Grundlagen der Theoretischen Informatik

Ernst-Rüdiger Olderog
Christopher Bischopink

Wintersemester 2019/20

CARL VON OSSIETZKY universität OLDENBURG

Determ. endlicher Automat

Definition 1.1 Ein deterministischer endlicher Automat (Akzeptor), kurz DEA, ist eine Struktur

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

mit folgenden Eigenschaften:

1. Σ ist eine endliche Menge, das Eingabealphabet,
2. Q ist eine endliche Menge von Zuständen,
3. $\delta : Q \times \Sigma \to Q$ ist die Überführungsfunktion,
4. $q_0 \in Q$ ist der Anfangszustand,
5. $F \subseteq Q$ ist die Menge der Endzustände.

DEA mit Transitionsrelation

Definition 1.1 Ein deterministischer endlicher Automat (Akzeptor), kurz DEA, ist eine Struktur

$$\mathcal{A} = (\Sigma, Q, \rightarrow, q_0, F)$$

mit folgenden Eigenschaften:

1. Σ ...
2. Q ...
3. $\rightarrow \subseteq Q \times \Sigma \times Q$

   ist eine deterministische Transitionsrelation, d.h.

   $\forall\, q \in Q\ \forall\, a \in \Sigma\ \exists$ genau ein $q' \in Q :\ (q, a, q') \in \rightarrow$
4. $q_0 \in Q$ ...
5. $F \subseteq Q$ ...

Akzeptanz

Definition 1.2
Sei $\mathcal{A} = (\Sigma, Q, \rightarrow, q_0, F)$ bzw. $\mathcal{A} = (\Sigma, Q, \delta, q_0, F)$ ein DEA.

1. Die von $\mathcal{A}$ akzeptierte (oder erkannte) Sprache ist

   $$L(\mathcal{A}) = \{w \in \Sigma^* \mid \exists\, q \in F : q_0 \stackrel{w}{\rightarrow} q\}$$

   bzw.

   $$L(\mathcal{A}) = \{w \in \Sigma^* \mid \delta^*(q_0, w) \in F\}.$$

   Eine Sprache L heißt endlich akzeptierbar, falls es einen DEA $\mathcal{A}$ mit $L = L(\mathcal{A})$ gibt.

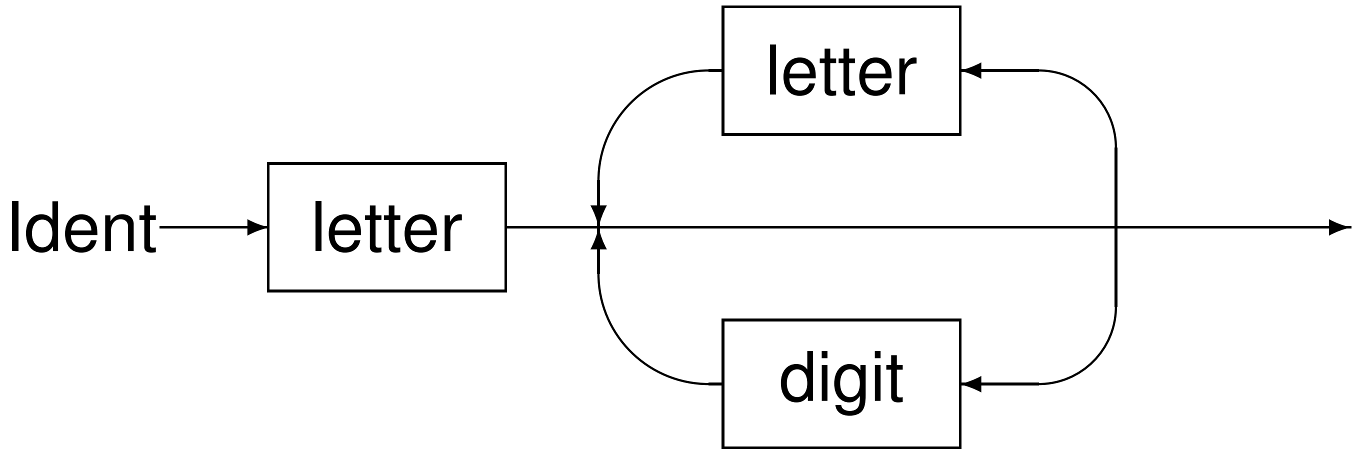
2. Ein Zustand $q \in Q$ heißt in $\mathcal{A}$ erreichbar, falls

   $$\exists w \in \Sigma^* : q_0 \stackrel{w}{\rightarrow} q.$$

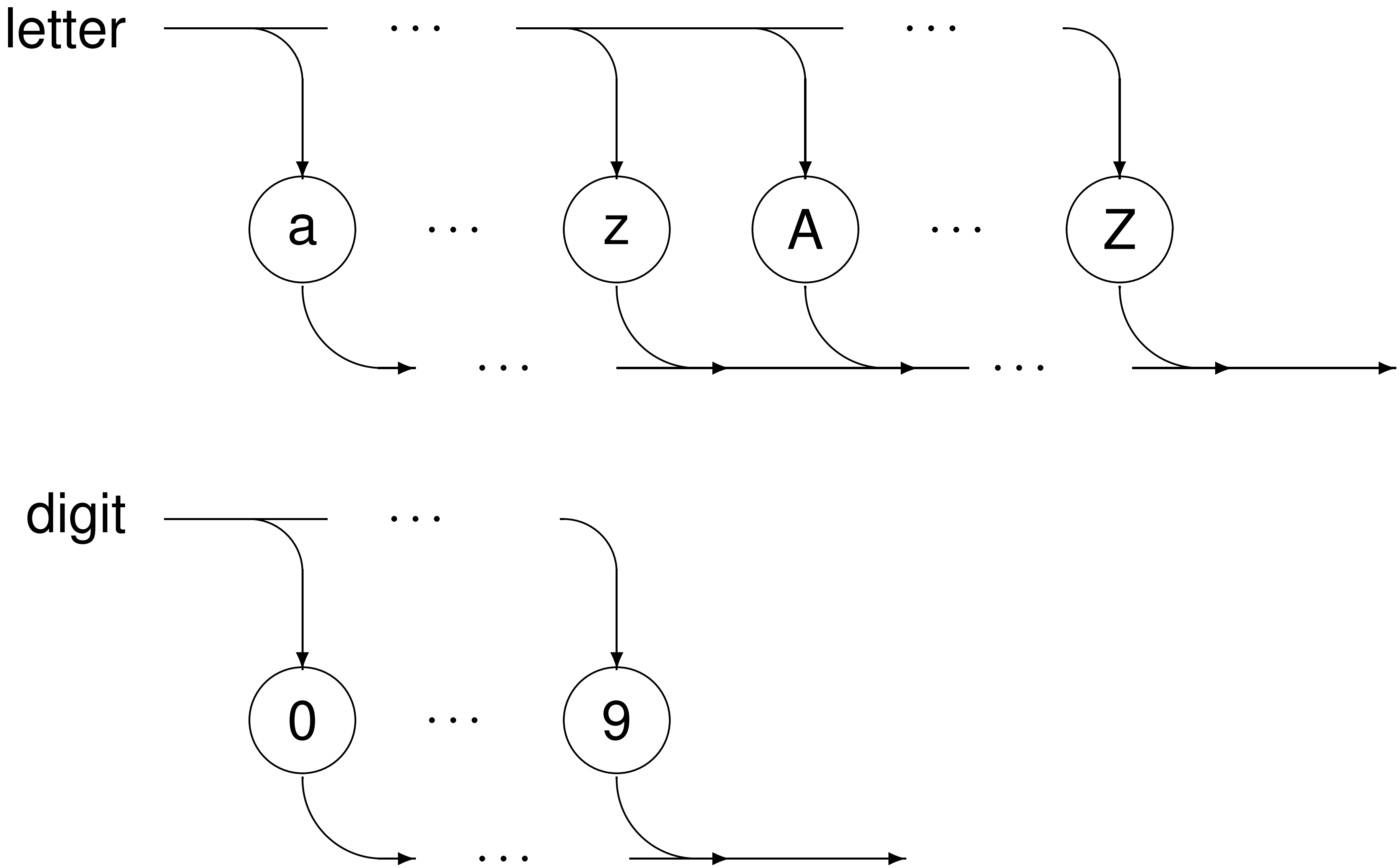
Syntaxdiagramme

der Programmiersprachen PASCAL und MODULA.

Beispiel: Identifikatoren



Syntaxdiagramme



Nichtdet. endlicher Automat

Definition 1.3
Ein nichtdeterministischer endl. Automat (Akzeptor), kurz NEA, ist eine Struktur

$$\mathcal{B} = (\Sigma, Q, \rightarrow, q_0, F)$$

wobei Σ, Q, q_0, F wie bei DEAs definiert sind und für $\rightarrow$ gilt:

$$\rightarrow \subseteq Q \times \Sigma \times Q.$$

Akzeptanz und Äquivalenz

Definition 1.4

(i) Die von einem NEA $\mathcal{B} = (\Sigma, Q, \to, q_0, F)$ akzeptierte (oder erkannte) Sprache ist

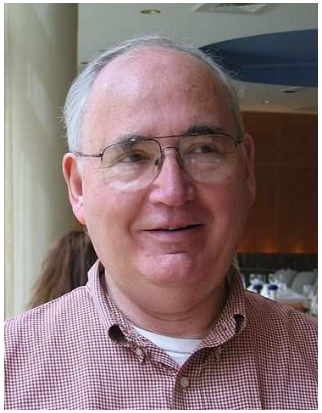
$$L(\mathcal{B}) = \{w \in \Sigma^* \mid \exists\, q \in F : q_0 \xrightarrow{w} q\}.$$

(ii) Zwei NEAs $\mathcal{B}_1$ und $\mathcal{B}_2$ heißen äquivalent, falls

$$L(\mathcal{B}_1) = L(\mathcal{B}_2)$$

gilt.

