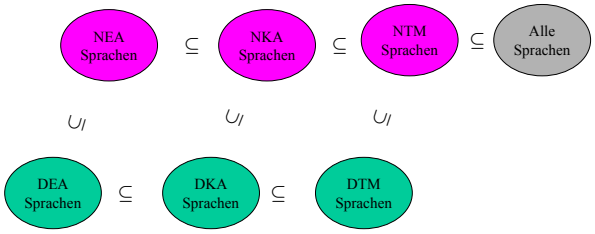
L NEA-Sprache : Es gibt einen nicht-deterministischen endlichen Automaten, der L akzeptiert.

L DKA-Sprache : Es gibt einen deterministischen Keller-Automaten, der L akzeptiert.

L NKA-Sprache : Es gibt einen nicht-deterministischen Keller-Automaten, der L akzeptiert.

L DTM-Sprache : Es gibt eine deterministischen Turing Maschine, die L akzeptiert.

L NTM-Sprache : Es gibt eine nicht-deterministische Turing Maschine, die L akzeptiert.



Funktion $f : A \to B$

f injektiv $\Leftrightarrow \forall a,a' \in A, a \neq a' : f(a) \neq f(a')$

f surjektiv $\Leftrightarrow \forall b \in B \; \exists a \in A : f(a) = b$

f bijektiv $\Leftrightarrow f$ injektiv und f surjektiv

Mengen A und B sind gleichmächtig $\Leftrightarrow \exists$ Bijektion $f : A \to B$

Menge A ist endlich $\Leftrightarrow \exists k \in \mathbb{N} :$ A gleichmächtig mit $\{1,\ldots,k\}$

Menge A ist abzählbar unendlich $\Leftrightarrow$ A gleichmächtig mit $\mathbb{N}$

Menge A ist abzählbar $\Leftrightarrow$ A ist endlich oder
A ist abzählbar unendlich

Funktion $f : A \to B$

f injektiv $\Leftrightarrow \forall\, a,a' \in A,\ a \neq a' : f(a) \neq f(a')$

f surjektiv $\Leftrightarrow \forall\, b \in B\ \exists\, a \in A : f(a) = b$

f bijektiv $\Leftrightarrow$ f injektiv und f surjektiv

Mengen A und B sind gleichmächtig $\Leftrightarrow \exists$ Bijektion $f : A \to B$

Menge A ist endlich $\Leftrightarrow \exists\, k \in \mathbb{N} :$ A gleichmächtig mit $\{1,\ldots,k\}$

Menge A ist abzählbar unendlich $\Leftrightarrow$ A gleichmächtig mit $\mathbb{N}$

Menge A ist abzählbar $\Leftrightarrow$ A ist endlich oder
A ist abzählbar unendlich

Lemma 0:

A abzählbar $\Leftrightarrow$ $\exists$ Injektion $A \to \mathbb{N}$ $\Leftrightarrow$ $\exists$ Surjektion $\mathbb{N} \to A$

B abzählbar und $A \subset B \Rightarrow$ A abzählbar

Lemma 1: $A_1, \ldots, A_k$ endlich $\Rightarrow$ $\bigcup_{1 \le i \le k} A_i$ ist endlich
$A_1 \times \cdots \times A_k$ ist endlich

Lemma 2: A_i endlich für $i \in \mathbb{N}$ (und $A_i \cap A_j = \emptyset$ für $i \neq j$)
$\Rightarrow$ $\bigcup_{i \in \mathbb{N}} A_i$ ist abzählbar

Kor.: Σ endliches Alphabet $\Rightarrow$ Σ^* ist abzählbar

($\Sigma^* = \bigcup_{i \in \mathbb{N}} \Sigma^i$)

Lemma 3: A, B abzählbar $\Rightarrow$ $A \cup B$ abzählbar
$A \times B$ abzählbar

(0,0)	(0,1)	(0,2)	(0,3)	(0,4)	...
(1,0)	(1,1)	(1,2)	(1,3)	(1,4)	...
(2,0)	(2,1)	(2,2)	(2,3)	(2,4)	...
(3,0)	(3,1)	(3,2)	(3,3)	(3,4)	...
(4,0)	(4,1)	(4,2)	(4,3)	(4,4)	...
⋮	⋮	⋮	⋮	⋮	⋱

Kor.: $\mathbb{Q}$ ist abzählbar

Lemma 4: A_i abzählbar für $i \in \mathbb{N}$ $\Rightarrow$ $\bigcup_{i \in \mathbb{N}} A_i$ ist abzählbar

Satz: (Cantor, Russell)

A Menge, $2^A = \{ B \mid B \subset A \}$ Potenzmenge von A

A und 2^A sind NICHT gleichmächtig, d.h.

es gibt keine Bijektion zwischen A und 2^A

Beweis: zu zeigen: jede Funktion $f : A \to 2^A$ ist NICHT surjektiv

d.h. $\exists\, \Delta = \Delta_f$ mit $\Delta \neq f(a)$ für alle $a \in A$

wähle $\Delta = \{ a \in A \mid a \notin f(a) \}$

dann gilt $\Delta \neq f(a)$ für alle $a \in A$, da $a \in \Delta \Leftrightarrow a \notin f(a)$

Korollar 1: $2^{\mathbb{N}}$ ist nicht abzählbar

Korollar 2: Die Menge aller unbeschränkten Bitfolgen
$\{ \alpha \mid \alpha : \mathbb{N} \to \{0,1\} \}$
ist nicht abzählbar.

	0	1	2	3	4	5	6	7	...
α_0	1	1	0	1	0	1	1	0	...
α_1	0	1	1	1	0	0	1	0	...
α_2	0	0	0	1	0	1	1	1	...
α_3	1	0	0	0	0	0	0	0	...
α_4	1	1	0	1	0	1	0	0	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱

Cantorsche Diagonalisierung

Korollar 1: $2^{\mathbb{N}}$ ist nicht abzählbar

Korollar 2: Die Menge aller unbeschränkten Bitfolgen
$\{ \alpha \mid \alpha : \mathbb{N} \to \{0,1\} \}$
ist nicht abzählbar.

	0	1	2	3	4	5	6	7	...
α_0	**1**0	1	0	1	0	1	1	0	...
α_1	0	**1**0	1	1	0	0	1	0	...
α_2	0	0	**0**1	1	0	1	1	1	...
α_3	1	0	0	**0**1	0	0	0	0	...
α_4	1	1	0	1	**0**1	1	0	0	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱

Cantorsche Diagonalisierung

$\delta(i) = 1 - \alpha_i(i)$

Korollar 1: $2^{\mathbb{N}}$ ist nicht abzählbar

Korollar 2: Die Menge aller unbeschränkten Bitfolgen
$\{ \alpha \mid \alpha : \mathbb{N} \to \{0,1\}\}$
ist nicht abzählbar.

	0	1	2	3	4	5	6	7	...
α_0	**1**0	1	0	1	0	1	1	0	...
α_1	0	**1**0	1	1	0	0	1	0	...
α_2	0	0	**0**1	1	0	1	1	1	...
α_3	1	0	0	**0**1	0	0	0	0	...
α_4	1	1	0	1	**0**1	1	0	0	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱

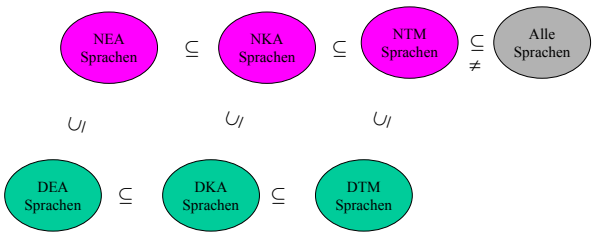
Cantorsche Diagonalisierung

$\delta(i) = 1-\alpha_i(i)$

Korollar 3: $\mathbb{R}$ ist nicht abzählbar

Korollar 4: Es gibt Sprachen, die keine NTM-Sprachen sind.

Beweis: es gibt nur abzählbare viele Turing Maschinen, also
nur abzählbar viele NTM-Sprachen
aber die Menge aller Sprachen ist nicht abzählbar.



Endlicher Automat M

Σ	Eingabealphabet
Q	Zustandsmenge (endlich)
$s \in Q$	Startzustand
$F \subset Q$	Endzustände
$\Delta \subset ((Q \times \Sigma) \times Q)$	Übergangsrelation

M **deterministisch**: $\forall (q,a) \in Q \times \Sigma : |\{q' \in Q : (q,a,q') \in \Delta\}| \leq 1$

Konfigurationen von M: $K_M = Q \times \Sigma^*$

Startkonfiguration für Eingabe $x \in \Sigma^*$: $start_x = (s,x)$

Endkonfigurationen: $Fin = \{(f,\varepsilon) \mid f \in F\}$

Endlicher Automat M

Σ	Eingabealphabet
Q	Zustandsmenge (endlich)
$s \in Q$	Startzustand
$F \subset Q$	Endzustände
$\Delta \subset ((Q \times \Sigma) \times Q)$	Übergangsrelation

M **deterministisch**: $\forall (q,a) \in Q \times \Sigma : |\{q' \in Q : (q,a,q') \in \Delta\}| \leq 1$

Konfigurationen von M: $K_M = Q \times \Sigma^*$

Startkonfiguration für Eingabe $x \in \Sigma^*$: $start_x = (s,x)$

Endkonfigurationen: $Fin = \{(f,\varepsilon) \mid f \in F\}$

Konfigurationen von M: $K_M = Q \times \Sigma^*$

Startkonfiguration für Eingabe $x \in \Sigma^*$: $start_x = (s,x)$

Endkonfigurationen: $Fin = \{(f,\varepsilon) \mid f \in F\}$

Rechenschrittrelation $\vdash_M$ auf K_M

$\forall q \in Q, a \in \Sigma, u \in \Sigma^*$: $(q,au) \vdash_M (q',u)$ g.d.w. $(q,a,q') \in \Delta$

Rechenrelation $\vdash_M^*$ auf K_M

reflexive, transitive Hülle von $\vdash_M$

$k \vdash_M^* k'$ g.d.w. $\exists\, m \geq 0\ \exists\, k_0, \cdots, k_m$ sodass

$k = k_0 \vdash_M k_1 \vdash_M \cdots \vdash_M k_m = k'$

M akzeptiert $x \in \Sigma^*$ g.d.w. $start_x \vdash_M^* (f,\varepsilon)$ für irgendein $f \in F$

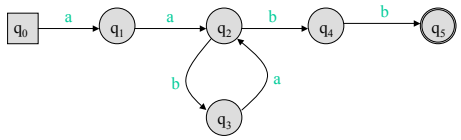
$L(M) = \{ x \in \Sigma^* \mid$ M akzeptiert x $\}$ die von M akzeptierte Sprache.

L DEA-Sprache g.d.w L=L(M) für irgendeinen DEA M

L NEA-Sprache g.d.w L=L(M) für irgendeinen NEA M

$G_M = (Q,E)$ **Übergangsgraph** von M

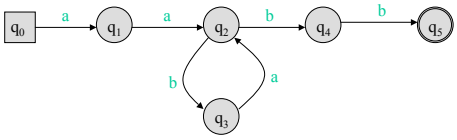
$q \xrightarrow{a} q'$ in E $\Leftrightarrow$ $(q,a,q') \in \Delta$



Startknoten

Endknoten

M akzeptiert Wort w genau dann, wenn w Beschriftung eines gerichteten Pfades in G_M von Startknoten zu einem Endknoten.



aabb $\in$ L(M)

aa ba bb $\in$ L(M)

aa babababa bb $\in$ L(M)

aa bb a $\notin$ L(M)

Fortsetzungssprachen

$L \subset \Sigma^*$ Sprache

$w \in \Sigma^*$: $F_L(w) = \{ x \in \Sigma^* \mid wx \in L \}$

Fortsetzungssprache von w bezüglich L

$\mathcal{F}_L = \{ \mathcal{F}_L(w) \mid w \in \Sigma^* \}$ Menge der Fortsetzungssprachen für L

Fortsetzungssprachen

$L \subset \Sigma^*$ Sprache

$w \in \Sigma^*$: $F_L(w) = \{ x \in \Sigma^* \mid wx \in L \}$

Fortsetzungssprache von w bezüglich L

$\mathcal{F}_L = \{ \mathcal{F}_L(w) \mid w \in \Sigma^* \}$ Menge der Fortsetzungssprachen für L

Bsp: $L = \Sigma^*$

für jedes w gilt: $F_L(w) = \Sigma^*$

also $\mathcal{F}_L = \{ \Sigma^* \}$

$L \subset \Sigma^*$ Sprache

$w \in \Sigma^*$: $F_L(w) = \{ x \in \Sigma^* \mid wx \in L \}$

Fortsetzungssprache von w bezüglich L

$\mathcal{F}_L = \{ \mathcal{F}_L(w) \mid w \in \Sigma^* \}$ Menge der Fortsetzungssprachen für L

Bsp: $\Sigma = \{a,b\}$ $L = \{ u \in \Sigma^* \mid \#_a(u) \text{ ist durch 3 teilbar} \}$

$F_L(bbab) = \{ x \in \Sigma^* \mid \#_a(x) \bmod 3 = 2 \}$

$F_L(ab) = \{ x \in \Sigma^* \mid \#_a(x) \bmod 3 = 2 \}$

$F_L(aab) = \{ x \in \Sigma^* \mid \#_a(x) \bmod 3 = 1 \}$

$F_L(abaab) = \{ x \in \Sigma^* \mid \#_a(x) \bmod 3 = 0 \}$

$\mathcal{F}_L = \{ L_0, L_1, L_2 \}$ $L_i = \{ x \in \Sigma^* \mid \#_a(x) \bmod 3 = i \}$

$L \subset \Sigma^*$ Sprache

$w \in \Sigma^*$: $F_L(w) = \{ x \in \Sigma^* \mid wx \in L \}$

Fortsetzungssprache von w bezüglich L

$\mathcal{F}_L = \{ \mathcal{F}_L(w) \mid w \in \Sigma^* \}$ Menge der Fortsetzungssprachen für L

Bsp: $\Sigma = \{a,b\}$ $L = \{ a^n b^n \mid n \in \mathbb{N} \}$

$F_L(bbab) = \{ \}$ $F_L(ab) = \{ \varepsilon \}$

$F_L(aab) = \{ b \}$ $F\_L(aaaaabb) = \{ bbb \}$

$F_L(aaa) = \{ bbb , abbbb , aabbbbb , \cdots \}$

$\mathcal{F}_L = \{ \{\} \} \cup \{ L_i \mid i \in \mathbb{N} \} \cup \{ S_j \mid j \in \mathbb{N} \}$

$L_i = \{ b^i \mid i \in \mathbb{N} \}$ $S_j = \{ a^{n-j} b^n \mid n \geq j \}$

Lemma: L DEA-Sprache $\Rightarrow$ $\mathcal{F}_L$ ist endlich

Lemma: L DEA-Sprache $\Rightarrow$ $\mathcal{F}_L$ ist endlich

Beweis: L DEA-Sprache $\Rightarrow \exists$ DEA $M=(Q,\Sigma,s,F,\Delta)$ mit $L(M)=L$

für $q \in Q$ sei $L_q = \{x \in \Sigma^* \mid (q,x) \vdash_M^* (f,\varepsilon) \text{ für irgendein } f \in F \}$

$F_L(w) = L_p$ mit p, so dass $(s,w) \vdash_M^* (p,\varepsilon)$

(und $F_L(w) = \{\}$, falls so ein p nicht existiert)

Also gilt $\mathcal{F}_L = \{ L_q \mid q \in Q \}$ (plus möglicherweise $\{\}$)

und $\mathcal{F}_L$ ist endlich.

Lemma: L DEA-Sprache $\Rightarrow$ $\mathcal{F}_L$ ist endlich

d.h. $\mathcal{F}_L$ **nicht** endlich $\Rightarrow$ L **nicht** DEA-Sprache

Um zu zeigen, dass eine Sprache L keine DEA-Sprache ist, reicht es eine unendliche Menge $\{w^{(i)} \in \Sigma^* \}$ von Worten zu finden mit $F_L(w^{(i)}) \neq F_L(w^{(j)})$ für $i \neq j$.

Bsp: $L = \{ a^n b^n \mid n \in \mathbb{N} \}$

Sei $w^{(i)} = a^i b$ für $i > 0$.

$F_L(w^{(i)}) = \{ b^{i-1} \}$

Also $F_L(w^{(i)}) \neq F_L(w^{(j)})$ für $i \neq j$

Um zu zeigen, dass eine Sprache L keine DEA-Sprache ist, reicht es eine unendliche Menge $\{w^{(i)} \in \Sigma^* \}$ von Worten zu finden mit $F_L(w^{(i)}) \neq F_L(w^{(j)})$ für $i \neq j$.

Bsp: $L = \{ a^{n^2} \mid n \in \mathbb{N} \}$

Sei $w^{(i)} = a^{i^2}$ für $i \in \mathbb{N}$.

$i^2 + (2i+1) = (i+1)^2 \quad \Rightarrow \quad w^{(i)} a^{2i+1} = a^{(i+1)^2} \in L$

$w^{(i)} a^{2j+1} \notin L$ für $j<i$

$F_L(w^{(i)}) = \{ a^{2i+1}, \cdots \}$

Also $F_L(w^{(i)}) \neq F_L(w^{(j)})$ für $j<i$

Um zu zeigen, dass eine Sprache L keine DEA-Sprache ist, reicht es eine unendliche Menge $\{w^{(i)} \in \Sigma^* \}$ von Worten zu finden mit $F_L(w^{(i)}) \neq F_L(w^{(j)})$ für $i \neq j$.

Bsp: $L = \{ a^n \mid n$ ist eine Primzahl $\}$

Sei $w^{(p)} = a^p$ mit p Primzahl. $\{ w^{(p)} \mid p$ Primzahl $\}$ ist unendlich.

p,q zwei verschiedene Primzahlen
$\Rightarrow$ p und q teilerfremd
$\Rightarrow \exists$ k mit p+k·q ist Primzahl (aber q+k·q ist keine Primzahl !)

Also $F_L(w^{(p)}) \neq F_L(w^{(q)})$ für $p \neq q$,
denn $a^{k \cdot q} \in F_L(w^{(p)})$ und $a^{k \cdot q} \notin F_L(w^{(q)})$

Fakten über Primzahlen: 1) Es gibt unendlich viele Primzahlen
2) $c,d \in \mathbb{N}$ teilerfremd $\Rightarrow \{ c+k \cdot d \mid k \in \mathbb{N} \}$ enthält eine Primzahl (sogar unendlich viele)
(Satz von Dirichlet)

$L \subset \Sigma^*$ Sprache

$w \in \Sigma^*$: $F_L(w) = \{ x \in \Sigma^* \mid wx \in L \}$

Fortsetzungssprache von w bezüglich L

$\mathcal{F}_L = \{ \mathcal{F}_L(w) \mid w \in \Sigma^* \}$ Menge der Fortsetzungssprachen für L

Satz (Myhill – Nerode):

L DEA-Sprache $\Leftrightarrow$ $\mathcal{F}_L$ ist endlich

Satz (Myhill – Nerode):

L DEA-Sprache $\Leftrightarrow$ $\mathcal{F}_L$ ist endlich

Bew: "$\Rightarrow$"

L DEA-Sprache $\Rightarrow \exists$ DEA $M=(Q,\Sigma,s,F,\Delta)$ mit $L(M)=L$

für $q \in Q$ sei $L_q = \{x \in \Sigma^* \mid (q,x) \vdash_M^* (f,\varepsilon)$ für irgendein $f \in F \}$

$\mathcal{F}_L = \{ L_q \mid q \in Q \}$ (plus möglicherweise $\{\}$) und $\mathcal{F}_L$ ist endlich.

Satz (Myhill – Nerode):

L DEA-Sprache $\Leftrightarrow$ $\mathcal{F}_L$ ist endlich

Bew: "$\Leftarrow$"

$\mathcal{F}_L$ endlich. Definiere NEA $M = (\Sigma,Q,s,F,\Delta)$ mit

$Q = \mathcal{F}_L$

$s = F_L(\varepsilon) = L$

$F = \{ S \in \mathcal{F}_L \mid \varepsilon \in S \}$

$\Delta = \{ (F_L(w),a,F_L(wa)) \mid w \in \Sigma^*, a \in \Sigma \}$

Dann gilt

$(F_L(x_1 \cdots x_k) , x_{k+1} \cdots x_n) \vdash_M (F_L(x_1 \cdots x_{k+1}) , x_{k+2} \cdots x_n)$

und M akzeptiert genau L.

Konsequenz 1: Wenn Sprache L von einem NEA akzeptiert wird, dann wird L auch von einem DEA akzeptiert.

Beweis: L von NEA $M=(\Sigma,Q,s,F,\Delta)$ akzeptiert.

für $q \in Q$ sei $L_q = \{x \in \Sigma^* \mid (q,x) \vdash_M^* (f,\varepsilon)$ für irgendein $f \in F \}$

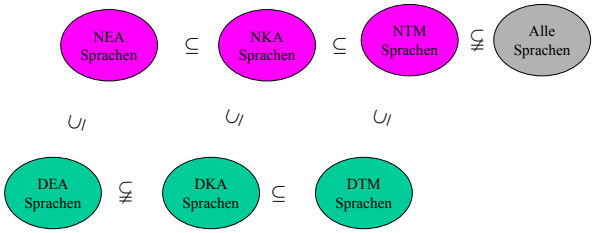
für $w \in \Sigma^*$ sei $Q(w) = \{ q \in Q \mid (s,w) \vdash_M^* (q,\varepsilon) \}$

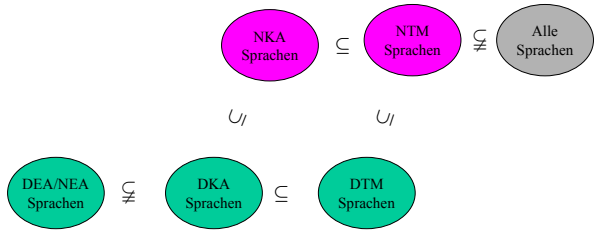
Dann gilt

$F_L(w) = \bigcup_{q \in Q(w)} L_q$

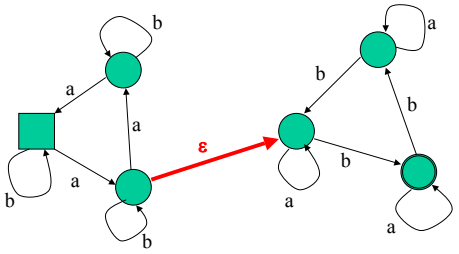
Damit ist $F_L(w)$ durch $Q(w) \subset Q$ eindeutig bestimmt. Aber Q hat nur endlich viele Teilmengen, also gibt es nur endlich viele verschiedene $F_L(w)$'s.

Also ist $\mathcal{F}_L$ endlich, und L wird daher von einem DEA akzeptiert.





Konsequenz 2: Automaten mit ε-Übergängen akzeptieren nur DEA-Sprachen



akzeptiert z.B. bab ε bbaabaaba = babbbaabaaba

Konsequenz 3: 2-Wege-Automaten (DEAs, deren Lesekopf auch pro Rechenschritt verweilen oder nach links rücken darf) akzeptiern nur DEA-Sprachen

Beweisidee: M 2-Wege Automat; L=L(M)

Zeige, dass der Einfluss von w auf die Frage, ob $x \in F_L(w)$, d.h. wx wird von Maschine akzeptiert, sich im Wesentlichen nur auf eine Funktion $\mathbf{c: Q \mapsto Q}$ beschränkt.

Die Funktion **c** gibt folgendes an: wenn M mit Lesekopf auf rechtestem Zeichen von w im Zustand **q** gestartet wird, dann ist M das erste Mal, wenn der Lesekopf über das rechte Ende von w hinausrückt im Zustand **c(q)**.