60 Jahre Satz von Scott und Rabin



Dana S. Scott

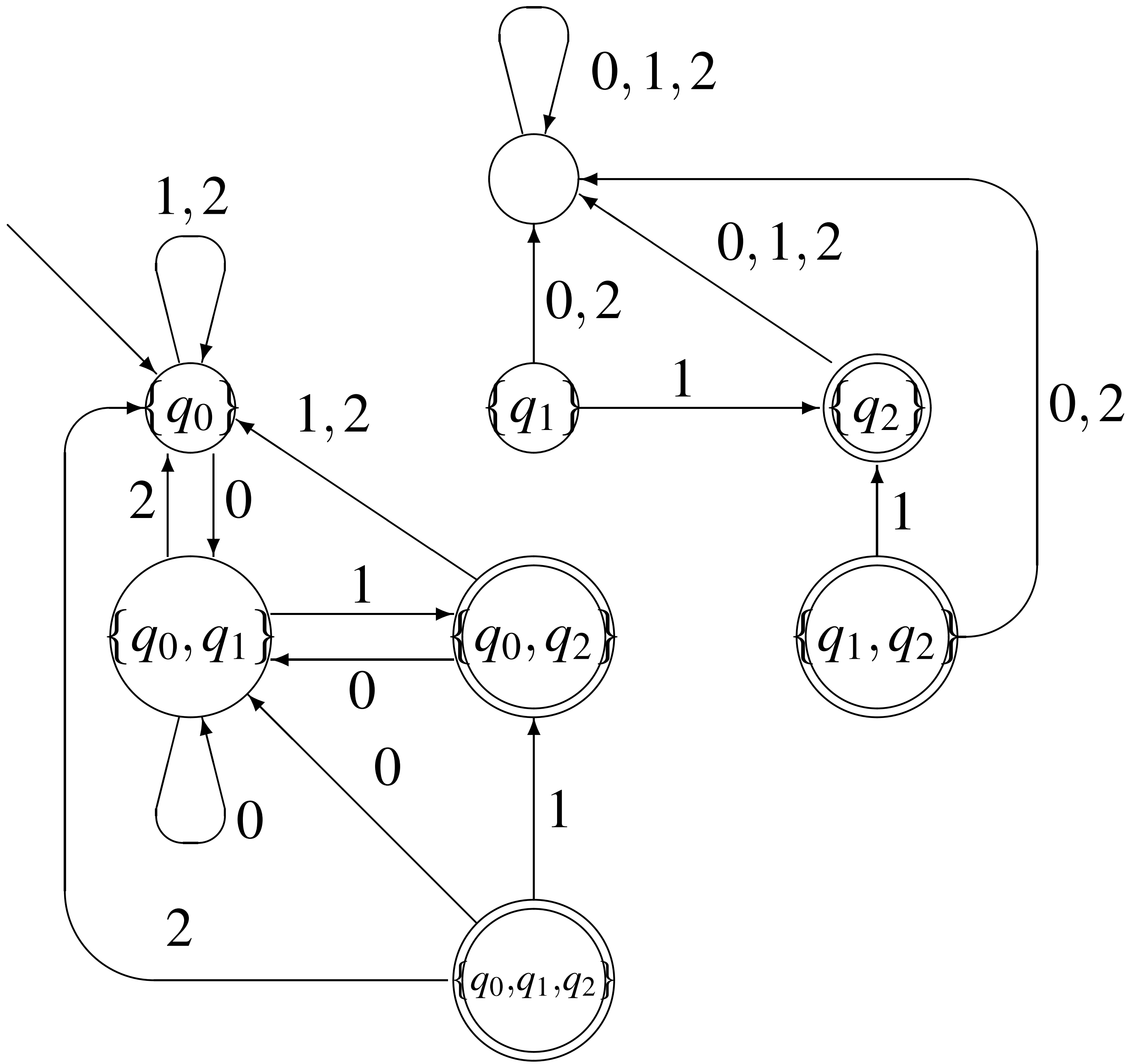
Michael O. Rabin

Satz (Scott & Rabin, 1959)

Zu jedem NEA gibt es
einen äquivalenten DEA.

FOTOS AUS WIKIPEDIA

Potenzmengen-Konstruktion



Spontane Übergänge

Definition 1.6
Ein nichtdet. endl. Automat (Akzeptor) mit ε-Übergängen, kurz ε-NEA, ist eine Struktur

$$\mathcal{B} = (\Sigma, Q, \rightarrow, q_0, F),$$

wobei Σ, Q, q_0, F wie bei NEAs bzw. DEAs definiert sind und für $\rightarrow$ gilt:

$$\rightarrow \;\subseteq\; Q \times (\Sigma \cup \{\varepsilon\}) \times Q.$$

Akzeptanz und Äquivalenz

Definition 1.7
Sei $\mathcal{B} = (\Sigma, Q, \rightarrow, q_0, F)$ ein ε-NEA.

(i) Die von $\mathcal{B}$ akzeptierte (oder erkannte) Sprache ist

$$L(\mathcal{B}) = \{w \in \Sigma^* \mid \exists\, q \in F : \; q_0 \stackrel{w}{\Rightarrow} q\}.$$

(ii) Zwei ε-NEAs $\mathcal{B}_1$ und $\mathcal{B}_2$ heißen äquivalent, falls

$$L(\mathcal{B}_1) = L(\mathcal{B}_2)$$

gilt.

Abschlusseigenschaften

Satz 2.1 Die Klasse der endl. akzeptierbaren Sprachen ist abgeschlossen unter den Operationen

1. Vereinigung,
2. Komplement,
3. Durchschnitt,
4. Differenz,
5. Konkatenation,
6. Iteration.

Reguläre Ausdrücke und Sprachen

Definition 3.1

1. Die Syntax der regulären Ausdrücke über Σ
   ist wie folgt gegeben:

   - $\emptyset$ und ε sind reguläre Ausdrücke über Σ.
   - Für jedes $a \in \Sigma$ ist

     a ein regulärer Ausdruck über Σ.
   - Wenn re, re_1, re_2 reguläre Ausdrücke über Σ sind, so auch

$$(re_1 + re_2),\ (re_1 \cdot re_2),\ re^*.$$

Reguläre Ausdrücke und Sprachen

Definition 3.1

2. Die Semantik eines regulären Ausdrucks re über Σ ist die Sprache $L(re) \subseteq \Sigma^*$, induktiv wie folgt definiert:
   - $L(\emptyset) = \emptyset$
   - $L(\varepsilon) = \{\varepsilon\}$
   - $L(a) = \{a\}$ für $a \in \Sigma$
   - $L((re_1 + re_2)) = L(re_1) \cup L(re_2)$
   - $L((re_1 \cdot re_2)) = L(re_1) \cdot L(re_2)$
   - $L(re^*) = L(re)^*$

3. Eine Sprache $L \subseteq \Sigma^*$ heißt regulär, falls es einen regulären Ausdruck re über Σ gibt mit $L = L(re)$.

Pumping-Lemma

$\forall L \subseteq \Sigma^* :\ L$ regulär $\Rightarrow P_{reg}(L)$, wobei

$$P_{reg}(L) \;\Leftrightarrow\; \exists\, n \in \mathbb{N}\ \forall z \in L \text{ mit } |z| \geq n$$

$$\exists\, u, v, w \in \Sigma^* :$$

$$z = u \cdot v \cdot w$$

$$\wedge\ v \neq \varepsilon$$

$$\wedge\ |uv| \leq n$$

$$\wedge\ \forall i \in \mathbb{N} :\ u \cdot v^i \cdot w \in L$$

Pumping-Lemma: Kontraposition

$\forall L \subseteq \Sigma^* : \neg P_{reg}(L) \Rightarrow \neg L$ regulär, wobei

$$\neg P_{reg}(L) \Leftrightarrow \forall n \in \mathbb{N}\ \exists z \in L \text{ mit } |z| \geq n$$

$$\forall u, v, w \in \Sigma^* :$$

$$(\quad z = u \cdot v \cdot w$$

$$\wedge\ v \neq \varepsilon$$

$$\wedge\ |uv| \leq n\ )$$

$$\Rightarrow \exists i \in \mathbb{N} :\ u \cdot v^i \cdot w \notin L$$

Nerode Rechtskongruenz

Anil Nerode

FOTO: MATH.CORNELL.EDU

Definition

Sei $L \subseteq \Sigma^*$ eine beliebige Sprache.

Die Nerode-Rechtskongruenz von L ist eine zweistellige Relation $\equiv_L$ auf Σ^*,

$$\equiv_L \ \subseteq \ \Sigma^* \times \Sigma^*,$$

die für $u, v \in \Sigma^*$ so definiert ist:

$u \equiv_L v$ genau dann, wenn für alle $w \in \Sigma^*$ gilt:

$$uw \in L \Leftrightarrow vw \in L$$

Algorithmische Konstruktionen

- ε-NEA $\rightarrow$ NEA $\rightarrow$ DEA
- DEA $\rightarrow$ Minimalautomat
- ε-NEAs für folgende Operationen auf endl. akzeptierbaren Sprachen:

$$L_1 \cup L_2, \quad \overline{L}, \quad L_1 \cap L_2, \quad L_1 \backslash L_2, \quad L_1 \cdot L_2, \quad L^*$$

- regulärer Ausdruck $\rightarrow$ NEA $\rightarrow$ DEA
  DEA $\rightarrow$ regulärer Ausdruck

Entscheidbarkeitsfragen

Wortproblem	Gegeben:	DEA $\mathcal{A}$ und Wort w
	Frage:	Gilt $w \in L(\mathcal{A})$?
Leerheitsproblem	Gegeben:	DEA $\mathcal{A}$
	Frage:	Gilt $L(\mathcal{A}) = \varnothing$?
Endlichkeitsproblem	Gegeben:	DEA $\mathcal{A}$
	Frage:	Ist $L(\mathcal{A})$ endlich ?
Äquivalenzproblem	Gegeben:	DEAs $\mathcal{A}_1$ und $\mathcal{A}_2$
	Frage:	Gilt $L(\mathcal{A}_1) = L(\mathcal{A}_2)$?
Inklusionsproblem	Gegeben:	DEAs $\mathcal{A}_1$ und $\mathcal{A}_2$
	Frage:	Gilt $L(\mathcal{A}_1) \subseteq L(\mathcal{A}_2)$?
Schnittproblem	Gegeben:	DEAs $\mathcal{A}_1$ und $\mathcal{A}_2$
	Frage:	Gilt $L(\mathcal{A}_1) \cap L(\mathcal{A}_2) = \varnothing$?

Kontextfreie Grammatik

Eine kontextfreie Grammatik ist eine Struktur $G = (N, T, P, S)$ mit

(i) $A, B, C, \cdots \in N$: Nichtterminalsymbole,

(ii) $a, b, c, \cdots \in T$: Terminalsymbole mit $N \cap T = \varnothing$,

(iii) $S \in N$: Startsymbol,

(iv) $P \subseteq N \times (N \cup T)^*$: Produktionen (Regeln)

$(A, v) \in P$ oder kurz $A \rightarrow v$

wobei $u, v, w, \cdots \in (N \cup T)^*$.

Kontextfreie Sprache

Die von einer kfr Grammatik G erzeugte Sprache ist

$$L(G) = \{ \, w \in T^* \mid S \vdash_G^* w \, \}.$$

Zwei kfr Grammatiken G_1 und G_2 heißen äquivalent, falls

$$L(G_1) = L(G_2).$$

Eine Sprache $L \subseteq T^*$ heißt kontextfrei, falls es eine kfr Grammatik G mit $L = L(G)$ gibt.

Ableitungsbaum

Ein Ableitungsbaum von A nach w in G ist ein Baum mit:

(i) Jeder Knoten ist mit einem Symbol aus $N \cup T \cup \{\varepsilon\}$ beschriftet. Die Wurzel ist mit A und jeder innere Knoten ist mit einem Symbol aus N beschriftet.

(ii) Wenn ein mit B beschrifteter innerer Knoten k Nachfolgeknoten besitzt, die von links nach rechts mit den Symbolen $\beta_1, \ldots, \beta_k$ beschriftet sind, dann gilt

a) $k = 1$ und $\beta_1 = \varepsilon$ und $B \to \varepsilon \in P$
oder

b) $k \geq 1$ und $\beta_1, \ldots, \beta_k \in N \cup T$ und $B \to \beta_1 \ldots \beta_k \in P$.

(iii) Das Wort w entsteht, indem man die Symbole an den Blättern von links nach rechts konkateniert.

Eindeutigkeit

(i) Eine kfr Grammatik $G = (N, T, P, S)$ heißt eindeutig, wenn es zu jedem Wort $w \in T^*$

höchstens einen Ableitungsbaum bzw.
höchstens eine Linksableitung von S nach w in G gibt.

Andernfalls heißt G mehrdeutig.

(ii) Eine kfr Sprache $L \subseteq T^*$ heißt eindeutig, wenn es eine eindeutige kfr Grammatik G mit $L = L(G)$ gibt.
Andernfalls heißt L (inhärent) mehrdeutig.