Es gibt nur endlich viele verschiedene solche Funktionen **c**, und jede Folgesprache wird durch so eine Funktion bestimmt.

$L \subset \Sigma^*$ Sprache

$w \in \Sigma^*$: $F_L(w) = \{ x \in \Sigma^* \mid wx \in L \}$

Fortsetzungssprache von w bezüglich L

$\mathcal{F}_L = \{ \mathcal{F}_L(w) \mid w \in \Sigma^* \}$ Menge der Fortsetzungssprachen für L

Satz (Myhill – Nerode):

L DEA-Sprache $\Leftrightarrow$ $\mathcal{F}_L$ ist endlich

Satz (Myhill – Nerode):

L DEA-Sprache $\Leftrightarrow$ $\mathcal{F}_L$ ist endlich

Bew: "$\Rightarrow$"

L DEA-Sprache $\Rightarrow \exists$ DEA $M=(Q,\Sigma,s,F,\Delta)$ mit $L(M)=L$

für $q \in Q$ sei $L_q = \{x \in \Sigma^* \mid (q,x) \vdash_M^* (f,\varepsilon)$ für irgendein $f \in F \}$

$\mathcal{F}_L = \{ L_q \mid q \in Q$ **mit q von s erreichbar**$\}$ (plus möglicherweise $\{\}$)

und $\mathcal{F}_L$ ist endlich.

Satz (Myhill – Nerode):

L DEA-Sprache $\Leftrightarrow$ $\mathcal{F}_L$ ist endlich

Bew: "$\Leftarrow$"

$\mathcal{F}_L$ endlich. Definiere NEA $M = (\Sigma,Q,s,F,\Delta)$ mit

$Q = \mathcal{F}_L$

$s = F_L(\varepsilon) = L$

$F = \{ S \in \mathcal{F}_L \mid \varepsilon \in S \}$

$\Delta = \{ (F_L(w),a,F_L(wa)) \mid w \in \Sigma^*, a \in \Sigma \}$

Dann gilt

$$(F_L(x_1 \cdots x_k) , x_{k+1} \cdots x_n) \vdash_M (F_L(x_1 \cdots x_{k+1}) , x_{k+2} \cdots x_n)$$

und M akzeptiert genau L.

Konsequenz 4: (DEA Minimierung) Sei M ein DEA. Man kann effektiv einen DEA M' konstruieren, sodass L(M')=L(M) und die Anzahl der Zustände von M ist minimal. Diese Konstruktion braucht nur Zeit polynomiell in der Beschreibungsgröße von M'.

Beweis: $M=(\Sigma,Q,s,F,\Delta)$

für $q \in Q$ sei $L_q = \{x \in \Sigma^* \mid (q,x) \vdash_M^* (f,\varepsilon)$ für irgendein $f \in F \}$

$\mathcal{F}_L = \{ L_q \mid q \in Q$ **mit q von s erreichbar**$\}$ (plus möglicherweise $\{\}$)

Idee: Betrachte Übergangsgraphen G_M

1) Eliminiere $q \in Q$, die nicht von s erreichbar von G_M
   Vervollständige G_M, d.h. füge Knoten $\perp$ hinzu und
   für jedes $(q,a) \in (Q \cup \{\perp\}) \times \Sigma$ ohne Übergangsregel
   füge Regel (Kante) $(q,a,\perp)$ hinzu.
2) Im neuen Graphen bestimme, für welche Knotenpaare
   $\{p,q\}$ gilt $L_p = L_q$. ($\Rightarrow$ Äquivalenzrelation auf den Knoten)
3) Aus den Äquivalenzklassen bilde den Minimalautomaten.

Algorithmus für Schritt 2:

Ziel: Berechne $U = \{ \{p,q\} \mid L_p \neq L_q \}$ unterscheidbare Paare
und $N = \{ \{p,q\} \mid L_p = L_q \}$ nicht unterscheidbare Paare

$U := \{ \{p,q\} \mid p \in F$ und $q \in Q \setminus F \}$
$N := \{ \{p,q\} \mid p,q \in F$ oder $p,q \in Q \setminus F \}$

while $\exists \{p,q\} \in N$ und $\exists \{p',q'\} \in U$ und $\exists a \in \Sigma$
mit $(p,a,p') \in \Delta$ und $(q,a,q') \in \Delta$
do
verschiebe $\{p,q\}$ aus N nach U

Abschlusseigenschaften von regulären Sprachen (DKA-Sprachen)

Satz: Wenn L und L' reguläre Sprachen sind, dann sind auch folgende Sprachen regulär:

1) $L \cup L'$, $L \cap L'$, das Komplement von L
2) $L \cdot L' = \{ xy \mid x \in L \text{ und } y \in L' \}$ (Konkatenation)
3) $L^R = \{ x^R \mid x \in L \}$ (Umkehrung)
4) $L^i = L \cdot L^{i-1}$ für $i>0$ $(L^0 = \{\varepsilon\})$
5) $L^* = \bigcup_{i \in \mathbb{N}} L^i$ (Kleene Stern Operator)

Eigenschaften von regulären Sprachen

Satz: L, L' reguläre Sprachen, jeweils gegeben durch DEA M, M'

Folgende Probleme sind entscheidbar
(können effektiv, automatisch gelöst werden):

(i) $L = \Sigma^*$?

(ii) $L = \{\}$?

(iii) $L = L'$?

(iv) $L \subseteq L'$?

Pumping Lemma für reguläre Sprachen:

L DEA-Sprache $\Rightarrow$

$\exists\, N \in \mathbb{N}: \forall\, x \in L$ mit $|x| \geq N$: $\exists$ Unterteilung $x=uvw$ mit $|uv| \leq N$ und $|v|>0$: $\forall\, i \in \mathbb{N}: uv^i w \in L$

Pumping Lemma für reguläre Sprachen:

L DEA-Sprache $\Rightarrow$

$\exists\, N \in \mathbb{N}: \forall\, x \in L$ mit $|x| \geq N$: $\exists$ Unterteilung $x=uvw$ mit $|uv| \leq N$ und $|v|>0$: $\forall\, i \in \mathbb{N}: uv^i w \in L$

Beweisidee:

Jeder Pfad mit mindestens N Kanten in einem Graphen mit N Knoten hat eine Schleife.

Pumping Lemma für reguläre Sprachen:

L DEA-Sprache $\Rightarrow$

$\exists\, N \in \mathbb{N}: \forall\, x \in L$ mit $|x| \geq N$: $\exists$ Unterteilung $x=uvw$ mit $|uv| \leq N$ und $|v|>0$: $\forall\, i \in \mathbb{N}: uv^i w \in L$

$\neg \big(\exists\, N \in \mathbb{N}: \forall\, x \in L$ mit $|x| \geq N$: $\exists$ Unterteilung $x=uvw$ mit $|uv| \leq N$ und $|v|>0$: $\forall\, i \in \mathbb{N}: uv^i w \notin L \big)$

$\Rightarrow \quad \neg$ (L DEA-Sprache)

Pumping Lemma für reguläre Sprachen:

L DEA-Sprache $\Rightarrow$

$\exists\, N \in \mathbb{N}: \forall\, x \in L$ mit $|x| \geq N$: $\exists$ Unterteilung $x=uvw$ mit $|uv| \leq N$ und $|v|>0$: $\forall\, i \in \mathbb{N}: uv^i w \in L$

$\neg \big(\exists\, N \in \mathbb{N}: \forall\, x \in L$ mit $|x| \geq N$: $\exists$ Unterteilung $x=uvw$ mit $|uv| \leq N$ und $|v|>0$: $\forall\, i \in \mathbb{N}: uv^i w \notin L \big)$

$\Rightarrow \quad \neg$ (L DEA-Sprache)

$\forall\, N \in \mathbb{N}: \exists\, x \in L$ mit $|x| \geq N$: $\forall$ Unterteilung $x=uvw$ mit $|uv| \leq N$ und $|v|>0$: $\exists\, i \in \mathbb{N}: uv^i w \notin L$

$\Rightarrow$ L ist **keine** DEA-Sprache

"Spiel" zum Zeigen, dass L keine DEA Sprache

1. Gegner gibt eine Zahl $N \in \mathbb{N}$ vor.
2. Ich wähle ein $x \in L$ mit $|x| \geq N$.
3. Gegner gibt eine Unterteilung $x = uvw$ vor mit $|uv| \geq N$, $|v|>0$.
4. Ich wähle ein $i \in \mathbb{N}$, sodass $uv^i w \notin L$.

Reguläre Ausdrücke

Mechanismus zum Spezifizieren von Sprachen über Alphabet Σ

Ausdruck	Bedeutung
$\emptyset$	$[\![\emptyset]\!] = \{\}$
ε	$[\![\varepsilon]\!] = \{\varepsilon\}$
$a \in \Sigma$ $\quad a$	$[\![a]\!] = \{a\}$

r_1, r_2 reg. Ausdrücke

(r_1+r_2)	$[\![(r_1+r_2)]\!] = [\![r_1]\!] \cup [\![r_2]\!]$
$r_1\, r_2$	$[\![r_1\, r_2]\!] = [\![r_1]\!] \cdot [\![r_2]\!]$
$(r_1)^*$	$[\![(r_1)^*]\!] = [\![r_1]\!]^*$

Reguläre Ausdrücke definieren genau die DKA-Sprachen.

Satz: $L = [\![r]\!]$ für einen regulären Ausdruck r genau dann, wenn $L = L(M)$ für irgendeinen DEA M.

Beweis: "⇒"

Idee: strukturelle Induktion; baue NEA mit ε-Übergängen

"⇐" Gegeben DEA $M=(\Sigma,Q,s,F,\Delta)$, Übergangsgraph G_M

$$Q = \{q_1,\cdots,q_n\},\ s = q_1,\ F = \{q_{j_1},\cdots,q_{j_f}\}$$

Für $0 \le k \le n$, $1 \le i,j \le n$ sei r^k_{ij} Ausdruck, der alle Beschriftungen von Pfaden in G_M beschreibt mit Anfangsknoten q_i, Endknoten q_j, und **internen** Knoten mit Index $\le k$.

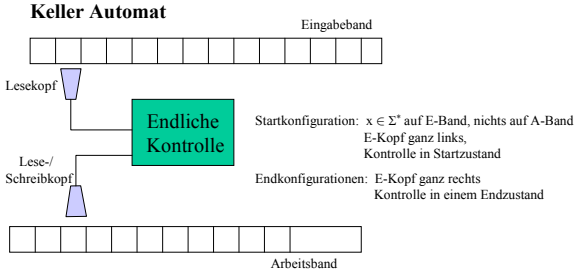
$$r^0_{ii} = \varepsilon + a_1 + \cdots a_t \text{ mit } (q_i,a_h,q_i)\in\Delta \text{ für } 1\le h\le t$$

$i \ne j$ $\quad r^0_{ij} = a_1 + \cdots a_t$ mit $(q_i,a_h,q_i)\in\Delta$ für $1\le h\le t$

$k>0$ $\quad r^k_{ij} = r^{k-1}_{ij} + r^{k-1}_{ik}(r^{k-1}_{kk})^* r^{k-1}_{kj}$

$$L(M) = r^n_{1j_1} + r^n_{1j_2} + \cdots + r^n_{1j_f}$$

Keller Automat



Startkonfiguration: $x \in \Sigma^*$ auf E-Band, nichts auf A-Band
E-Kopf ganz links,
Kontrolle in Startzustand

Endkonfigurationen: E-Kopf ganz rechts
Kontrolle in einem Endzustand

Rechenschrittregeln:

derzeitiger Zustand | gelesenes E-Symbol | gelesenes A-Symbol → neuer Zustand | E-Kopf nach rechts | schreibe A-Symbol | bewege A-Kopf

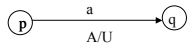
Wenn A-Kopf nach rechts bewegt wird, hinterlässt er leere Zelle.

Formale Spezifikation von Kellerautomaten

Σ Eingabealphabet
Γ Kelleralphabet
$\epsilon \in \Gamma$ Kellerboden
Q Zustandsmenge (endlich)
$s \in Q$ Startzustand
$F \subset Q$ Endzustände
$\Delta \subset (Q \times \Gamma \times (\Sigma \cup \{\varepsilon\})) \times \Gamma^* \times Q$ Regeln (endlich)

derz. Zust. | derz. Keller-gipfel | konumierte Eingabe | neuer Keller-gipfel | neuer Zustand

$(p, A, a, U, q) \in \Delta$ notiert durch



Konfiguration: $Q \times \Gamma^* \times \Sigma^*$

Zustand Kellerinhalt Eingaberest

Rechenschrittrelation:

$(p, AU, au) \vdash_M (q, VU, u)$ g.d.w. $(p,A,a,V,q) \in \Delta$
$(p, AU, u) \vdash_M (q, VU, u)$ g.d.w. $(p,A,\varepsilon,V,q) \in \Delta$

$A \in \Gamma$, $U \in \Gamma^*$, $a \in \Sigma$, $u \in \Sigma^*$

Rechenrelation:
$\vdash_M^*$ reflexive, transitive Hülle von $\vdash_M$

Startkonfiguration für Eingabe x : $start_x = (s, \epsilon, x)$

Endkonfigurationen: $Fin = \{ (f, W, \varepsilon) \mid W \in \Gamma^*, f \in F \}$
(Akzeptanz durch Endzustand)

$Fin_\varepsilon = \{ (q, \varepsilon, \varepsilon) \mid q \in Q \}$
(Akzeptanz durch leeren Keller)

Von M akzeptierter Sprache:

$L(M) = \{ x \in \Sigma^* \mid start_x \vdash_M^* \varphi \text{ für irgendein } \varphi \in Fin \}$

$L_\varepsilon(M) = \{ x \in \Sigma^* \mid start_x \vdash_M^* \varphi \text{ für irgendein } \varphi \in Fin_\varepsilon \}$

Startkonfiguration für Eingabe x : $start_x = (s, \epsilon, x)$

Endkonfigurationen: $Fin = \{ (f, W, \varepsilon) \mid W \in \Gamma^*, f \in F \}$
(Akzeptanz durch Endzustand)

$Fin_\varepsilon = \{ (q, \varepsilon, \varepsilon) \mid q \in Q \}$
(Akzeptanz durch leeren Keller)

Von M akzeptierter Sprache:

$L(M) = \{ x \in \Sigma^* \mid start_x \vdash_M^* \varphi \text{ für irgendein } \varphi \in Fin \}$

$L_\varepsilon(M) = \{ x \in \Sigma^* \mid start_x \vdash_M^* \varphi \text{ für irgendein } \varphi \in Fin_\varepsilon \}$

Achtung: Um Wort x zu akzeptieren, muss das gesamte x konsumiert werden.

Satz: **L=L(M) für irgendeinen NKA M**

$\Leftrightarrow$

$L=L_\varepsilon(M')$ für irgendeinen NKA M'

"Äquivalenz von Akzeptanz durch Endzustand und Akzeptanz durch leeren Keller."

Def.: L heißt *NKA-Sprache* (oder *kontextfreie Sprache*), wenn $L=L(M)$ oder $L=L_\varepsilon(M)$ für irgendeinen NKA M.

Pumping Lemma für NKA Sprachen:

L NKA-Sprache $\Rightarrow$

$\exists N \in \mathbb{N}: \forall z \in L$ mit $|z| \geq N$: $\exists$ Unterteilung $z=uvwxy$ mit $|vwx| \leq N$ und $|vx|>0$: $\forall i \in \mathbb{N}: uv^iwx^iy \in L$

Pumping Lemma für NKA Sprachen:

L NKA-Sprache $\Rightarrow$

$\exists N \in \mathbb{N}: \forall z \in L$ mit $|z| \geq N$: $\exists$ Unterteilung $z=uvwxy$ mit $|vwx| \leq N$ und $|vx|>0$: $\forall i \in \mathbb{N}: uv^iwx^iy \in L$

$\neg$ ($\exists N \in \mathbb{N}: \forall z \in L$ mit $|z| \geq N$: $\exists$ Unterteilung $z=uvwxy$ mit $|vwx| \leq N$ und $|vx|>0$: $\forall i \in \mathbb{N}: uv^iwx^iy \in L$)

$\Rightarrow$ $\neg$ (L NKA-Sprache)

Pumping Lemma für NKA Sprachen:

L NKA-Sprache $\Rightarrow$

$\exists N \in \mathbb{N}: \forall z \in L$ mit $|z| \geq N$: $\exists$ Unterteilung $z=uvwxy$ mit $|vwx| \leq N$ und $|vx|>0$: $\forall i \in \mathbb{N}: uv^iwx^iy \in L$

$\neg$ ($\exists N \in \mathbb{N}: \forall z \in L$ mit $|z| \geq N$: $\exists$ Unterteilung $z=uvwxy$ mit $|vwx| \leq N$ und $|vx|>0$: $\forall i \in \mathbb{N}: uv^iwx^iy \in L$)

$\Rightarrow$ $\neg$ (L NKA-Sprache)

$\forall N \in \mathbb{N}: \exists z \in L$ mit $|z| \geq N$: $\forall$ Unterteilung $z=uvwxy$ mit $|vwx| \leq N$ und $|vx|>0$: $\exists i \in \mathbb{N}: uv^iwx^iy \notin L$

$\Rightarrow$ L ist **keine** NKA-Sprache

"Spiel" zum Zeigen, dass L keine NKA Sprache

1. Gegner gibt eine Zahl $N \in \mathbb{N}$ vor.
2. Ich wähle ein $z \in L$ mit $|z| \geq N$.
3. Gegner gibt eine Unterteilung $z = uvwxy$ vor mit $|vwx| \leq N$, $|vx|>0$.
4. Ich wähle ein $i \in \mathbb{N}$, sodass $uv^iwx^iy \notin L$.

“Spiel” zum Zeigen, dass L keine NKA Sprache

1. Gegner gibt eine Zahl $N \in \mathbb{N}$ vor.
2. Ich wähle ein $z \in L$ mit $|z| \geq N$.
3. Gegner gibt eine Unterteilung $z = uvwxy$ vor mit $|vwx| \leq N$, $|vx| > 0$.
4. Ich wähle ein $i \in \mathbb{N}$, sodass $uv^i wx^i y \notin L$.

1. Gegner gibt eine Zahl $N \in \mathbb{N}$ vor.
2. Ich wähle ein $z \in L$ mit $|z| \geq N$.
3. Gegner gibt eine Unterteilung $z = uvwxy$ vor mit $|vwx| \leq N$, $|vx| > 0$.
4. Ich wähle ein $i \in \mathbb{N}$, sodass $uv^i wx^i y \notin L$.

Bsp: $L_{abc} = \{ a^n b^n c^n \mid n \in \mathbb{N} \}$ ist nicht DKA-Sprache.

1. Gegner gibt eine Zahl $N \in \mathbb{N}$ vor.
2. Ich wähle ein $z \in L$ mit $|z| \geq N$.
3. Gegner gibt eine Unterteilung $z = uvwxy$ vor mit $|vwx| \leq N$, $|vx| > 0$.
4. Ich wähle ein $i \in \mathbb{N}$, sodass $uv^i wx^i y \notin L$.

Bsp: $L_{abc} = \{ a^n b^n c^n \mid n \in \mathbb{N} \}$ ist nicht DKA-Sprache.

Bew: 1. Gegner gibt N vor.

1. Gegner gibt eine Zahl $N \in \mathbb{N}$ vor.
2. Ich wähle ein $z \in L$ mit $|z| \geq N$.
3. Gegner gibt eine Unterteilung $z = uvwxy$ vor mit $|vwx| \leq N$, $|vx| > 0$.
4. Ich wähle ein $i \in \mathbb{N}$, sodass $uv^i wx^i y \notin L$.

Bsp: $L_{abc} = \{ a^n b^n c^n \mid n \in \mathbb{N} \}$ ist nicht DKA-Sprache.

Bew: 1. Gegner gibt N vor.

2. Ich wähle $z = a^N b^N c^N \in L_{abc}$

1. Gegner gibt eine Zahl $N \in \mathbb{N}$ vor.
2. Ich wähle ein $z \in L$ mit $|z| \geq N$.
3. Gegner gibt eine Unterteilung $z = uvwxy$ vor mit $|vwx| \leq N$, $|vx| > 0$.
4. Ich wähle ein $i \in \mathbb{N}$, sodass $uv^i wx^i y \notin L$.

Bsp: $L_{abc} = \{ a^n b^n c^n \mid n \in \mathbb{N} \}$ ist nicht DKA-Sprache.

Bew: 1. Gegner gibt N vor.

2. Ich wähle $z = a^N b^N c^N \in L_{abc}$
3. Gegner unterteilt: $a^N b^N c^N = uvwxy$ mit $|vwx| \leq N$, $|vx| > 0$.

Da $|vwx| \leq N$ kommt einer der Buchstaben a,b,c nicht in vwx vor, nennen wir ihn δ.

1. Gegner gibt eine Zahl $N \in \mathbb{N}$ vor.
2. Ich wähle ein $z \in L$ mit $|z| \geq N$.
3. Gegner gibt eine Unterteilung $z = uvwxy$ vor mit $|vwx| \leq N$, $|vx| > 0$.
4. Ich wähle ein $i \in \mathbb{N}$, sodass $uv^i wx^i y \notin L$.

Bsp: $L_{abc} = \{ a^n b^n c^n \mid n \in \mathbb{N} \}$ ist nicht DKA-Sprache.

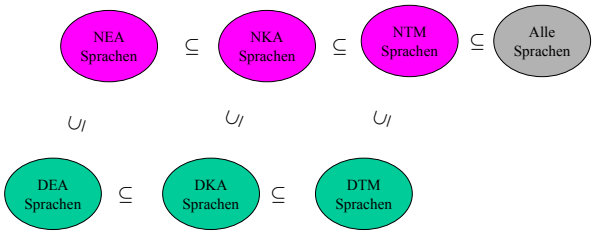
Bew: 1. Gegner gibt N vor.

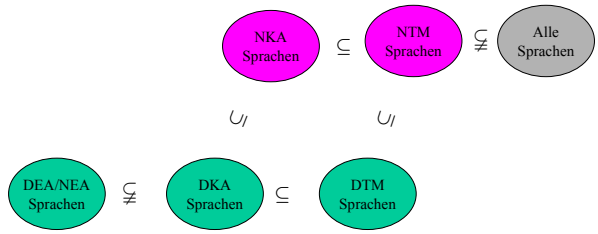
2. Ich wähle $z = a^N b^N c^N \in L_{abc}$
3. Gegner unterteilt: $a^N b^N c^N = uvwxy$ mit $|vwx| \leq N$, $|vx| > 0$.

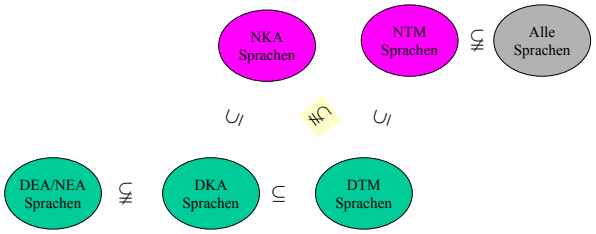
Da $|vwx| \leq N$ kommt einer der Buchstaben a,b,c nicht in vwx vor, nennen wir ihn δ.

4. Ich wähle $i=0$: $uv^0 wx^0 y = uwy$ enthält δ genau N mal, aber einen der anderen beiden Buchstaben weniger als N mal. (Da $|vx| > 0$)

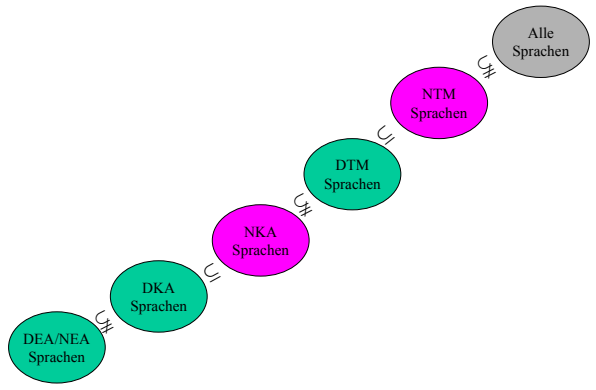
Also gilt $uv^0 wx^0 y \notin L_{abc}$







L_{abc} ist nicht NKA-Sprache, aber DTM-Sprache.