Pumping-Lemma: kfr Sprachen

$\forall L \subseteq \Sigma^* :\ L$ kontextfrei $\Rightarrow P_{kfr}(L),$ wobei

$$P_{kfr}(L) \Leftrightarrow \exists\, n \in \mathbb{N}\ \forall z \in L \text{ mit } |\,z\,| \geq n$$

$$\exists\, u, v, w, x, y \in \Sigma^* :$$

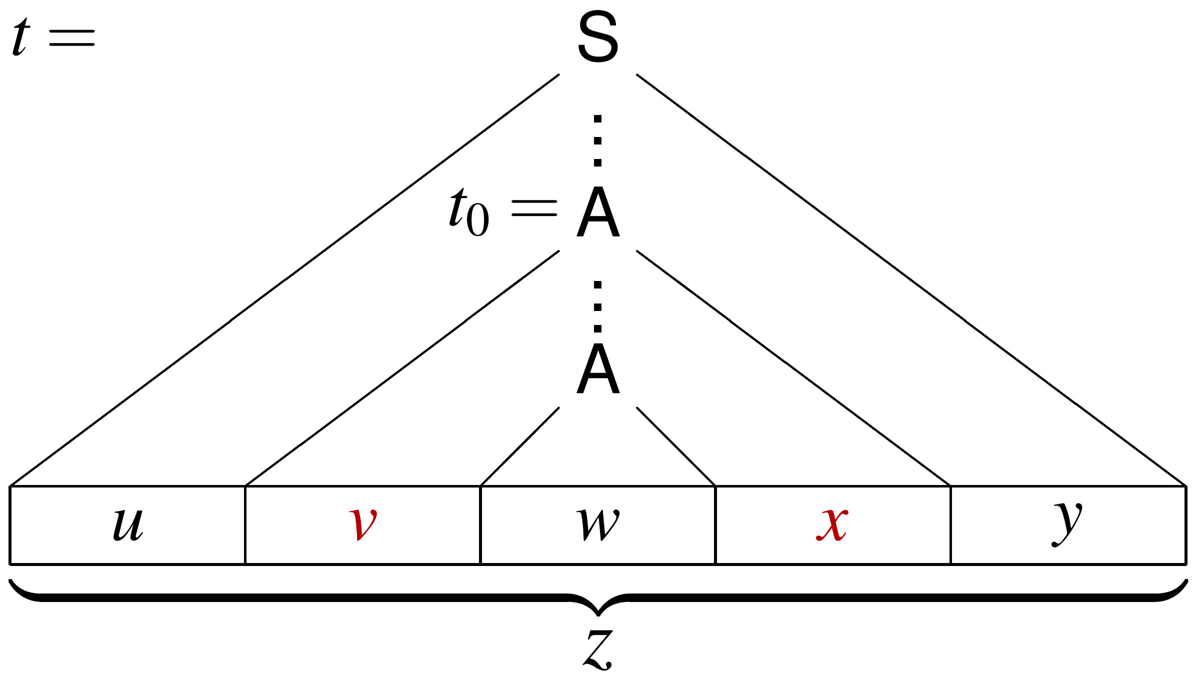
$$z = u \cdot v \cdot w \cdot x \cdot y$$

$$\wedge\ vx \neq \varepsilon$$

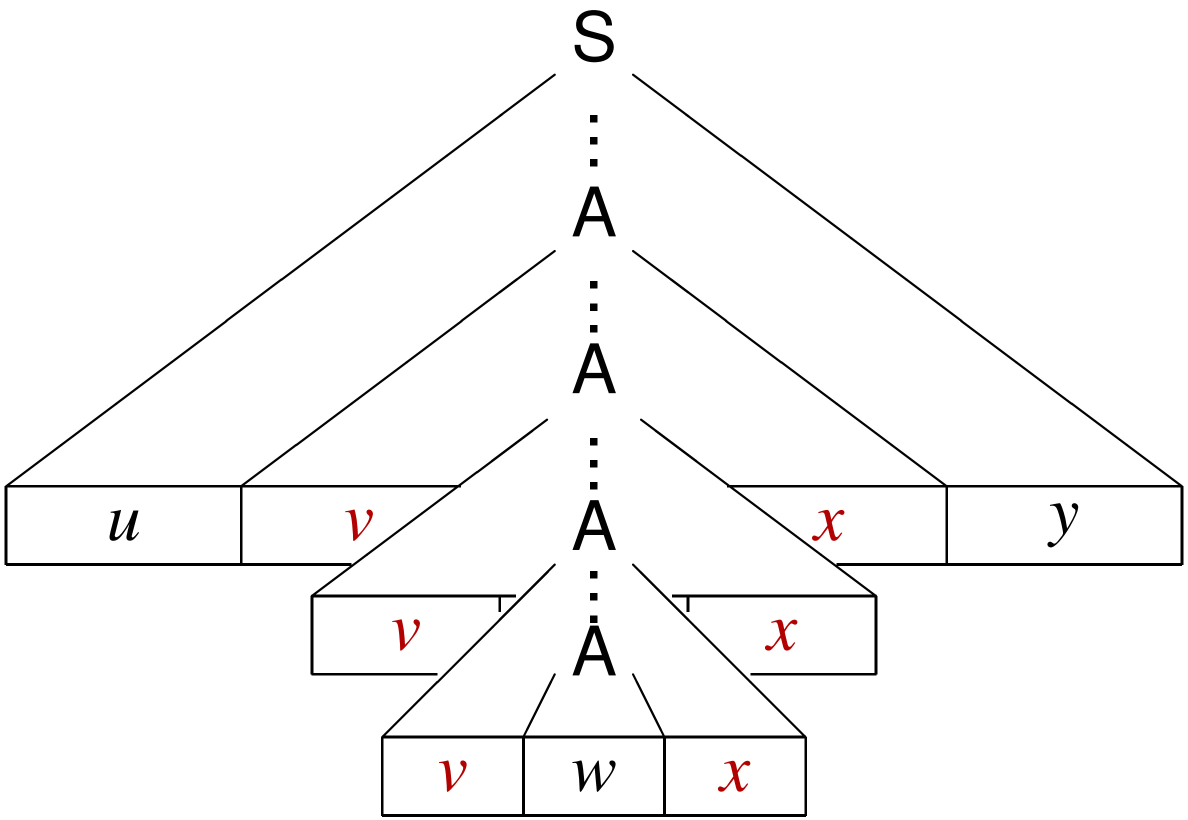
$$\wedge\ |\,vwx\,| \leq n$$

$$\wedge\ \forall i \in \mathbb{N} :\ u \cdot v^i \cdot w \cdot x^i \cdot y \in L$$

Zerlegung: $z = u \cdot v \cdot w \cdot x \cdot y$



Aufpumpen für $i = 3$



kfr Pumping-Lemma: Kontraposition

$\forall L \subseteq \Sigma^* : \; \neg P_{kfr}(L) \;\Rightarrow \neg L$ kontextfrei, wobei

$$\neg P_{kfr}(L) \;\Leftrightarrow\; \forall n \in \mathbb{N} \;\exists z \in L \text{ mit } |\,z\,| \geq n$$

$$\forall u, v, w, x, y \in \Sigma^* :$$

$$(\quad z = u \cdot v \cdot w \cdot x \cdot y$$

$$\wedge \; vx \neq \varepsilon$$

$$\wedge \; |\,vwx\,| \leq n \;)$$

$$\Rightarrow \exists i \in \mathbb{N} : \; u \cdot v^i \cdot w \cdot x^i \cdot y \notin L$$

Java ist nicht kontextfrei

Betrachte $n \in \mathbb{N}$ folgende Java-Klasse:

$$
\begin{array}{l}
\texttt{class}\ C\ \{ \\
\quad \texttt{int}\ a\underbrace{1\ldots1}_{n}; \\
\quad \texttt{void}\ m()\ \{ \\
\qquad a\underbrace{1\ldots1}_{n} = a\underbrace{1\ldots1}_{n} \\
\quad \} \\
\}
\end{array}
$$

Kellerautomat

Ein (nichtdeterministischer) Kellerautomat (Pushdown-Automat), kurz KA (oder PDA), ist eine Struktur

$$\mathcal{K} = (\Sigma, Q, \Gamma, \rightarrow, q_0, Z_0, F)$$

mit folgenden Eigenschaften:

(i) Σ ist das Eingabealphabet,

(ii) Q ist eine endliche Menge von Zuständen,

(iii) Γ ist das Kelleralphabet,

(iv) $\rightarrow \subseteq Q \times \Gamma \times (\Sigma \cup \{\varepsilon\}) \times Q \times \Gamma^*$ ist die Transitionsrelation,

(v) $q_0 \in Q$ ist der Anfangszustand,

(vi) $Z_0 \in \Gamma$ ist das Startsymbol des Kellers,

(vii) $F \subseteq Q$ ist die Menge der Endzustände.

Akzeptanz

Sei $\mathcal{K} = (\Sigma, Q, \Gamma, \to, q_0, Z_0, F)$ ein KA und $w \in \Sigma^*$.

(i) $\mathcal{K}$ akzeptiert w, falls

$$\exists q \in F \quad \exists \gamma \in \Gamma^* : \quad (q_0, Z_0) \overset{w}{\Longrightarrow} (q, \gamma).$$

Die von $\mathcal{K}$ akzeptierte (oder erkannte) Sprache ist

$$L(\mathcal{K}) = \{w \in \Sigma^* \mid \mathcal{K} \text{ akzeptiert } w\}.$$

(ii) $\mathcal{K}$ akzeptiert w mit leeren Keller, falls

$$\exists q \in Q : \quad (q_0, Z_0) \overset{w}{\Longrightarrow} (q, \varepsilon).$$

Die von $\mathcal{K}$ mit leerem Keller akzeptierte (oder erkannte) Sprache ist

$$L_\varepsilon(\mathcal{K}) = \{w \in \Sigma^* \mid \mathcal{K} \text{ akzeptiert } w \text{ mit leerem Keller } \}.$$

Deterministischer Kellerautomat

Ein KA $\mathcal{K} = (\Sigma, Q, \Gamma, \rightarrow, q_0, Z_0, F)$ heißt deterministisch, falls für die Transitionsrelation $\rightarrow$ gilt:

$\forall q \in Q,\ Z \in \Gamma,\ a \in \Sigma$:

(Anzahl der Transitionen der Form $(q, Z) \xrightarrow{a} \cdots$

\+ Anzahl der Transitionen der Form $(q, Z) \xrightarrow{\varepsilon} \cdots$) ≤ 1

Äquivalente Akzeptanzen

Satz 3.6

(1) Zu jedem KA $\mathcal{A}$ kann ein KA B mit

$$L(\mathcal{A}) = L_{\varepsilon}(B)$$

konstruiert werden.

(2) Zu jedem KA $\mathcal{A}$ kann ein KA B mit

$$L_{\varepsilon}(\mathcal{A}) = L(B)$$

konstruiert werden.

Kfr Gramm $\mapsto$ KA

Satz 3.7

Zu jeder kontextfreien Grammatik G
kann ein Kellerautomat $\mathcal{K}$
mit $L_\varepsilon(\mathcal{K}) = L(G)$ konstruiert werden.

KA $\mapsto$ kfr Gramm

Satz 3.8

Zu jedem Kellerautomaten $\mathcal{K}$
kann eine kontextfreie Grammatik G
mit $L(G) = L_\varepsilon(\mathcal{K})$ konstruiert werden.

Abschlusseigenschaften

Satz 4.1 Die Klasse der kontextfreien Sprachen ist abgeschlossen unter den Operationen

(i) Vereinigung,

(ii) Konkatenation,

(iii) Iteration,

(iv) Durchschnitt mit regulären Sprachen.

Dagegen ist diese Klasse nicht abgeschlossen unter den Operationen

(v) Durchschnitt,

(vi) Komplement.

Deterministische kfr Sprachen

(i) *Wdh*: Ein Kellerautomat $\mathcal{K} = (\Sigma, Q, \Gamma, \to, q_0, Z_0, F)$ heißt deterministisch, falls für die Transitionsrelation $\to$ gilt:

$\forall\, q \in Q,\ Z \in \Gamma,\ a \in \Sigma$:

(Anzahl der Transitionen der Form $(q,Z) \xrightarrow{a} \cdots$
\+ Anzahl der Transitionen der Form $(q,Z) \xrightarrow{\varepsilon} \cdots$) ≤ 1

(ii) Eine kontextfreie Sprache L heißt deterministisch, falls es einen determ. Kellerautomaten $\mathcal{K}$ mit

$$L = L(\mathcal{K})$$

(Akzeptanz mit Endzuständen) gibt.