Satz: Seien L und L' NKA-Sprachen (kontextfreie Sprachen) und sei R eine reguläre Sprache. Es gilt:

1) $L \cup L'$ ist NKA-Sprache
2) $L \cap R$ ist NKA-Sprache
3) $L \cap L'$ ist nicht unbedingt NKA-Sprache
4) Das Komplement von L ist nicht unbedingt NKA-Sprache.

Satz: Seien L und L' NKA-Sprachen (kontextfreie Sprachen) und sei R eine reguläre Sprache. Es gilt:

1) $L \cup L'$ ist NKA-Sprache
2) $L \cap R$ ist NKA-Sprache
3) $L \cap L'$ ist nicht unbedingt NKA-Sprache
4) Das Komplement von L ist nicht unbedingt NKA-Sprache.

Beweis von 3): $L = \{ a^k b^k c^m \mid k,m \in \mathbb{N} \}$ ist NKA-Sprache
$L' = \{ a^m b^n c^n \mid m,n \in \mathbb{N} \}$ ist NKA-Sprache

$L \cap L' = \{ a^n b^n c^n \mid n \in \mathbb{N} \} = L_{abc}$ ist nicht NKA-Sprache.

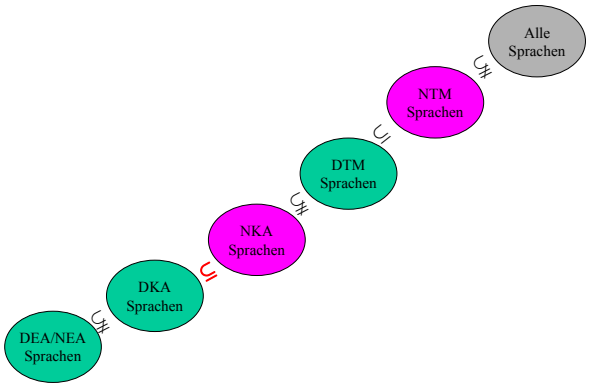
Satz: Seien L und L' NKA-Sprachen (kontextfreie Sprachen) und sei R eine reguläre Sprache. Es gilt:

1) $L \cup L'$ ist NKA-Sprache
2) $L \cap R$ ist NKA-Sprache
3) $L \cap L'$ ist nicht unbedingt NKA-Sprache
4) Das Komplement von L ist nicht unbedingt NKA-Sprache.

Beweis von 4): Sei L das Komplement von L_{abc}.

Überlege, dass L eine NKA-Sprache ist.

Aber das Komplement von L ist L_{abc} , und das ist keine NKA-Sprache.



Satz: NKA-Sprachen (kontextfreie Sprachen) sind unter Komplementbildung **nicht** abgeschlossen.

Das heißt,
wenn L eine NKA-Sprache ist, dann ist das Komplement von L nicht unbedingt eine NKA Sprache.

Beweis: Betrachte L, das Komplement von $L_{abc}=\{a^n b^n c^n \mid n\in\mathbb{N}\}$.

Überlege, dass L eine NKA-Sprache ist.

Aber das Komplement von L ist L_{abc} , und das ist keine NKA-Sprache.

Satz: NKA-Sprachen (kontextfreie Sprachen) sind unter Komplementbildung **nicht** abgeschlossen.

Das heißt,
wenn L eine NKA-Sprache ist, dann ist das Komplement von L nicht unbedingt eine NKA Sprache.

Satz: DKA-Sprachen sind unter Komplementbildung abgeschlossen.

Das heißt,
wenn L eine DKA-Sprache ist, dann ist das Komplement von L auf jeden Fall eine DKA Sprache.

Satz: NKA-Sprachen (kontextfreie Sprachen) sind unter Komplementbildung **nicht** abgeschlossen.

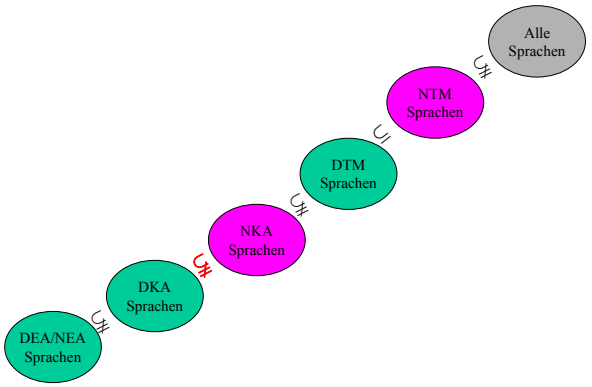
Das heißt,
wenn L eine NKA-Sprache ist, dann ist das Komplement von L nicht unbedingt eine NKA Sprache.

Satz: DKA-Sprachen sind unter Komplementbildung abgeschlossen.

Das heißt,
wenn L eine DKA-Sprache ist, dann ist das Komplement von L auf jeden Fall eine DKA Sprache.

Kor: DKA-Sprachen sind eine echte Teilmenge der NKA-Sprachen.

z.B. $\overline{L_{abc}}$ ist NKA-Spr. aber nicht DKA-Spr.



Satz: DKA-Sprachen sind unter Komplementbildung abgeschlossen.

Das heißt,
wenn L eine DKA-Sprache ist, dann ist das Komplement von L auf jeden Fall eine DKA Sprache.

Beweis:

Idee: Baue aus DKA M für L, einen DKA M' für das Komplement von L
(durch Vertauschung von End- und Nicht-Endzuständen).

Details: 1) fehlende Übergänge
2) Endlosschleifen von ε-Übergängen
3) Folgen von ε-Übergängen durch End- und Nicht-Endzuständen nach Konsum der gesamten Eingabe.

Satz: Jede NKA-Sprache über einem 1-buchstabigen Alphabet ist eine reguläre Sprache.

Beweis: geschickte Anwendung des Pumpinglemmas;

Darstellung der Sprache als Vereinigung endlich vieler „einfacher“ regulärere Sprchen.

Satz: Wird eine Sprache L von einem NKA M akzeptiert, dann wird L auch von einem NKA M' durch leeren Keller akzeptiert, wobei M' nur **einen Zustand** besitzt.

Satz: Wird eine Sprache L von einem NKA M akzeptiert, dann wird L auch von einem NKA M' durch leeren Keller akzeptiert, wobei M' nur **einen Zustand** besitzt.

Beachte: Man kann M' auch als NKA **ohne Zustand** auffassen.

Satz: Wird eine Sprache L von einem NKA M akzeptiert, dann wird L auch von einem NKA M' durch leeren Keller akzeptiert, wobei M' nur **einen Zustand** besitzt.

Beweisidee: Lasse M' den NKA M nicht-deterministisch simulieren
Der Zustand von M wird auf dem Keller von M' gespeichert
Instanz von A auf Keller von M wird realisiert als
(q,A,q') auf Keller von M' mit der Bedeutung:

Wenn diese Instanz das nächste Mal oberstes Kellersymbol ist,
dann ist M in Zustand q,
und wenn das Kellersymbol unter dieser Instanz betrachtet wird,
dann ist M im Zustand q'.

Beweisidee: Lasse M' den NKA M nicht-deterministisch simulieren
Der Zustand von M wird auf dem Keller von M' gespeichert
Instanz von A auf Keller von M wird realisiert als
(q,A,q') auf Keller von M' mit der Bedeutung:

Wenn diese Instanz das nächste Mal oberstes Kellersymbol ist,
dann ist M in Zustand q,
und wenn das Kellersymbol unter dieser Instanz betrachtet wird,
dann ist M im Zustand q'.

Kellerinhalt von M im Zustand q $A_k\, A_{k-1}\, A_{k-2} \cdots A_1$
entspricht
Kellerinhalt von M' $(q,A_k,p_{k-1})\,(p_{k-1},A_{k-1},p_{k-2})\,(p_{k-2},A_{k-2},p_{k-3}) \cdots (p_1,A_1,p_0)$

für geeignete (nicht-deterministisch gewählte) Zustände $p_{k-1},p_{k-2},p_{k-3} \cdots p_1,p_0$

Grammatik alternativer Sprachspezifikationsmechanismus

$G = (\Sigma,V,S,P)$
Σ Terminalalphabet
V Variablen- (Nicht-terminal) –alphabet
$S \in V$ Startvariable
$P \subset FVF \times F$ "Produktionen" ($F = (\Sigma \cup V)^*$)

Σ,V,S,P müssen endlich sein
$(\alpha,\beta) \in P$ wird geschrieben als $\alpha \rightarrow \beta$

Grammatik alternativer Sprachspezifikationsmechanismus

$G = (\Sigma,V,S,P)$
Σ Terminalalphabet
V Variablen- (Nicht-terminal) –alphabet
$S \in V$ Startvariable
$P \subset FVF \times F$ "Produktionen" ($F = (\Sigma \cup V)^*$)

Σ,V,S,P müssen endlich sein
$(\alpha,\beta) \in P$ wird geschrieben als $\alpha \rightarrow \beta$

G induziert Ableitungsschritt-Relation (Derivationsschritt-Relation) $\Rightarrow_G$ auf F durch $\gamma\alpha\gamma' \Rightarrow_G \gamma\beta\gamma'$ wenn $\alpha \rightarrow \beta$ Produktion in P

$\Rightarrow_G^*$ reflexive, transitive Hülle von $\Rightarrow_G$: Ableitungsrelation (Derivationsrelation) auf F

G induziert Ableitungsschritt-Relation (Derivationsschritt-Relation) $\Rightarrow_G$ auf F durch $\gamma\alpha\gamma' \Rightarrow_G \gamma\beta\gamma'$ wenn $\alpha \rightarrow \beta$ Produktion in P

$\Rightarrow_G^*$ reflexive, transitive Hülle von $\Rightarrow_G$: Ableitungsrelation (Derivationsrelation) auf F

Die von G generierte Sprache

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

Beispiel: $G = (\Sigma,V,S,P)$ mit $\Sigma=\{a,b,c\}$, $V=\{C,D,S\}$ und

$P = \{ S \rightarrow aDbc , S \rightarrow \varepsilon ,$
$D \rightarrow aDbC , D \rightarrow \varepsilon ,$
$Cb \rightarrow bC ,$
$Cc \rightarrow cc \quad \}$

Beispielableitung:

$S \Rightarrow_G aDbc \Rightarrow_G aaDbCbc \Rightarrow_G aaaDbCbCbc \Rightarrow_G aaabCbCbc \Rightarrow_G$
$aaabbCCbc \Rightarrow_G aaabbCbCc \Rightarrow_G aaabbbCCc \Rightarrow_G$
$aaabbbCcc \Rightarrow_G aaabbbccc$

Beispiel: $G = (\Sigma,V,S,P)$ mit $\Sigma=\{a,b\}$, $V=\{C,D,S\}$ und

$P = \{ S \rightarrow aDbc , S \rightarrow \varepsilon ,$
$D \rightarrow aDbC , D \rightarrow \varepsilon ,$
$Cb \rightarrow bC ,$
$Cc \rightarrow cc \quad \}$

Behauptung: $L(G) = \{ a^n b^n c^n \mid n \in \mathbb{N} \}$

Beweis: "$\subseteq$"

1) Jeder abgeleitete String enthält gleich viele a's wie b's wie {c,C}'s.
2) Alle a's kommen vor allen b's und vor allen {c,C}'s vor.
3) Die c's, die in einem abgeleiteten String vorkommen, bilden immer einen Suffix dieses Strings.

Grammatik alternativer Sprachspezifikationsmechanismus

$G = (\Sigma, V, S, P)$

Σ Terminalalphabet
V Variablen- (Nicht-terminal) –alphabet
$S \in V$ Startvariable
$P \subset FVF \times F$ "Produktionen" ($F = (\Sigma \cup V)^*$)

Σ, V, S, P müssen endlich sein
$(\alpha,\beta) \in P$ wird geschrieben als $\alpha \rightarrow \beta$

Grammatik alternativer Sprachspezifikationsmechanismus

$G = (\Sigma, V, S, P)$

Σ Terminalalphabet
V Variablen- (Nicht-terminal) –alphabet
$S \in V$ Startvariable
$P \subset FVF \times F$ "Produktionen" ($F = (\Sigma \cup V)^*$)

Σ, V, S, P müssen endlich sein
$(\alpha,\beta) \in P$ wird geschrieben als $\alpha \rightarrow \beta$

G induziert Ableitungsschritt-Relation (Derivationsschritt-Relation)
$\Rightarrow_G$ auf F durch $\gamma\alpha\gamma' \Rightarrow_G \gamma\beta\gamma'$ wenn $\alpha \rightarrow \beta$ Produktion in P

$\Rightarrow_G^*$ reflexive, transitive Hülle von $\Rightarrow_G$: Ableitungsrelation (Derivationsrelation) auf F

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

Chomsky Hierarchie für Grammatiken und Sprachen

Typ 0 (unbeschränkt)

Typ 1 (kontextsensitiv)
nur Regeln $\alpha \rightarrow \beta$ mit $|\alpha| \leq |\beta|$

Typ 2 (kontextfrei)
nur Regeln $A \rightarrow \alpha$ mit $A \in V$

Typ 3 (rechtslinear)
nur Regeln $A \rightarrow uB$, $A \rightarrow u$, $A \rightarrow \varepsilon$ mit $A,B \in V$ und $u \in \Sigma$

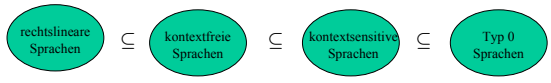
Chomsky Hierarchie für Grammatiken und Sprachen

Typ 0 (unbeschränkt)

Typ 1 (kontextsensitiv)
nur Regeln $\alpha \rightarrow \beta$ mit $|\alpha| \leq |\beta|$

Typ 2 (kontextfrei)
nur Regeln $A \rightarrow \alpha$ mit $A \in V$

Typ 3 (rechtslinear)
nur Regeln $A \rightarrow uB$, $A \rightarrow u$, $A \rightarrow \varepsilon$ mit $A,B \in V$ und $u \in \Sigma$



Satz: Die rechtslinearen Sprachen sind genau die regulären Sprachen.

Beweis:
1) L rechtslinear $\Rightarrow$ L regulär
Idee: zeige, dass L nur endlich viele Fortsetzungssprachen hat
2) L regulär $\Rightarrow$ L rechtslinear

L regulär $\Rightarrow L = L(M)$ für DEA $M = (\Sigma, Q, s, F, \Delta)$

Betrachte Grammatik $G = (\Sigma, Q, s, P)$ mit

$P = \{ p \rightarrow uq \mid (p,u,q) \in \Delta \} \cup \{ p \rightarrow \varepsilon \mid p \in F \}$

Kontextfreie Grammatiken und Sprachen

Bsp: kfG für geklammerte arithmetische Ausdrücke über Binärzahlen und Variablennamen über $\{a,b\}^*$

$G = (\Sigma, V, E, P)$ mit $\Sigma = \{a,b,0,1,(,),*,+\}$
$V = \{E,K,W\}$
und Produktionen P: $E \rightarrow E+E \mid E*E \mid (E) \mid K \mid W$
$W \rightarrow aW \mid bW \mid a \mid b$
$K \rightarrow 0K \mid 1K \mid 0 \mid 1$

$11*(a+1) \in L(G)$ $(a+b)(a+1) \notin L(G)$

$G = (\Sigma,V,E,P)$ mit $\Sigma = \{a,b,0,1,(,),*,+\}$
$V=\{E,K,W\}$
und Produktionen P: $E \to E+E \mid E*E \mid (E) \mid K \mid W$
$W \to aW \mid bW \mid a \mid b$
$K \to 0K \mid 1K \mid 0 \mid 1$

$E \Rightarrow E*\underline{E} \Rightarrow E*(E) \Rightarrow \underline{E}*(E+E)$
$\Rightarrow K*(\underline{E}+E) \Rightarrow K*(W+\underline{E})$
$\Rightarrow \underline{K}*(W+K) \Rightarrow 1\underline{K}*(W+K)$
$\Rightarrow 11*(W+\underline{K}) \Rightarrow 11*(\underline{W}+1)$
$\Rightarrow 11*(a+1)$

$G = (\Sigma,V,E,P)$ mit $\Sigma = \{a,b,0,1,(,),*,+\}$
$V=\{E,K,W\}$
und Produktionen P: $E \to E+E \mid E*E \mid (E) \mid K \mid W$
$W \to aW \mid bW \mid a \mid b$
$K \to 0K \mid 1K \mid 0 \mid 1$

$E \Rightarrow E*\underline{E} \Rightarrow E*(E) \Rightarrow \underline{E}*(E+E)$
$\Rightarrow K*(\underline{E}+E) \Rightarrow K*(W+\underline{E})$
$\Rightarrow \underline{K}*(W+K) \Rightarrow 1\underline{K}*(W+K)$
$\Rightarrow 11*(W+\underline{K}) \Rightarrow 11*(\underline{W}+1)$
$\Rightarrow 11*(a+1)$

$E \Rightarrow \underline{E}*E \Rightarrow \underline{K}*E \Rightarrow 1\underline{K}*E$
$\Rightarrow 11*\underline{E} \Rightarrow 11*(E) \Rightarrow 11*(\underline{E}+E)$
$\Rightarrow 11*(\underline{W}+E) \Rightarrow 11*(a+\underline{E})$
$\Rightarrow 11*(a+\underline{K}) \Rightarrow 11*(a+1)$

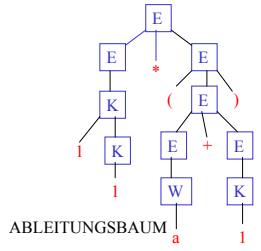
LINKSABLEITUNG

$G = (\Sigma,V,E,P)$ mit $\Sigma = \{a,b,0,1,(,),*,+\}$
$V=\{E,K,W\}$
und Produktionen P: $E \to E+E \mid E*E \mid (E) \mid K \mid W$
$W \to aW \mid bW \mid a \mid b$
$K \to 0K \mid 1K \mid 0 \mid 1$

$E \Rightarrow E*\underline{E} \Rightarrow E*(E) \Rightarrow \underline{E}*(E+E)$
$\Rightarrow K*(\underline{E}+E) \Rightarrow K*(W+\underline{E})$
$\Rightarrow \underline{K}*(W+K) \Rightarrow 1\underline{K}*(W+K)$
$\Rightarrow 11*(W+\underline{K}) \Rightarrow 11*(\underline{W}+1)$
$\Rightarrow 11*(a+1)$

$E \Rightarrow \underline{E}*E \Rightarrow \underline{K}*E \Rightarrow 1\underline{K}*E$
$\Rightarrow 11*\underline{E} \Rightarrow 11*(E) \Rightarrow 11*(\underline{E}+E)$
$\Rightarrow 11*(\underline{W}+E) \Rightarrow 11*(a+\underline{E})$
$\Rightarrow 11*(a+\underline{K}) \Rightarrow 11*(a+1)$

LINKSABLEITUNG



ABLEITUNGSBAUM

Lemma: $G = (\Sigma,V,S,P)$ kontextfreie Grammatik, $\alpha \in (\Sigma \cup V)^*$

$\exists$ Ableitung $S \Rightarrow_G^* \alpha$
$\Leftrightarrow$ $\exists$ Linksableitung $S \Rightarrow_G^* \alpha$
$\Leftrightarrow$ $\exists$ Ableitungsbaum mit α als Blätterbeschriftung

Satz: Die kontextfreien Sprachen sind genau die NKA-Sprachen.
$L=L(G)$ für irgendeine kfG $G = (\Sigma,V,S,P)$
$\Leftrightarrow$ $L=L_\varepsilon(M)$ für irgendeinen NKA $M=(\Sigma,\Gamma,\epsilon,Q,s,\Delta)$

Satz: Die kontextfreien Sprachen sind genau die NKA-Sprachen.
$L=L(G)$ für irgendeine kfG $G = (\Sigma,V,S,P)$
$\Leftrightarrow$ $L=L_\varepsilon(M)$ für irgendeinen NKA $M=(\Sigma,\Gamma,\epsilon,Q,s,\Delta)$

Beweis: 1) "$\Rightarrow$" Gegeben Grammatik G, baue NKA M, dessen akzeptierende Berechnungen Linksableitungen in G entsprechen

$\Gamma = V \cup \Sigma \qquad \epsilon = S \qquad Q=\{s\}$

$\Delta = \{ (s,A,\varepsilon,\alpha,s) \mid A \to \alpha \text{ in } P\} \cup \{ (s,a,a,\varepsilon,s) \mid a \in \Sigma \}$

Satz: Die kontextfreien Sprachen sind genau die NKA-Sprachen.
L=L(G) für irgendeine kfG G = (Σ,V,S,P)
⇔ L=L_ε(M) für irgendeinen NKA M=(Σ,Γ,Є,Q,s,Δ)

Satz: Die kontextfreien Sprachen sind genau die NKA-Sprachen.
L=L(G) für irgendeine kfG G = (Σ,V,S,P)
⇔ L=L_ε(M) für irgendeinen NKA M=(Σ,Γ,Є,Q,s,Δ)

Beweis: 1) "⇒" Gegeben Grammatik G, baue NKA M, dessen akzeptierende Berechnungen Linksableitungen in G entsprechen

Γ = V∪Σ Є = S Q={s}

Δ = { (s,A,ε,α,s) | A→α in P} ∪ { (s,a,a,ε,s) | a∈Σ }

Satz: Die kontextfreien Sprachen sind genau die NKA-Sprachen.
L=L(G) für irgendeine kfG G = (Σ,V,S,P)
⇔ L=L_ε(M) für irgendeinen NKA M=(Σ,Γ,Є,Q,s,Δ)

Beweis: 2) "⇐" Gegeben NKA M=(Σ,Γ,Є,Q,s,Δ), konstruiere kf Grammatik G, sodass Linksableitungen in G den akzeptierende Berechnungen von M entsprechen. O.B.d.A. gilt |Q|=1

V = Γ S = Є Q={s}

P = { A→aα | (s,A,a,α,s) ∈ Δ } (a ∈ Σ∪{ε})

Chomsky-Normalform (für kf Grammatiken)
alle Produktionen von Form A→a oder A→BC (A,B,C∈V, a∈Σ)

Chomsky-Normalform (für kf Grammatiken)
alle Produktionen von Form A→a oder A→BC (A,B,C∈V, a∈Σ)

Greibach-Normalform (für kf Grammatiken)
alle Produktionen von Form A→aU (U∈V^*, a∈Σ)

S→ε auch zugelassen, aber dann S auf keiner rechten Produktionsseite

Chomsky-Normalform (für kf Grammatiken)
alle Produktionen von Form A→a oder A→BC (A,B,C∈V, a∈Σ)

Greibach-Normalform (für kf Grammatiken)
alle Produktionen von Form A→aU (U∈V^*, a∈Σ)

S→ε auch zugelassen, aber dann S auf keiner rechten Produktionsseite

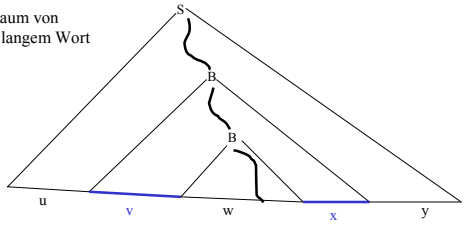
Satz: L = L(G) für kf Grammatik G
⇔ ∃ kf Grammatik G in Chomsky-Normalform mit L(G) = G
⇔ ∃ kf Grammatik G in Greibach-Normalform mit L(G) = G