Abschlusseigenschaften

Sprachklasse	$\cup$	$\cap$	$\overline{L}$	$\cdot$	L^*	$\cap Reg$
regulär	✓	✓	✓	✓	✓	✓
kfr	✓	—	—	✓	✓	✓
determ. kfr	—	—	✓	—	—	✓

Normalformen von kfr Gramm

Eine ε-Produktion ist eine Regel der Form $A \to \varepsilon$.

Definition. Eine kfr Grammatik $G = (N, T, P, S)$ heißt ε-frei, falls es in G

(i) *entweder* überhaupt keine ε-Produktion gibt

(ii) *oder* nur die ε-Produktion $S \to \varepsilon$ gibt
und S dann nicht auf der rechten Seite
irgendeiner Produktion in G auftritt.

Chomsky-Normalform

Definition. Eine kfr Grammatik $G = (N, T, P, S)$ ist in Chomsky-Normalform, wenn Folgendes gilt:

- G ist ε-frei (also höchstens $S \to \varepsilon \in P$),
- jede Produktion in P anders als $S \to \varepsilon$ hat die Form

  $$A \to a \quad \text{oder} \quad A \to BC,$$

  wobei $A, B, C \in N$ und $a \in T$ sind.

Greibach-Normalform

Definition. Eine kfr Grammatik $G = (N, T, P, S)$ ist in Greibach-Normalform, wenn Folgendes gilt:

- G ist ε-frei (also höchstens $S \to \varepsilon \in P$),
- jede Produktion in P anders als $S \to \varepsilon$ hat die Form

  $$A \to a B_1 \ldots B_k,$$

  wobei $k \geq 0$, $A, B_1, \ldots, B_k \in N$ und $a \in T$ sind.

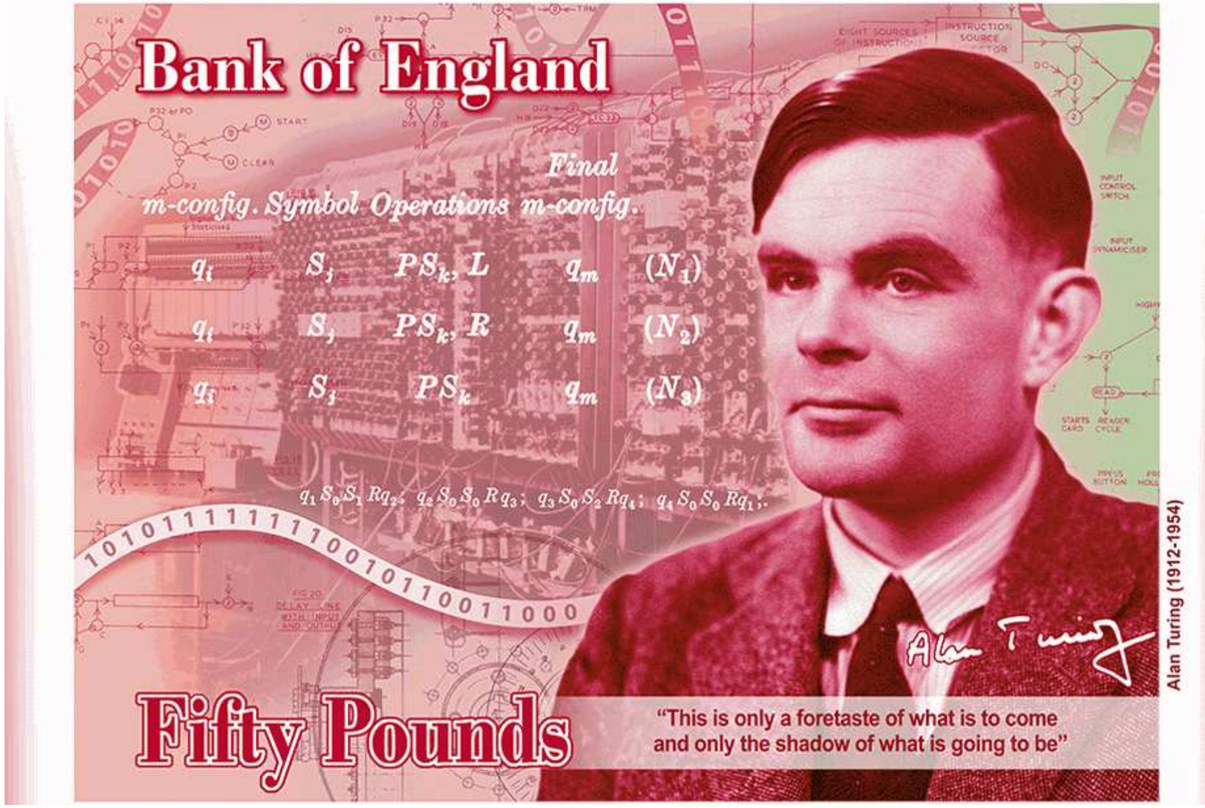
Entscheidbarkeitsfragen

Wortproblem	Gegeben:	kfr. Gramm. G und Wort w
	Frage:	Gilt $w \in L(G)$?
Leerheitsproblem	Gegeben:	kfr. Gramm. G
	Frage:	Gilt $L(G) = \varnothing$?
Endlichkeitsproblem	Gegeben:	kfr. Gramm. G
	Frage:	Ist $L(G$ endlich ?
Äquivalenzproblem	Gegeben:	kfr. Gramm. G_1 und G_2
	Frage:	Gilt $L(G_1) = L(G_2)$?
Inklusionsproblem	Gegeben:	kfr. Gramm. G_1 und G_2
	Frage:	Gilt $L(G_1) \subseteq L(G_2)$?
Schnittproblem	Gegeben:	kfr. Gramm. G_1 und G_2
	Frage:	Gilt $L(G_1) \cap L(G_2) = \varnothing$?

Entscheidbarkeitsresultate

Sprachklasse	Wort	Leer	Endl	Äquiv	Inkl	Schnitt$= \emptyset$
regulär	✓	✓	✓	✓	✓	✓
kfr	✓	✓	✓	–	–	–

Alan M. Turing (1912–1954)



Neue 50 Pounds Bank Note, die Ende 2021 in Umlauf geht.

BILD: BANK OF ENGLAND

A Turing Machine in Action

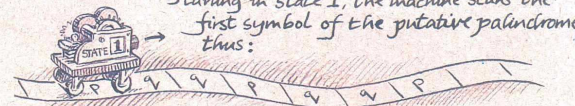
Alan Turing devised his famous machine in 1936 to explore the scope of mechanical processes in logic and mathematics. To what extent, he was asking himself, could a mathematician be replaced by a machine? Using the machine as a "thought experiment," Turing proved that there is no way of finding out whether any given mathematical statement can be proved or not - a formidable achievement which earned the machine a place in the history of mathematics.

Here we illustrate a far less ambitious task than a test for provability. We see how the machine tests whether a sequence is or is not a palindrome - a string of symbols that reads the same backwards as fowards.

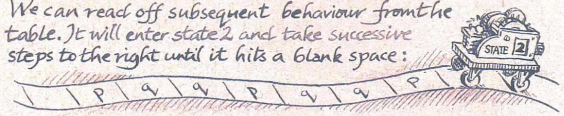
The machine starts in state 1 and scans the first symbol of the putative palindrome. At each step the machine takes its instructions from the table shown below. Turing talked of the machine being in a particular "state" which changes as it progresses with its task. The state at any one time tells the machine what it has to do, and what state to go into next, in effect the state defines what is in the machine's "mind". The "scanner" moves back and forth erasing the first letter and the last letter each time and checking that they are the same. If they are the same, then the sequence is, by definition a palindrome. The machine continues until nothing is left and then goes into state 9, which is a "halt".

Symbol scanned / State of machine	p	q	blank
1	move one step right enter state 2	right state 3	print Y right state 9
2	right state 2	right state 2	left state 4
3	right state 3	right state 3	left state 5
4	erase left state 6	right state 8	
5	right state 8	erase left state 6	
6	left state 6	left state 6	right state 7
7	erase right state 1	erase right state 1	print Y right state 9
8			print N right state 9
9		.	no move state 9

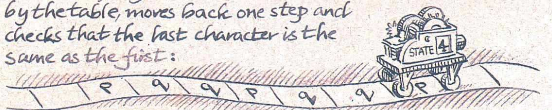
Starting in state 1, the machine scans the first symbol of the putative palindrome thus:



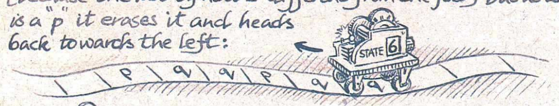
We can read off subsequent behaviour from the table. It will enter state 2 and take successive steps to the right until it hits a blank space:



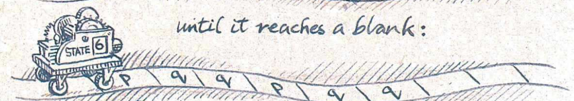
Then it stops, goes into state 4 as instructed by the table, moves back one step and checks that the last character is the same as the first:



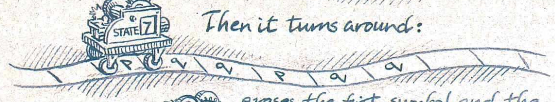
If it had found a "q" in this position it would have printed N, indicating that the sequence cannot be a palindrome (because the last symbol is different from the first) but as it is a "p" it erases it and heads back towards the left:



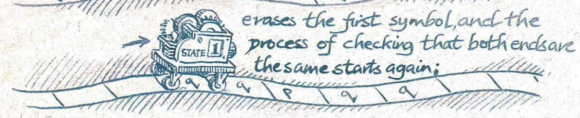
until it reaches a blank:



Then it turns around:

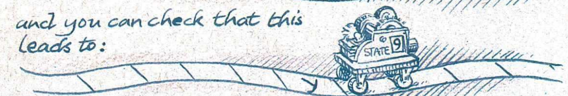


erases the first symbol, and the process of checking that both ends are the same starts again;



Then the process eventually reduces the palindrome to:

and you can check that this leads to:



where it stops and marks time forever.
You can check that the machine also works if the palindrome has an even number of letters.

QUELLE: NEW SCIENTIST, VOL.99, NO.1375, 15 SEPT.1983

Turingmaschine

Eine Turingmaschine, kurz TM, ist eine Struktur

$\tau = (Q, \Sigma, \Gamma, \delta, q_0, \sqcup)$ mit folgenden Eigenschaften:

(i) Q ist endl. nichtleere Menge von Zuständen,

(ii) $q_0 \in Q$ ist der Anfangszustand,

(iii) Γ ist endl. nichtleere Menge, das Bandalphabet, mit $Q \cap \Gamma = \varnothing$,

(iv) $\Sigma \subseteq \Gamma$ ist das Eingabealphabet,

(v) $\sqcup \in \Gamma - \Sigma$ ist das Leerzeichen oder Blank,

(vi) $\delta : Q \times \Gamma \xrightarrow{part} Q \times \Gamma \times \{R, L, S\}$

ist die Überführungsfunktion.

Endzustände

Eine Turingmaschine mit Endzuständen ist eine Struktur $\tau = (Q, \Sigma, \Gamma, \delta, q_0, \sqcup, F)$ mit folgenden Eigenschaften:

(i) Q ist endliche, nichtleere Menge von Zuständen,

(ii) $q_0 \in Q$ ist der Anfangszustand,

(iii) Γ ist endliche nichtleere Menge, das Bandalphabet,
mit $Q \cap \Gamma = \varnothing$,

(iv) $\Sigma \subseteq \Gamma$ ist das Eingabealphabet,

(v) $\sqcup \in \Gamma - \Sigma$ ist das Leerzeichen oder Blank,

(vi) $\delta : Q \times \Gamma \xrightarrow{part} Q \times \Gamma \times \{R, L, S\}$
ist die Überführungsfunktion.

(vii) $F \subseteq Q$ eine Menge von Endzuständen ist.

Transitionsrelation $\vdash_\tau \subseteq \mathcal{K}_\tau \times \mathcal{K}_\tau$

Es gilt

$$K \vdash_\tau K',$$

falls $\exists u, v \in \Gamma^* \; \exists a, b \in \Gamma \; \exists q, q' \in Q :$

$$
\begin{aligned}
& (K = uqav \wedge \delta(q,a) = (q',a',S) \wedge K' = uq'a'v) \\
\vee\; & (K = uqabv \wedge \delta(q,a) = (q',a',R) \wedge K' = ua'q'bv) \\
\vee\; & (K = uqa \wedge \delta(q,a) = (q',a',R) \wedge K' = ua'q'\sqcup) \\
\vee\; & (K = ubqav \wedge \delta(q,a) = (q',a',L) \wedge K' = uq'ba'v) \\
\vee\; & (K = qav \wedge \delta(q,a) = (q',a',L) \wedge K' = q'\sqcup a'v)
\end{aligned}
$$

Berechnete Funktion

Die von TM $\tau = (Q, \Sigma, \Gamma, \delta, q_0, \sqcup)$ berechnete Funktion ist

$$h_\tau : \quad \Sigma^* \xrightarrow{part} \Gamma^*$$

mit

$$h_\tau(v) = \begin{cases} w & \text{falls} \quad \exists \text{ Endkonfiguration } K \in \mathcal{K}_\tau : \\ & \qquad\quad \alpha(v) \vdash_\tau{}^* K \ \wedge \ w = \omega(K) \\ \text{undef.} & \text{sonst.} \end{cases}$$

Es wird auch Res_τ für h_τ geschrieben (Resultatsfunktion).

Berechnungsbeispiel

Berechnung von gerade

$$
\begin{tikzcd}
{||||} \arrow[r, "\alpha"] \arrow[dddddd, "\mathit{gerade} = h_\tau"'] & \alpha(||||) = q_0\,|||| \arrow[d, phantom, "\top_\tau"] \\
 & \sqcup\, q_1\, ||| \arrow[d, phantom, "\top_\tau"] \\
 & \sqcup\sqcup\, q_0\, || \arrow[d, phantom, "\top_\tau"] \\
 & \sqcup\sqcup\sqcup\, q_1\, | \arrow[d, phantom, "\top_\tau"] \\
 & \sqcup\sqcup\sqcup\sqcup\, q_0\, \sqcup \arrow[d, phantom, "\top_\tau"] \\
 & \\
J & K_{fin} = \sqcup\sqcup\sqcup\sqcup\, q_e\, J \arrow[l, "\omega"']
\end{tikzcd}
$$

Halte- und Ergebnisbereich

Betrachte die von τ berechnete Funktion

$$h_\tau : \quad \Sigma^* \xrightarrow{part} \Gamma^*.$$

Eine Menge $M \in \Sigma^*$ heißt
Haltebereich oder Definitionsbereich von τ, falls gilt:

$$M = \{v \in \Sigma^* \mid h_\tau(v) \text{ ist definiert .}\}$$

Eine Menge $N \in \Gamma^*$ heißt
Ergebnisbereich oder Wertebereich von τ, falls

$$N = \{w \in \Gamma^* \mid \exists v \in \Sigma^* : h_\tau(v) = w\}.$$

Turing-Berechenbarkeit

Es seien A, B Alphabete.

(i) Eine partiell definierte Funktion

$$h : A^* \xrightarrow{part} B^*$$

heißt Turing-berechenbar, falls es eine TM $\tau = (Q, \Sigma, \Gamma, \delta, q_0, \sqcup)$ gibt mit $A = \Sigma$, $B \subseteq \Gamma$ und

$$h = h_\tau,$$

d.h. $h(v) = h_\tau(v)$ für alle $v \in A^*$.

(ii) $\mathcal{T}_{A,B} =_{def} \{h : A^* \xrightarrow{part} B^* \mid h$ ist Turing-berechenbar $\}$

(iii) $\mathcal{T}$ sei die Klasse aller Turing-berechenbaren Funktionen (für beliebige Alphabete A, B).

Turing-Entscheidbarkeit

Es sei A ein Alphabet.

(i) Eine Menge $L \subseteq A^*$ heißt Turing-entscheidbar, falls die charakteristische Funktion von L

$$\chi_L : \quad A^* \longrightarrow \{0,1\}$$

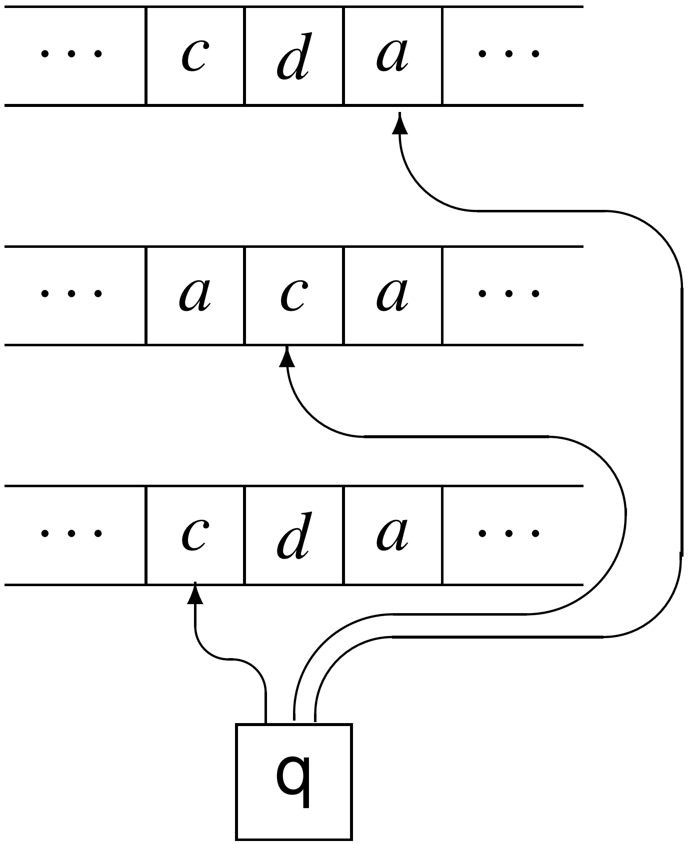
Turing-berechenbar ist.

(ii) Eine Eigenschaft $E : A^* \longrightarrow \{\, wahr, falsch \,\}$ heißt Turing-entscheidbar, falls die Menge

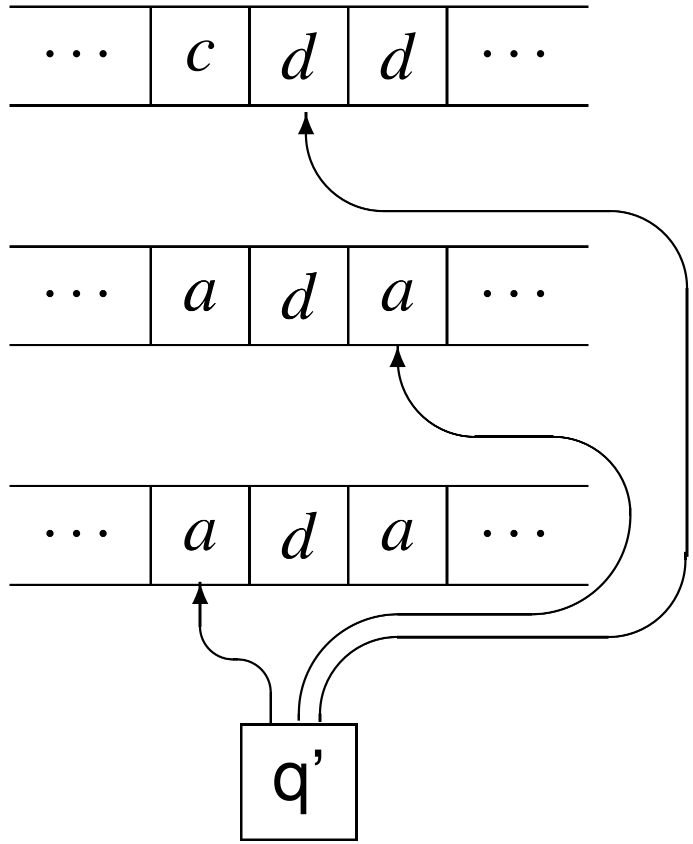
$$\{v \in A^* \mid E(v) = wahr\,\}$$

der Wörter mit Eigenschaft E Turing-entscheidbar ist.

3-Band Turingmaschine



$\vdash_\tau$



Simulation von k-Band TM durch 1-Band TM

k-Band Konfig.		1-Band Konfig.
$K = (u_1 q a_1 v_1,$ $\ldots\ldots$ $u_k q a_k v_k)$	$\longmapsto$	$sim(K) =$ $u_1 q \tilde{a}_1 v_1 \# \ldots \# u_k \tilde{a}_k v_k$

Akzeptanz mittels TM

- Turingmaschine mit Endzuständen $F \subseteq Q$:

$$\tau = (Q, \Sigma, \Gamma, \delta, q_0, \sqcup, F)$$

- $K = uqv$ heißt akzeptierend, falls $q \in F$ ist.

- τ akzeptiert $w \in \Sigma^*$, falls
  $\exists$ akzeptierende Endkonfiguration K: $\alpha(w) \vdash_\tau{}^* K$

Akzeptanz mittels TM

- Turingmaschine mit Endzuständen $F \subseteq Q$:

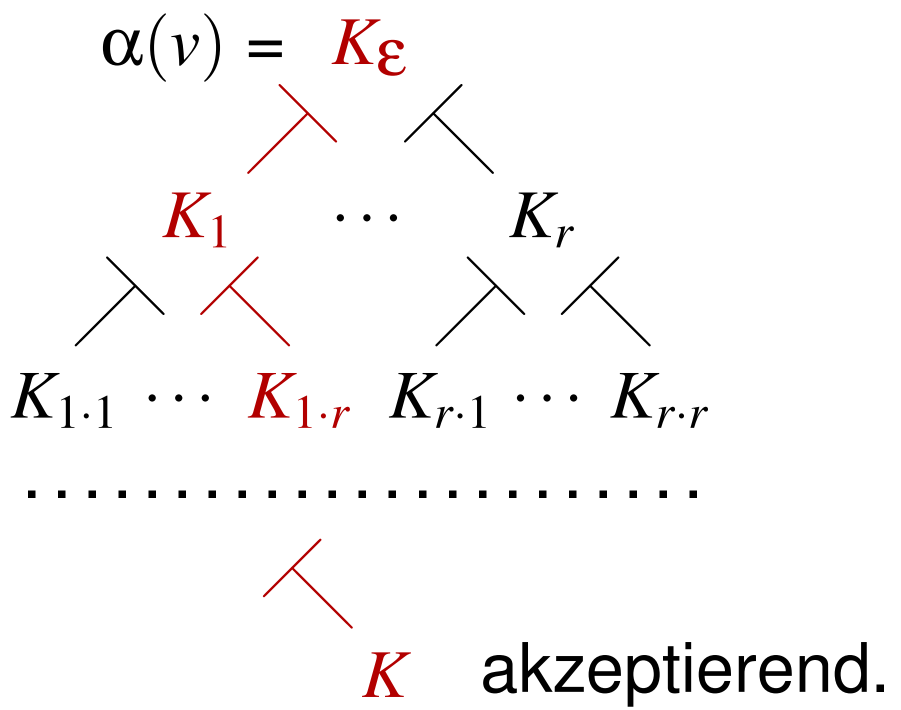
$$\tau = (Q, \Sigma, \Gamma, \delta, q_0, \sqcup, F)$$

- $K = uqv$ heißt akzeptierend, falls $q \in F$ ist.
- τ akzeptiert $w \in \Sigma^*$, falls
  $\exists$ akzeptierende Endkonfiguration K: $\alpha(w) \vdash_\tau^* K$
- Die von τ akzeptierte Sprache ist

$$L(\tau) = \{w \in \Sigma^* \mid \tau \text{ akzeptiert } w\}.$$

Eine Sprache $L \subseteq \Sigma^*$ heißt Turing-akzeptierbar, falls es eine TM τ mit Endzuständen und $L = L(\tau)$ gibt.