Beweis des Pumping Lemmas für NKA Sprachen:

L NKA-Sprache $\Rightarrow$

$\exists$ N$\in\mathbb{N}$: $\forall$ z$\in$L : $\exists$ Unterteilung z=uvwxy : $\forall$ i$\in\mathbb{N}$: $uv^iwx^iy\in L$
mit $|z|\geq N$ mit $|vwx|\leq N$ und $|vx|>0$

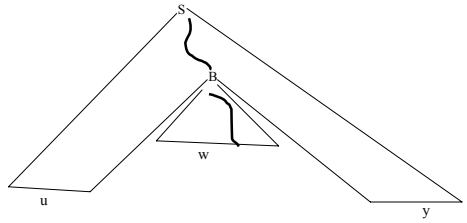
Ableitungsbaum von hinreichend langem Wort

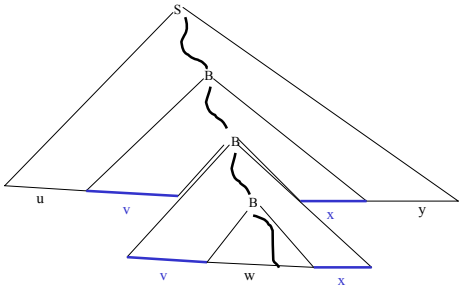


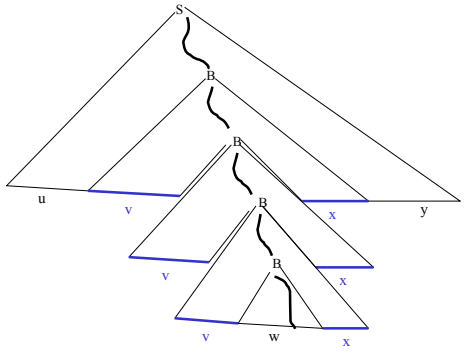
Beweis des Pumping Lemmas für NKA Sprachen:

L NKA-Sprache $\Rightarrow$

$\exists$ N$\in\mathbb{N}$: $\forall$ z$\in$L : $\exists$ Unterteilung z=uvwxy : $\forall$ i$\in\mathbb{N}$: $uv^iwx^iy\in L$
mit $|z|\geq N$ mit $|vwx|\leq N$ und $|vx|>0$







Das Wortproblem für kontextfreie Sprachen

Gegeben kfG G = (Σ,V,S,P) in Chomsky-Normalform und x$\in\Sigma^*$, entscheide deterministisch, ob x$\in$L(G).

$x=x_0x_1\cdots x_{n-1}$ $x[i:j]=x_i\cdots x_{j-1}$ $V[i:j]=\{ A\in V \mid A \Rightarrow_G^* x[i:j] \}$

Idee: Berechne V[0:n] und teste, ob S$\in$V[0:n]

Gegeben kfG G = (Σ,V,S,P) in Chomsky-Normalform und x$\in\Sigma^*$, entscheide deterministisch, ob x$\in$L(G).

$x=x_0x_1\cdots x_{n-1}$ $x[i:j]=x_i\cdots x_{j-1}$ $V[i:j]=\{ A\in V \mid A \Rightarrow_G^* x[i:j] \}$

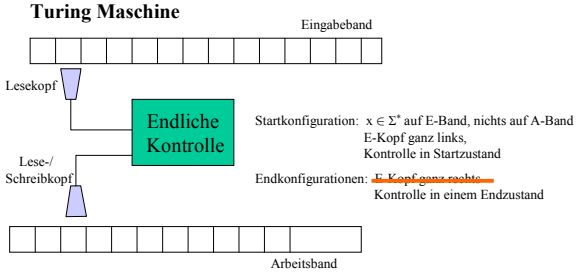
Idee: Berechne V[0:n] und teste, ob S$\in$V[0:n]

for $0\leq i<n$ **do** $V[i:i+1]=\{ A\in V \mid A\rightarrow x_i \in P \}$

for $2\leq m\leq n$ **do**
  for $0\leq i\leq n-m$ **do**
    $V[i:i+m] = \bigcup_{0<j<m} \{ A\in V \mid A\rightarrow BC \text{ und } B\in V[i:i+j], C\in V[i+j:i+m] \}$

Laufzeit $O(n^3)$, wenn n groß und G konstant.

Turing Maschine



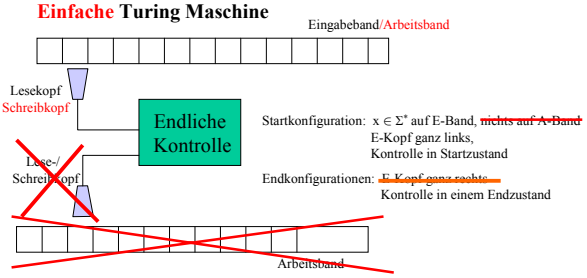
Startkonfiguration: $x \in \Sigma^*$ auf E-Band, nichts auf A-Band
E-Kopf ganz links,
Kontrolle in Startzustand

Endkonfigurationen: ~~E-Kopf ganz rechts~~
Kontrolle in einem Endzustand

Rechenschrittregeln: (endlich viele)

derzeitiger Zustand | gelesenes E-Symbol | gelesenes A-Symbol → neuer Zustand | E-Kopf nach rechts | schreibe A-Symbol | bewege A-Kopf

Einfache Turing Maschine



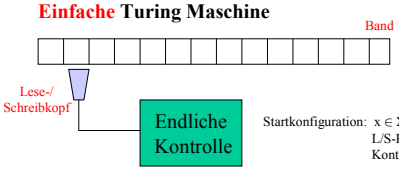
Startkonfiguration: $x \in \Sigma^*$ auf E-Band, ~~nichts auf A-Band~~
E-Kopf ganz links,
Kontrolle in Startzustand

Endkonfigurationen: ~~E-Kopf ganz rechts~~
Kontrolle in einem Endzustand

Rechenschrittregeln: (endlich viele)

derzeitiger Zustand | ~~gelesenes A-Symbol~~ → neuer Zustand | ~~E-Kopf nach rechts~~ | ~~schreibe A-Symbol~~ | ~~bewege A-Kopf~~

Einfache Turing Maschine



Startkonfiguration: $x \in \Sigma^*$ auf Band
L/S-Kopf ganz links,
Kontrolle in Startzustand

Endkonfigurationen: Kontrolle in einem Endzustand

Rechenschrittregeln: (endlich viele)

derzeitiger Zustand | gelesenes Symbol → neuer Zustand | schreibe Symbol | bewege L/S-Kopf

Formale Spezifikation einer einfachen Turing Maschine

$M = (\Sigma,\Gamma,\#,Q,s,F,\Delta)$

Σ	Eingabealphabet
$\Gamma \supset \Sigma$	Bandalphabet
$\# \in \Gamma \setminus \Sigma$	Leerzeichen (Inhalt von nicht initialisierten Zellen)
Q	Zustandsmenge (endlich)
$s \in Q$	Startzustand
$F \subset Q$	Endzustände

$\Delta \subset (Q \times \Gamma) \times (Q \times \Gamma \times \{L,B,R\})$ Rechenschrittregeln

derzeitiger Zustand | derzeitig gelesenes Symbol | neuer Zustand | geschriebenes Symbol | Kopfbewegung L = links R = rechts B = bleibe

Formale Spezifikation einer einfachen Turing Maschine

$M = (\Sigma,\Gamma,\#,Q,s,F,\Delta)$

Konfigurationsraum für M: $K_M = \Gamma^* Q \Gamma^+$

$A_1A_2\cdots A_{i-1}\, q\, A_i\cdots A_n$ bedeutet: Bandinhalt ist $A_1\cdots A_n$
Zustand ist q
S/L-Kopf über A_i

Achtung: nocht nicht benutzte Bandzellen sind in Konfiguration nicht kodiert.

Anfangskonfiguration für $x \in \Sigma^+$: $\text{init}(x) = s\, x$
für $x \in \Sigma^0$: $\text{init}(\varepsilon) = s\, \#$

Endkonfigurationen: $\Phi = \Gamma^* F \Gamma^+$

Formale Spezifikation einer einfachen Turing Maschine

$M = (\Sigma,\Gamma,\#,Q,s,F,\Delta)$

Konfigurationsraum für M: $K_M = \Gamma^* Q \Gamma^+$

$A_1A_2\cdots A_{i-1}\, q\, A_i\cdots A_n$ bedeutet: Bandinhalt ist $A_1\cdots A_n$
Zustand ist q
S/L-Kopf über A_i

Achtung: nocht nicht benutzte Bandzellen sind in Konfiguration nicht kodiert.

Anfangskonfiguration für $x \in \Sigma^+$: $\text{init}(x) = s\, x$
für $x \in \Sigma^0$: $\text{init}(\varepsilon) = s\, \#$

Alternative Endkonfigurationen: $\Phi_e = \#^* F \#^+$ Am Ende Bandinhalt löschen

Rechenschrittrelation $\vdash_M$ auf K_M

$A_1A_2\cdots A_{i-1}\ p\ A_iA_{i+1}\cdots A_n$
$\quad\vdash_M$ g.d.w $(p,A_i,q,C,B)\in\Delta$
$A_1A_2\cdots A_{i-1}\ q\ CA_{i+1}\cdots A_n$

$A_1A_2\cdots A_{i-1}\ p\ A_iA_{i+1}\cdots A_n$
$\quad\vdash_M$ g.d.w $(p,A_i,q,C,L)\in\Delta$
$A_1A_2\cdots\ q\ A_{i-1}CA_{i+1}\cdots A_n$

$A_1A_2\cdots A_{i-1}\ p\ A_iA_{i+1}A_n$
$\quad\vdash_M$ g.d.w $(p,A_i,q,C,R)\in\Delta$
$A_1A_2\cdots A_{i-1}C\ q\ A_{i+1}\cdots A_n$

Rechenschrittrelation $\vdash_M$ auf K_M

(neue Bandzellen werden zum ersten Mal benutzt)

$p\ A_1A_2\cdots A_n$
$\quad\vdash_M$ g.d.w $(p,A_1,q,C,L)\in\Delta$
$q\ \#CA_2\cdots A_n$

$A_1\cdots A_{n-1}\ p\ A_n$
$\quad\vdash_M$ g.d.w $(p,A_n,q,C,R)\in\Delta$
$A_1\cdots A_{n-1}C\ q\ \#$

Rechenrelation $\vdash_M^*$ auf K_M : reflexive, transitive Hülle von $\vdash_M$

TM M akzeptiert Eingabe $x\in\Sigma^*$

g.d.w.