Simulation nichtdeterm. TM



Kontextfreie Grammatik

Eine kontextfreie Grammatik ist eine Struktur $G = (N, T, P, S)$ mit

(i) N : Nichtterminalsymbole,

(ii) T : Terminalsymbole mit $N \cap T = \varnothing$,

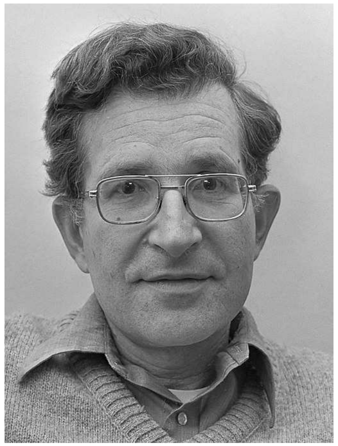
(iii) $S \in N$: Startsymbol,

(iv) $P \subseteq N \times (N \cup T)^*$: Produktionen (Regeln)

$(A, v) \in P$ oder kurz $A \to v$

wobei $u, v, w, \dots \in (N \cup T)^*$.

CHOMSKY-0-**Grammatik**



Noam Chomsky
1977

FOTO: WIKIPEDIA

Eine CHOMSKY-0-Grammatik ist eine Struktur $G = (N, T, P, S)$ mit

(i) N : Nichtterminalsymbole,

(ii) T : Terminalsymbole mit $N \cap T = \varnothing$,

(iii) $S \in N$: Startsymbol,

(iv) $P \subseteq (N \cup T)^* \times (N \cup T)^*$: Produktionen (Regeln)

$(u, v) \in P$ oder kurz $u \rightarrow v$

wobei in u mind. ein Nichtterminalsymbol vorkommt.

CHOMSKY-Grammatik-Typen

Eine CHOMSKY-0-Grammatik $G = (N, T, P, S)$ heißt

(i) kontextsensitiv (CHOMSKY-1-Grammatik) gdw.

$$\forall\, p \to q \in P \quad \exists\, A \in N,\ u, v, w \in (N \cup T)^*,\ v \neq \varepsilon:$$

$$p = \ u\, A\, w \quad \wedge \quad q = u\, v\, w$$

(Sonderregelung: $S \to \varepsilon$ möglich)

CHOMSKY-Grammatik-Typen

Eine CHOMSKY-0-Grammatik $G = (N, T, P, S)$ heißt

(i) kontextsensitiv (CHOMSKY-1-Grammatik) gdw.

$\forall\, p \to q \in P \quad \exists\, A \in N,\ u, v, w \in (N \cup T)^*,\ v \neq \varepsilon$:

$$p = u\,A\,w \quad \wedge \quad q = u\,v\,w$$

(Sonderregelung: $S \to \varepsilon$ möglich)

(ii) kontextfrei (CHOMSKY-2-Grammatik) gdw.

$\forall\, p \to q \in P : \quad p \in N$

CHOMSKY-Grammatik-Typen

Eine CHOMSKY-0-Grammatik $G = (N, T, P, S)$ heißt

(i) kontextsensitiv (CHOMSKY-1-Grammatik) gdw.

$\forall\, p \to q \in P \quad \exists\, A \in N,\ u, v, w \in (N \cup T)^*,\ v \neq \varepsilon$:

$$p = \ u\, A\, w \quad \wedge \quad q = u\, v\, w$$

(Sonderregelung: $S \to \varepsilon$ möglich)

(ii) kontextfrei (CHOMSKY-2-Grammatik) gdw.

$\forall\, p \to q \in P: \quad p \in N$

(iii) rechtslinear (CHOMSKY-3-Grammatik) gdw.

$\forall\, p \to q \in P: \quad p \in N \quad \wedge \quad q \in T^* \cdot N \ \cup\ T^*$

Chomsky-Hierarchie

Betrachtete Sprachklassen:

$CH0$:	Sprachen von	Chomsky-0-	Gramm. erzeugt
$CS = CH1$:		kontextsensitiven	
$CF = CH2$:		kontextfreien	
$RLIN = CH3$:		rechtslinearen	

Es gelten die Inklusionen:

$$RLIN \subseteq CF \subseteq CS \subseteq CH0$$

Beispiel kss Grammatik

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

Monotone Chomsky-0-Grammatik: $G = (N, T, P, S)$

mit $N = \{S, S', B, C\}$ und $T = \{a, b, c\}$ und P wie folgt:

(0) $S \to \varepsilon \mid S'$

(1) $S' \to a\,b\,C \mid a\,S'\,B\,C$

(2) $C\,B \to B\,C$

(3) $b\,B \to b\,b$

(4) $C \to c$

Beispiel kss Grammatik

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

Monotone Chomsky-0-Grammatik: $G = (N, T, P, S)$

mit $N = \{S, S', B, C\}$ und $T = \{a, b, c\}$ und P wie folgt:

$$
\begin{array}{ll}
(0) & S \to \varepsilon \mid S' \\
(1) & S' \to a\,b\,C \mid a\,S'\,B\,C \\
(2) & C\,B \to B\,C \\
(3) & b\,B \to b\,b \\
(4) & C \to c
\end{array}
$$

Umwandlung der Produktion (2) in kss Regeln:

$$
\begin{array}{ll}
(2.1) & C\,B \to C_1\,B \\
(2.2) & C_1\,B \to C_1\,C_2 \\
(2.3) & C_1\,C_2 \to C_1\,C \\
(2.4) & C_1\,C \to B\,C
\end{array}
$$

für neue Nichterminalsymbole C_1 und C_2

Binärkodierung von TM

Beispiel: kleine Rechtsmaschine

$$\tau = r = (\{q_0, q_1\}, \{0, 1\}, \{0, 1, \sqcup\}, \delta, q_0, \sqcup)$$

$$\delta: \quad \begin{array}{ll|lll} q_0 & 0 & q_1 & 0 & R \\ q_0 & 1 & q_1 & 1 & R \\ q_0 & \sqcup & q_1 & \sqcup & R \end{array}$$

Dann ist

$$\begin{array}{llll} w_\tau = & 1 \sharp 2 & \text{und} \quad bw_\tau = & 011\ 0111 \\ & \sharp 0 \sharp 0 \sharp 1 \sharp 0 \sharp 0 & & 01\ 01\ 011\ 01\ 01 \\ & \sharp 0 \sharp 1 \sharp 1 \sharp 1 \sharp 0 & & 01\ 011\ 011\ 011\ 01 \\ & \sharp 0 \sharp 2 \sharp 1 \sharp 2 \sharp 0 & & 01\ 0111\ 011\ 0111\ 01 \end{array}$$

Binärkodierung von TM

Beispiel: kleine Linksmaschine

$$\tau = l = (\{q_0, q_1\}, \{0,1\}, \{0,1,\sqcup\}, \delta, q_0, \sqcup)$$

$$\delta: \quad \begin{array}{cc|ccc} q_0 & 0 & q_1 & 0 & L \\ q_0 & 1 & q_1 & 1 & L \\ q_0 & \sqcup & q_1 & \sqcup & L \end{array}$$

Dann ist

$$\begin{array}{rlcrl} w_\tau = & 1 \sharp 2 & \text{und} & bw_\tau = & 011\ 0111 \\ & \sharp 0 \sharp 0 \sharp 1 \sharp 0 \sharp 1 & & & 01\ 01\ 011\ 01\ 011 \\ & \sharp 0 \sharp 1 \sharp 1 \sharp 1 \sharp 1 & & & 01\ 011\ 011\ 011\ 011 \\ & \sharp 0 \sharp 2 \sharp 1 \sharp 2 \sharp 1 & & & 01\ 0111\ 011\ 0111\ 011 \end{array}$$

Reduktion

Seien $L_1 \subseteq \Sigma_1^*$ und $L_2 \subseteq \Sigma_2^*$ Sprachen (Probleme).

Dann heißt L_1 auf L_2 reduzierbar, falls es eine totale, berechenbare Funktion $f : \Sigma_1^* \to \Sigma_2^*$ gibt, so dass für alle $w \in \Sigma_1^*$ gilt :

$$w \in L_1 \Leftrightarrow f(w) \in L_2.$$

Notation:

$$L_1 \leq L_2 \text{ (mittels } f)$$

Anschaulich: L_1 ist ein Spezialfall von L_2.

Reduktions-Lemma

Sei $L_1 \leq L_2$. Dann gilt:

1. Falls L_2 entscheidbar ist, so auch L_1.
2. Falls L_1 *un*entscheidbar ist, so auch L_2

Reduktions-Lemma

Sei $L_1 \leq L_2$. Dann gilt:

1. Falls L_2 entscheidbar ist, so auch L_1.
2. Falls L_1 *un*entscheidbar ist, so auch L_2
3. Falls L_2 rekursiv aufzählbar ist, so auch L_1.

Rekursive Aufzählbarkeit

Wdh: $L \subseteq A^*$ heißt abzählbar, falls $L = \emptyset$ oder es eine Funktion $\beta : \mathbb{N} \to A^*$ gibt mit

$$L = \beta(\mathbb{N}) = \{\beta(0), \beta(1), \beta(2), \dots\}.$$

Def: $L \subseteq A^*$ heißt rekursiv aufzählbar (r.a.), falls $L = \emptyset$ oder es eine Turing-berechenbare Funktion $\beta : \mathbb{N} \to A^*$ gibt mit

$$L = \beta(\mathbb{N}) = \{\beta(0), \beta(1), \beta(2), \dots\}.$$

Semi-Entscheidbarkeit

Wdh: $L \subseteq A^*$ heißt Turing-entscheidbar, falls die charakteristische Funktion

$$\chi_L : A^* \to \{0, 1\}$$

Turing-berechenbar ist.

Def: $L \subseteq A^*$ heißt semi-entscheidbar, falls die partiell charakteristische Funktion

$$\Psi_L : A^* \xrightarrow{part} \{1\}$$

Turing-berechenbar ist.

Bem: L entscheidbar $\iff$ L and $\overline{L}$ semi-entscheidbar.

Turing-Akzeptierbarkeit

Wdh: $L \subseteq A^*$ heißt Turing-akzeptierbar, falls die es eine TM τ mit Endzuständen gibt mit

$$L = L(\tau).$$

Bem: L semi-entscheidbar $\Longleftrightarrow$ L Turing-akzeptierbar.

Zusammenhang: r.a. und Semi-Entsch.

Wdh: $L \subseteq A^*$ heißt rekursiv aufzählbar (r.a.), falls $L = \emptyset$ oder es eine Turing-berechenbare Funktion $\beta : \mathbb{N} \to A^*$ gibt mit

$$L = \beta(\mathbb{N}) = \{\beta(0), \beta(1), \beta(2), \ldots\}.$$

Wdh: $L \subseteq A^*$ heißt semi-entscheidbar, falls die partiell charakteristische Funktion

$$\Psi_L : A^* \xrightarrow{part} \{1\}$$

Turing-berechenbar ist.

Lemma: L r.a. $\iff$ L semi-entscheidbar.

Charakterisierung von r.a.

Für alle Sprachen $L \subseteq A^*$ sind äquivalent:

1. L ist rekursiv aufzählbar (r.a.).
2. L ist Ergebnisbereich einer Turingmaschine τ, d.h.
   $L = \{v \in A^* \mid \exists w \in \Sigma^*$ mit $h_\tau(w) = v\}$.
3. L ist semi-entscheidbar.
4. L ist Haltebereich einer Turingmaschine τ, d.h.
   $L = \{v \in A^* \mid h_\tau(v)$ existiert$\}$.
5. L ist TURING-akzeptierbar.
6. L ist Chomsky-0.

Verifikationsproblem

Gegeben: Programm $\mathcal{P}$ und Spezifikation $\mathcal{S}$

Frage: Erfüllt $\mathcal{P}$ die Spezifikation $\mathcal{S}$?

Wir formalisieren dieses Problem wie folgt:

- Programm $\mathcal{P}$: Turingmaschine τ mit Eingabealphabet $B = \{0,1\}$
- Spezifikation $\mathcal{S}$: Teilmenge $\mathcal{S}$ von $\mathcal{T}_{B,B}$, der Menge aller TURING-berechenbaren Funktionen : $B^* \xrightarrow{part} B^*$
- $\mathcal{P}$ erfüllt $\mathcal{S}$: $h_\tau \in \mathcal{S}$

Satz von Rice

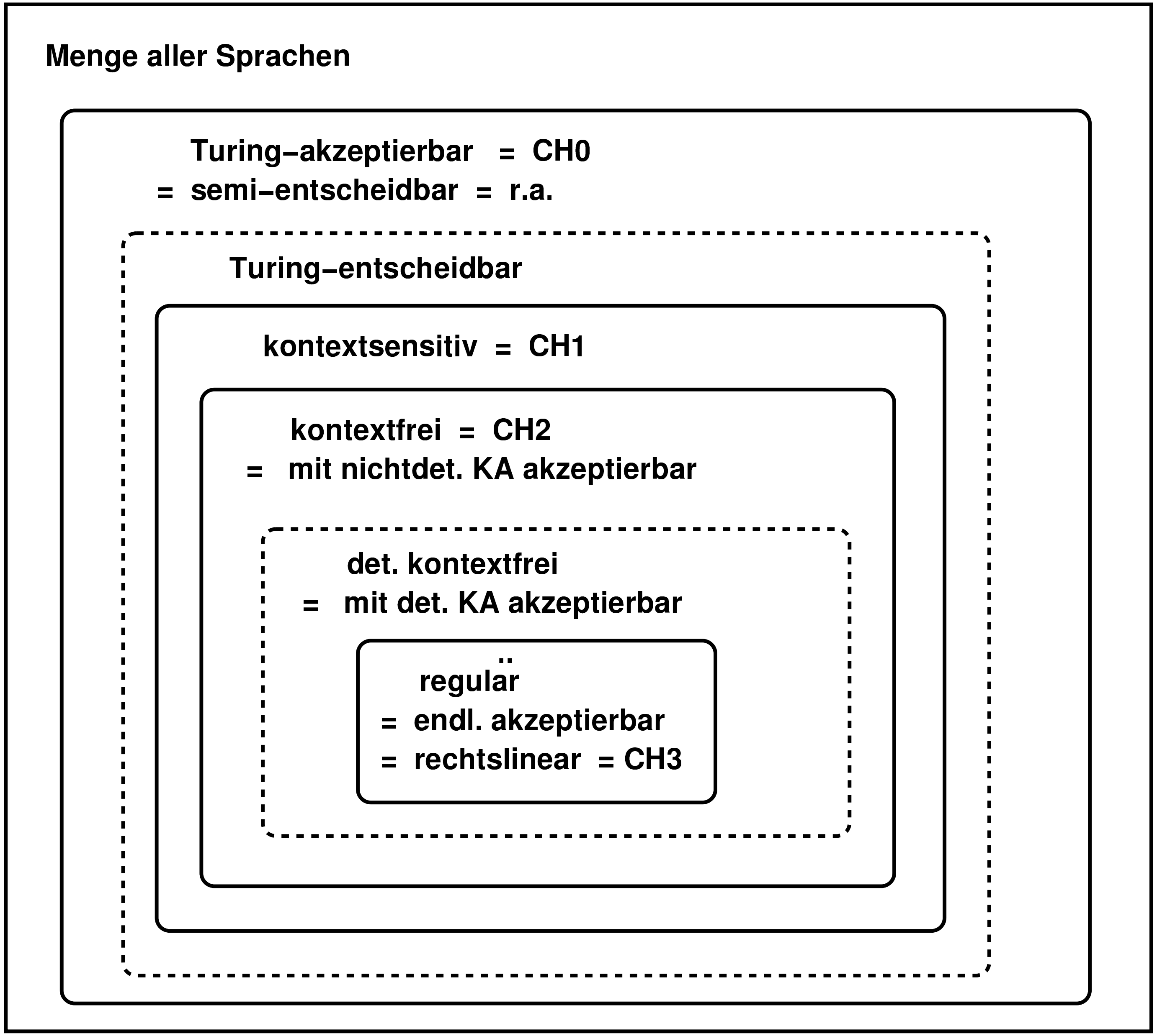
Sei $\mathcal{S}$ eine beliebige, nicht-triviale Teilmenge von $\mathcal{T}_{B,B}$, d.h. es gelte $\varnothing \subset \mathcal{S} \subset \mathcal{T}_{B,B}$.

Dann ist die Sprache

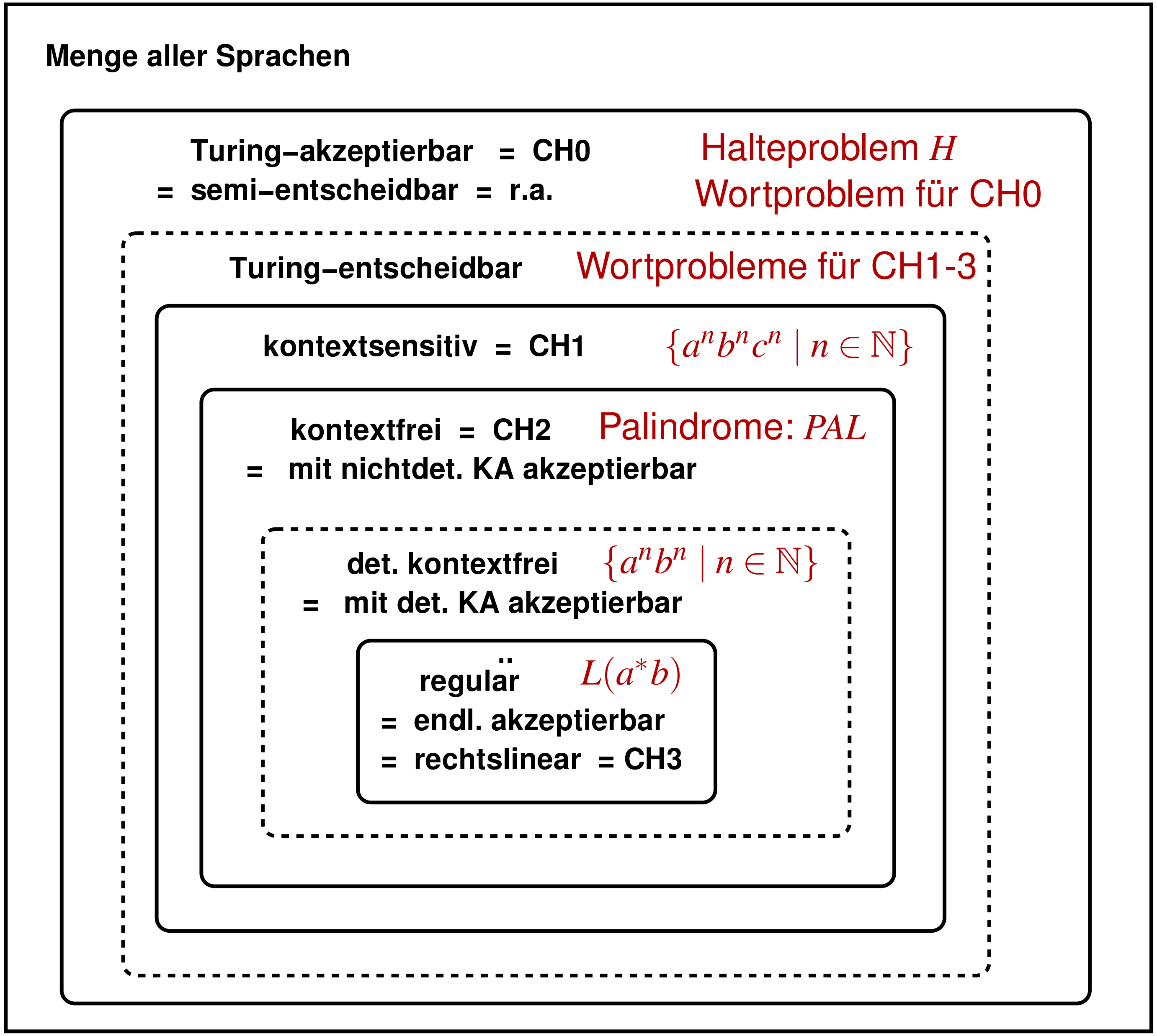
$$BW(\mathcal{S}) = \{bw_\tau \mid h_\tau \in \mathcal{S}\} \subseteq B^*$$

der Binärkodierung aller Turingmaschinen τ, deren berechnete Funktion h_τ in der Funktionenmenge $\mathcal{S}$ liegt, unentscheidbar.

Chomsky-Hierarchie



Chomsky-Hierarchie



PCP: Post’s Correspondence Problem

- Eine Eingabe / Instanz des PCP ist eine endliche Folge Y von Wortpaaren

$$Y = ((u_1, v_1), \ldots, (u_n, v_n))$$

mit $n \geq 1$ und $u_i, v_i \in SYM^*$ für $i = 1, \ldots, n$. Falls $u_i, v_i \in X^*$ gilt, heißt Y auch Eingabe des PCP über X.

PCP: Post's Correspondence Problem

- Eine Eingabe / Instanz des PCP ist
  eine endliche Folge Y von Wortpaaren

  $$Y = ((u_1, v_1), \ldots, (u_n, v_n))$$

  mit $n \geq 1$ und $u_i, v_i \in SYM^*$ für $i = 1, \ldots, n$. Falls $u_i, v_i \in X^*$ gilt,
  heißt Y auch Eingabe des PCP über X.

- Eine Korrespondenz oder Lösung von Y ist
  eine endliche Folge von Indizes

  $$(i_1, \ldots, i_m)$$

  mit $m \geq 1$ und $i_1, \ldots, i_m \in \{1, \ldots, n\}$, so dass

  $$u_{i_1} u_{i_2} \ldots u_{i_m} = v_{i_1} v_{i_2} \ldots v_{i_m} \ .$$

  Y heißt lösbar, falls es eine Lösung von Y gibt.

Postsches Korrespondenzproblem

1. Das PCP ist folgendes Problem:

   Gegeben: Eingabe Y des PCP
   Frage: Besitzt Y eine Korrespondenz ?

2. Das PCP über X ist folgendes Problem:

   Gegeben: Eingabe Y des PCP über X
   Frage: Besitzt Y eine Korrespondenz ?

3. Das modifizierte PCP (kurz MPCP) ist folgendes Problem:

   Gegeben: Eingabe $Y = ((u_1, v_1), \ldots, (u_n, v_n))$ des PCP,
   wobei $u_i \neq \varepsilon$ für $i = 1, \ldots, n$
   Frage: Besitzt Y eine Korrespondenz $(i_1, \ldots, i_m)$ mit $i_1 = 1$?

Komplexitätsklassen

Sei $f : \mathbb{N} \longrightarrow \mathbb{R}$.