$init(x)\vdash_M^*\phi$ für irgendein $\phi\in\Phi$

$L(M) = \{ x\in\Sigma^* \mid M \text{ akzeptiert } x \}$

Satz: L Sprache
L = L(M) für irgendeine einfache Turingmaschine $M = (\Sigma,\Gamma,\#,Q,s,F,\Delta)$
$\Leftrightarrow$ L = L(G) für irgendeine Grammatik $G = (\Sigma,V,S,P)$

Beweisidee:
Ableitungsfolge entspricht umgekehrter Rechenschrittfolge

$\cdots\ k_i\ \vdash_M\ k_{i+1}\ \vdash_M\ k_{i+2}\ \vdash_M\ \cdots$
$\cdots\ k_i\ {}_G\Leftarrow\ k_{i+1}\ {}_G\Leftarrow\ k_{i+2}\ {}_G\Leftarrow\ \cdots$

Satz: L Sprache
L = L(M) für irgendeine einfache Turingmaschine $M = (\Sigma,\Gamma,\#,Q,s,F,\Delta)$
$\Leftrightarrow$ L = L(G) für irgendeine Grammatik $G = (\Sigma,V,S,P)$

Beweis: "$\Rightarrow$"
Ableitungsfolge entspricht umgekehrter Rechenschrittfolge

neuer Startzustand

$s'\ x\ \vdash_M\ s\ x\ \vdash_M\cdots\vdash_M\ k_i\ \vdash_M\cdots\vdash_M\ X\ f\ Y$

$x\ {}_G\Leftarrow s'\ x\ {}_G\Leftarrow s\ x\ {}_G\Leftarrow\cdots{}_G\Leftarrow k_i\ {}_G\Leftarrow\cdots{}_G\Leftarrow\ X\ f\ Y\quad {}^*{}_G\Leftarrow\ f\ {}_G\Leftarrow\ S$

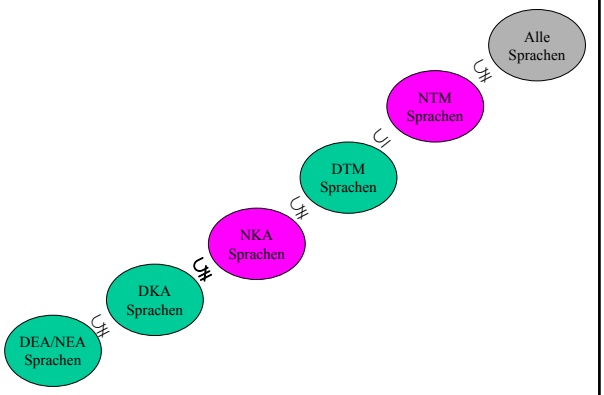
Satz: L Sprache
L = L(M) für irgendeine einfache Turingmaschine $M = (\Sigma,\Gamma,\#,Q,s,F,\Delta)$
$\Leftrightarrow$ L = L(G) für irgendeine Grammatik $G = (\Sigma,V,S,P)$

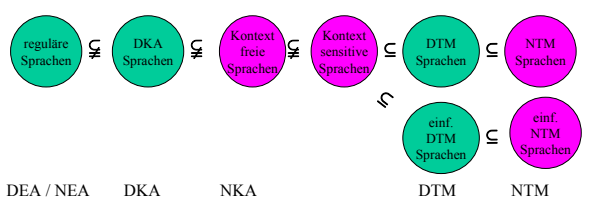
Beweis: "$\Leftarrow$"
Ableitungsfolge entspricht umgekehrter Rechenschrittfolge

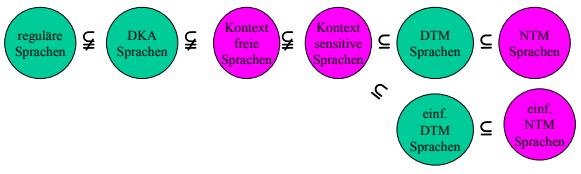
Produktionsanwendungen werden nichtdeterministisch rückgängig gemacht.

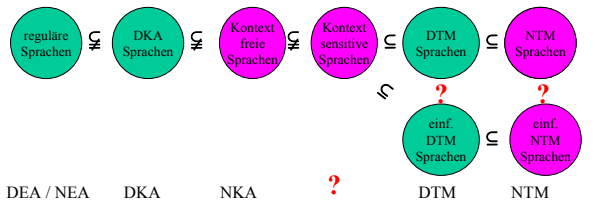
$x\ {}_G\Leftarrow\cdots{}_G\Leftarrow\ U\ {}_G\Leftarrow\quad V\ {}_G\Leftarrow\cdots{}_G\Leftarrow\quad S$

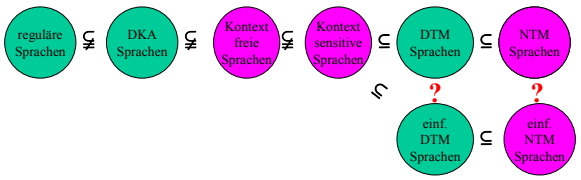
$s\ x\vdash_M s\in x\vdash_M^*\cdots\vdash_M^*\ s\in U\quad\vdash_M^*\ s\in V\ \vdash_M^*\cdots\vdash_M^*\ s\in\#^*S\#^*\vdash_M^*\ f\in\#^*S\#^*$





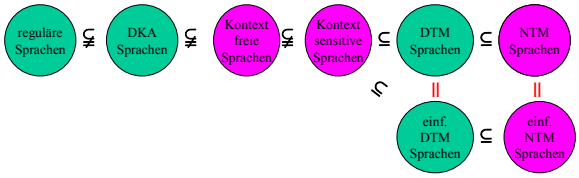






LBA linear beschränkter Automat:
nicht deterministische einfache TM, die nur die Bandzellen der Eingabe verwendet

Satz: Die Sprachen, die durch kontextsensitive Grammatiken generiert werden können, sind genau die Sprachen, die durch LBAs akzeptiert werden können.



DEA / NEA	DKA	NKA	**LBA**	DTM	NTM
linkslineare Grammatik (rechtslin. Gr.)	LR(k) Gr.	kontext freie Gr.	kontext sensitive Gr.	?	Typ 0 Grammatik

reguläre Ausdrücke
Myhill-Nerode

k-Band Turingmaschine:

Eingabeband mit Lesekopf (kein Schreiben)

k Arbeitsbänder, jedes mit einem unabhängigen Lese-/Schreibkopf

$M = (\Sigma, \Gamma, \#, Q, s, F, \Delta)$

Σ	Eingabealphabet
Γ	Arbeitsbandandalphabet
$\#$	Leerzeichen (Inhalt von nicht initialisierten Zellen)
Q	Zustandsmenge (endlich)
$s \in Q$	Startzustand
$F \subset Q$	Endzustände

$\Delta \subset (Q \times \Sigma \times \Gamma^k) \times (Q \times \{L,B,R\} \times (\Gamma \times \{L,B,R\})^k)$ Rechenschrittregeln

Zustand derzeitig | E-Band | A-Bänder | Zustand neu | E-Kopf bewegung | zu schreibendes Symbol | A-Kopf-bewegung

Satz:

L von einf. DTM akzeptiert
⇔ L von DTM akzeptiert
⇔ L von k-Band DTM akzeptiert

Beh 1: L von einf. DTM akzeptiert ⇒ L von DTM akzeptiert

Beh 2: L von DTM akzeptiert ⇒ L von k-Band DTM akzeptiert

Beh 3:
L von k-Band DTM akzeptiert ⇒ L von einf. DTM akzeptiert

Beh 3: L von k-Band DTM akzeptiert ⇒ L von einf. DTM akzeptiert

Beweisidee: Sei M die k-Band DTM, die L akzeptert.

Baue einfache DTM M', die M schrittweise simuliert.

das Eingabeband und die k Arbeitsbänder von M
werden als k+1 „Spuren" auf Band von M' realisiert
(plus Markierungen für simulierte Kopfpositionen)

$\Gamma' = \Sigma \cup (\Gamma \times \{0,1\})^{k+1}$

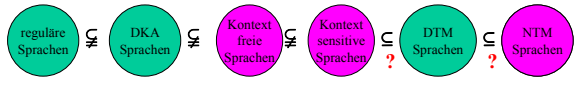
M' geht in 2 Phasen vor:

Phase 0: Wandle Eingabe um und "Formatiere" Band in k+1 Spuren

Phase 1: Simuliere jeden Schritt von M mit Hilfe von je (2k+2)-maligen Überstreichen des gesamten Bandinhalts

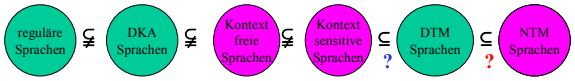
Satz:

L von einf. NTM akzeptiert
⇔ L von NTM akzeptiert
⇔ L von k-Band NTM akzeptiert



DEA / NEA	DKA	NKA	**LBA**	DTM	NTM
linkslineare Grammatik (rechtslin. Gr.)	LR(k) Gr.	kontext freie Gr.	kontext sensitive Gr.	?	Typ 0 Grammatik

reguläre Ausdrücke
Myhill-Nerode



DEA / NEA	DKA	NKA	LBA	DTM	NTM
linkslineare Grammatik (rechtslin. Gr.)	LR(k) Gr.	kontext freie Gr.	kontext sensitive Gr.	?	Typ 0 Grammatik

reguläre Ausdrücke
Myhill-Nerode

Satz:
Die DTM-Sprachen sind genau das Gleiche wie die NTM-Sprachen.

Beweis: Es reicht zu zeigen:

Für jede k-Band NTM $M=(\Sigma,\Gamma,\#,Q,s,F,\Delta)$ gibt es eine (k+1)-Band DTM M', sodass $x\in L(M)$ genau dann wenn $x\in L(M')$.

Idee: Simuliere "alle" möglichen Abläufe von M bei Eingabe x.
Für jede einzelnen Ablauf verwende einen „Leitstring", um die nicht-deterministischen Wahlmöglichkeiten deterministisch zu treffen.

Für „Situation" $S = (q,a,A_1,\cdots,A_k) \in Q\times\Sigma\times\Gamma^k$ sei W_S die Anzahl der anwendbaren Regeln in Δ.

$$W = \max\{ W\_S \mid S \in Q\times\Sigma\times\Gamma^k \}$$

Für jede Situation S, gib den in S anwendbaren Regeln in Δ eine feste Ordnung.

Sei $LS = \{1,\cdots,W\}^*$ die Menge der "Leitstrings"

Verwendung von Leitstring $I = (i_1,i_2,\cdots,i_k) \in LS$ bedeutet:

Simuliere M mit Eingabe x für k Schritte.
Verwende im j-ten Schritt die i_j-te anwendbare Regel
(falls so eine Regel nicht existiert, stop)

Die deterministische TM M' verwendet Band k+1 fürs Speichern des aktuellen Leitstrings.

M' generiert einen Leitstring nach dem anderen auf Band k+1.

Für jeden Leitstring I:
M' verwendet Leitstring I für Eingabe x
Wenn dabei M Eingabe x akzeptiert,
dann akzeptiert M' ebenfalls Eingabe x.
Wenn nicht, dann löscht M' die Arbeitsbänder 1 bis k,
und bewegt Eingabelesekopf zum Eingabeanfang.

DEA / NEA	DKA	NKA	LBA	DTM NTM
linkslineare Grammatik (rechtslin. Gr.)	LR(k) Gr.	kontext freie Gr.	kontext sensitive Gr.	Typ 0 ammatik

reguläre Ausdrücke
Myhill-Nerode

Berechnen von Funktionen durch Turingmaschinen

Erkläre ein Band von det. TM M zum „Ausgabeband".

Wenn M mit Eingabe x stoppt, dann ist der Inhalt dieses Bandes „die Ausgabe von M bei Eingabe x".

Eine partielle Funktion $f : \Sigma^*\to\Gamma^*$ heißt (Turing-)berechenbar, wenn es eine det. TM M gibt, sodass für jedes $x\in\Sigma^*$
M bei Eingabe x mit Ausgabe f(x) hält, falls f(x) definiert,
und M bei Eingabe x nicht hält, falls f(x) nicht definiert.

Eine partielle Funktion $f : \mathbb{N}^k\to\mathbb{N}$ heißt (Turing-)berechenbar, falls die Funktion
$\text{bin}(x_1)\$\text{bin}(x_2)\$\cdots\$\text{bin}(x_k) \mapsto \text{bin}(\ f(x_1,\cdots,x_k)\ )$
(Turing-)berechenbar ist. (bin(x) .. binäre Kodierung von x)

Achtung: f Turing-berechenbar ist etwas anderes als das genaue Verhalten von f zu wissen.

$$f(n) = \begin{cases} 1 & \text{falls in der Dezimaldarstellung von } \pi \text{ irgendwo mindesten } n \text{ Zifferen } 7 \text{ hintereinander vorkommen} \\ 0 & \text{sonst.} \end{cases}$$

Diese Funktion ist auf jeden Fall berechenbar, aber man weiß nicht, was sie ist.

Church-Turing These:

Jede „intuitiv“ berechenbare Funktion ist Turing-berechenbar.

Church-Turing These:

Jede „intuitiv“ berechenbare Funktion ist Turing-berechenbar.

Church-Turing These:

Kann eine Funktion nicht durch eine Turingmaschine berechnet werden, dann ist sie **überhaupt** nicht berechenbar.

Praktisches Konstruieren von TMen aus Teilmaschinen