DTIME $(f(n)) = \{L \mid$ es gibt eine determ. Mehrband-TM der Zeitkomplexität $f(n)$, die L akzeptiert $\}$

NTIME $(f(n)) = \{L \mid$ es gibt eine nichtdet. Mehrband-TM der Zeitkomplexität $f(n)$, die L akzeptiert $\}$

DSPACE $(f(n)) = \{L \mid$ es gibt eine determ. Mehrband-TM der Platzkomplexität $f(n)$, die L akzeptiert $\}$

NSPACE $(f(n)) = \{L \mid$ es gibt eine nichtdet. Mehrband-TM der Platzkomplexität $f(n)$, die L akzeptiert $\}$

Die Klassen P und NP

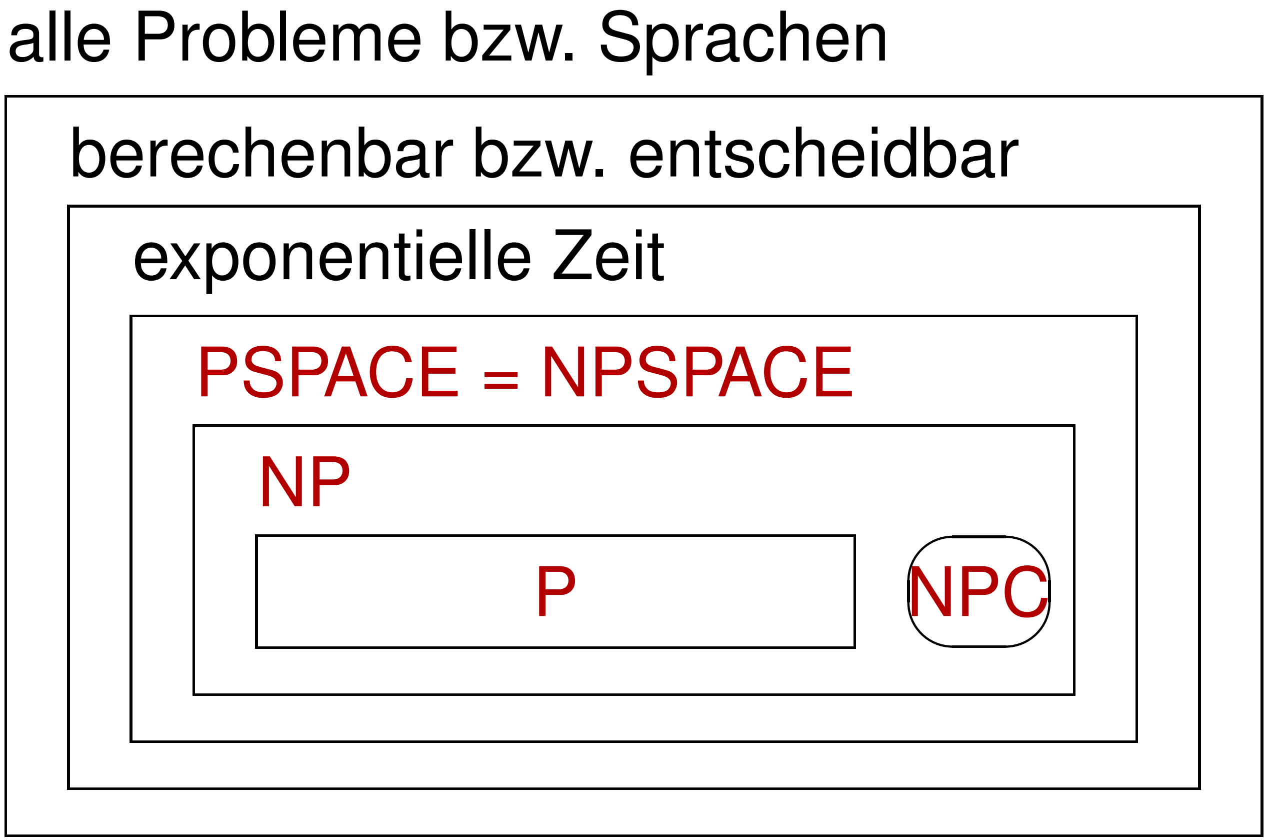
Betrachte Polynome

$$p(n) = a_k n^k + \cdots + a_1 n + a_0$$

wobei $a_i \in \mathbb{N}$ für $i = 0, \ldots, k$ mit $k \in \mathbb{N}$ und $a_k \neq 0$.

$$\begin{aligned}
\mathrm{P} &= \bigcup_{p \text{ Polynom in } n} \mathrm{DTIME}(p(n)) \\
\mathrm{NP} &= \bigcup_{p \text{ Polynom in } n} \mathrm{NTIME}(p(n)) \\
\mathrm{PSPACE} &= \bigcup_{p \text{ Polynom in } n} \mathrm{DSPACE}(p(n)) \\
\mathrm{NPSPACE} &= \bigcup_{p \text{ Polynom in } n} \mathrm{NSPACE}(p(n))
\end{aligned}$$

Komplexitätshierarchie



Polynomielle Reduzierbarkeit

Seien $L_1 \subseteq \Sigma_1^*$ und $L_2 \subseteq \Sigma_2^*$ Sprachen (Probleme).

Dann heißt L_1 auf L_2 polynomiell reduzierbar, falls es eine totale und mit der Zeitkomplexität eines Polynoms berechenbare Funktion $f : \Sigma_1^* \to \Sigma_2^*$ gibt,
so dass für alle $w \in \Sigma_1^*$ gilt:

$$w \in L_1 \Leftrightarrow f(w) \in L_2.$$

Notation:

$$L_1 \leq_p L_2 \text{ (mittels } f\text{)}$$

Lemma: polynomielle Reduktion

Sei $L_1 \leq_p L_2$. Dann gilt:

1. Falls $L_2 \in P$ gilt, so auch $L_1 \in P$.
2. Falls $L_2 \in NP$ gilt, so auch $L_1 \in NP$.

Lemma: polynomielle Reduktion

Sei $L_1 \leq_p L_2$. Dann gilt:

1. Falls $L_2 \in P$ gilt, so auch $L_1 \in P$.
2. Falls $L_2 \in NP$ gilt, so auch $L_1 \in NP$.
3. Falls L_1 NP-vollständig ist und $L_2 \in NP$ gilt, so ist auch L_2 NP-vollständig.

Satz von Cook

Stephen A. Cook

FOTO AUS WIKIPEDIA

Satz (Cook, 1971)

SAT ist NP-vollständig.

Boolesche Ausdrücke

Ein Boolescher Ausdruck (eine Boolesche Formel) ist ein Wort über $B = \{0, 1\}$ und Variablen aus $V = \{v_1, v_2, \ldots\}$ mit den Operationen Negation ($\neg$), Konjunktion ($\wedge$) und Disjunktion ($\vee$):

(1) Jedes $v_i \in V$ ist ein Boolescher Ausdruck.

(2) Jedes $b \in B$ ist ein Boolescher Ausdruck.

(3) Wenn E ein Boolescher Ausdruck ist, dann auch $\neg E$.

(4) Wenn E_1, E_2 Boolesche Ausdrücke sind, dann auch $(E_1 \vee E_2)$ und $(E_1 \wedge E_2)$.

Belegung

Eine Belegung β ist eine Abbildung

$$\beta : V \rightarrow \{0,1\},$$

die wie folgt auf Boolesche Ausdrücke fortgesetzt wird:

- $\beta(0) = 0$ und $\beta(1) = 1$
- $\beta(\neg E) = 1 - \beta(E)$
- $\beta(E_1 \wedge E_2) = Min(\beta(E_1), \beta(E_2))$
- $\beta(E_1 \vee E_2) = Max(\beta(E_1), \beta(E_2))$.

Erfüllbarkeitsproblem (SAT)

- Ein Boolescher Ausdruck E heißt

  erfüllbar (engl. satisfiable),

  wenn es eine Belegung β mit $\beta(E) = 1$ gibt.

- Das Erfüllbarkeitsproblem (engl. satisfiability problem) wird durch die Sprache

  $$\text{SAT} = \{E \mid E \text{ ist Boolescher Ausdruck und } E \text{ ist erfüllbar}\}.$$

  beschreiben.

SAT ist NP-hart

Sei $L \in$ NP. Dann existiert nichtdet. 1-Band TM τ, die L in polyn. Zeitkomplexität $p(n)$ akzeptiert, wobei $n = |\, w \,|$ für das Eingabewort w ist.

Zu zeigen: $L \leq_p$ SAT

Beweisskizze:

Wir konstruieren zum Eingabewort w einen Booleschen Ausdruck $E_{\tau,w}$, der das Verhalten von τ wie folgt beschreibt

$$\tau \text{ akzeptiert } w \;\Leftrightarrow\; E_{\tau,w} \text{ ist erfüllbar.}$$

$E_{\tau,w}$ ist von der Form

$$E_{\tau,w} \;:=\; R \wedge A \wedge T_1 \wedge T_2 \wedge End\,.$$

Genau eine Variable ist 1

$$
\begin{array}{rrrrrrrrrrrr}
G(v_1,\ldots,v_m) & := (& & v_1 & \wedge & \neg v_2 & \wedge & \ldots & \wedge & \neg v_m & \\
& & \vee & \neg v_1 & \wedge & v_2 & \wedge & \ldots & \wedge & \neg v_m & \\
& & & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \\
& & \vee & \neg v_1 & \wedge & \neg v_2 & \wedge & \ldots & \wedge & v_m &)
\end{array}
$$

Randbedingungen

$$
\begin{array}{rcl}
R & := & \bigwedge_t G(zust_{t,q_0},\ldots,zust_{t,q_k}) \\
& \wedge & \bigwedge_t G(pos_{t,-p(n)},\ldots,pos_{t,p(n)}) \\
& \wedge & \bigwedge_{t,i} G(band_{t,i,a_1},\ldots,band_{t,i,a_l})
\end{array}
$$

Anfangsbedingungen

$$
\begin{aligned}
A \quad := \quad & zust_{0,q_0} \wedge pos_{0,1} \\
\wedge \quad & \bigwedge_{i=-p(n)}^{0} band_{0,i,\sqcup} \\
\wedge \quad & \bigwedge_{i=1}^{n} band_{0,i,x_i} \\
\wedge \quad & \bigwedge_{i=n+1}^{p(n)} band_{0,i,\sqcup}
\end{aligned}
$$

Transitionen

$$T_1 \;:=\; \bigwedge_{t,q,i,a} \; (zust_{t,q} \wedge pos_{t,i} \wedge band_{t,i,a} \longrightarrow$$

$$\bigvee_{(q',a',P)\in\delta(q,a)} zust_{t+1,q'} \wedge pos_{t+1,i+m(P)} \wedge band_{t+1,i,a'})$$

$$T_2 \;:=\; \bigwedge_{t,q,i,a} \; (\neg pos_{t,i} \wedge band_{t,i,a} \longrightarrow band_{t+1,i,a})$$

Endbedingung

$$End \;:=\; \bigvee_{q\in F} zust_{p(n),q}$$

Konjunktive Normalform (KNF)

(0) Wenn $v \in V$, dann heißen v und $\overline{v}$ (bzw. $\neg v$) Literale.

(1) Wenn $y_1, y_2, \ldots, y_k$ Literale sind,
dann ist $(y_1 \vee y_2 \vee \ldots \vee y_k)$ eine Klausel (der Ordnung k)

(2) Wenn $c_1, c_2, \ldots, c_r$ Klauseln (der Ordnung $\leq k$) sind,
dann ist $c_1 \wedge c_2 \wedge \ldots \wedge c_r$ ein Boolescher Ausdruck in
konjunktiver Normalform (KNF) (der Ordnung $\leq k$).

Wenn mindestens eine der Klauseln k Literale enthält,
dann heißt dieser Ausdruck eine KNF der Ordnung k.

Varianten von SAT

KNF-SAT = $\{ E \in$ SAT $| E$ ist Boolescher Ausdruck in KNF $\}$

KNF-SAT(k) = $\{ E \in$ KNF-SAT $| E$ ist von der Ordnung k $\}$.

Satz: KNF-SAT(3) ist NP-vollständig.