Formale Grundlagen der Informatik I
WS 08/09

Prof. Dr. Ulrich Kohlenbach
TUD, Fachbereich Mathematik

Inhaltsübersicht

1) **Mengen, Relationen, Funktionen, algebraische Strukturen und logische Grundbegriffe**
   - Mengenoperationen, Relationen, Funktionen
   - Algebraische Strukturen: Halbgruppen, Monoide, Gruppen, Boolesche Algebren
   - Logische Grundbegriffe: aussagenlogische Verknüpfungen, Quantoren
   - Beweise durch Induktion

2) **Endliche Automaten und reguläre Sprachen**

- Wörter, Sprachen, reguläre Sprachen
- Deterministische/nichtdeterministische endliche Automaten
- Erkennung regulärer Sprachen durch endliche Automaten
- Die Sätze von Kleene und Myhill-Nerode
- Nichtreguläre Sprachen und Entscheidungsprobleme

3) **Grammatiken und Chomsky-Hierarchie**

- Grammatiken und Normalformen – Stufen der Chomsky-Hierarchie
- kontextfreie Sprachen und Kellerautomaten
- kontextsensitive Sprachen

4) **Universelles Berechnungsmodell**

- Turing-Maschinen
- Berechenbarkeit, Entscheidbarkeit, Aufzählbarkeit
- Unentscheidbarkeit

Literatur

- **M. Otto**: Skript zu Formale Grundl. d. Inf. I (WS06/07). Siehe Vorlesungs-Homepage.
- **U. Schöning**: Theoretische Informatik - kurzgefasst. Spektrum, 4. Aufl. 2001.
- J.E. Hopcroft, R. Motwani, J.D. Ullmann: Einführung in die Automatentheorie, Formale Sprachen und Komplexitätstheorie. Pearson Studium 2. Aufl. 2002
- D. C. Kozen: Automata and Computability. Springer Undergraduate Texts in Computer Science. 1997
- M.D. Davis, R. Sigal, E.J. Weyuker: Computability, Complexity, and Languages. 2nd edition. Academic Press 1994.

Einführung: Transitionssysteme

Idee: Beschreibe ein System durch eine endliche Menge Q von **Zuständen** und den möglichen Übergängen (**Transitionen**) zwischen Zuständen $q, q' \in Q$, die durch Zeichen a aus einer endlichen Menge Σ gekennzeichnet sind:

$$q \xrightarrow{a} q' \qquad (q, q' \in Q, a \in \Sigma)$$

besagt, dass das System mit einer a-Transition vom Zustand q in den Zustand q' übergehen kann.

Transitionssysteme: Beispiel

Weckzeit-Kontrolle eines Weckers:

Zustände: (h, m, q)

$$\begin{cases} h \in \mathrm{H} = \{0, \ldots, 23\} \\ m \in \mathrm{M} = \{0, \ldots, 59\} \\ q \in \{\mathrm{SETH}, \mathrm{SETM}, \mathrm{NIL}, \mathrm{ERROR}\} \end{cases}$$

Aktionen/Operationen: *seth*, *setm*, $+$, $-$, *set*, *reset*

Typische Transitionen z.B.:

$$(h, m, \mathrm{NIL}) \xrightarrow{seth} (h, m, \mathrm{SETH}) \quad \text{(in den H-Setzen Modus)}$$

$$(h, m, \mathrm{SETH}) \xrightarrow{set} (h, m, \mathrm{NIL}) \quad \text{(Ende H-Setzen Modus)}$$

$$(h, m, \mathrm{SETH}) \xrightarrow{seth} (h, m, \mathrm{ERROR}) \quad \text{(bereits in H-Setzen Modus)}$$

$$(h, m, \mathrm{NIL}) \xrightarrow{+} (h, m, \mathrm{ERROR}) \quad \text{(da nicht in Setzen Modus)}$$

$$(h, m, \mathrm{SETH}) \xrightarrow{+} ((h+1) \bmod 24, m, \mathrm{SETH}) \quad \text{(H vorstellen)}$$

$$(h, m, \mathrm{ERROR}) \xrightarrow{reset} (0, 0, \mathrm{NIL}) \quad \text{(reset)}$$

Mengen

„Eine Menge ist die Zusammenfassung von bestimmten, wohlunterschiedenen Objekten unserer Anschauung oder unseres Denkens, welche die Elemente der Menge genannt werden, zu einem Ganzen." (G. Cantor 1895)

Mengen sind unstrukturierte Ansammlungen von Objekten. Diese heißen **Elemente** der Menge. Welche (unendlichen) Mengen es gibt und welche Eigenschaften diese haben, ist Gegenstand der Axiomatischen Mengenlehre. In dieser Vorlesung kommen nur Mengen vor, die wir durch naive Aufzählung ihrer Elemente direkt angeben können.

Beispiele:

- $\emptyset := \{\}$ die leere Menge
- $\{\emptyset\}$ die Menge, deren einziges Element die leere Menge ist
- $\mathbb{B} := \{0, 1\}$ Menge der Booleschen Werte
- $\mathbb{N} := \{0, 1, 2, 3, \ldots\}$ Menge der natürlichen Zahlen (mit 0)
- $\mathbb{Z} := \{0, 1, -1, 2, -2, 3, -3, \ldots\}$ Menge der ganzen Zahlen
- $\mathbb{R}$ Menge der reellen Zahlen (Punkte der Zahlengerade)

Im folgenden stehen A, B für Mengen.

Mengenoperationen I

- **Elementbeziehung:** $a \in A$ („a ist Element von A"),
  $a \notin A$ („a ist nicht Element von A")
- **Teilmengen (Inklusion):**

  $B \subseteq A :\Leftrightarrow$ jedes Element von B ist auch Element von A,

  z.B. $\emptyset \subseteq \{0,1\} \subseteq \mathbb{N} \subseteq \mathbb{Z} \subseteq \mathbb{R}$. B heißt **Teilmenge** von A.
- **Potenzmenge (einer Menge A):**

  $\mathcal{P}(A) := \{B : B \subseteq A\}$ (Menge aller Teilmengen von A).
- **Mengengleichheit:** $A = B :\Leftrightarrow (A \subseteq B$ und $B \subseteq A)$, d.h. A und B haben dieselben Elemente.

Mengenoperationen II

- **Echte Inklusion:** $B \subset A :\Leftrightarrow (B \subseteq A$ und $B \neq A)$
- **Definition von Teilmengen:** $B := \{a \in A : a$ erfüllt $P\}$ für eine Eigenschaft P.
- **Mengendurchschnitt:** $A \cap B := \{c : c \in A$ und $c \in B\}$ A und B sind **disjunkt** gdw. $A \cap B = \emptyset$.
- **Mengenvereinigung:** $A \cup B := \{c : c \in A$ oder $c \in B\}$
- **Mengendifferenz:** $A \setminus B := \{a \in A : a \notin B\}$.
- **Komplement:** für Teilmengen einer festen Menge M

$$\overline{B} : M \setminus B \text{ (Komplement bzgl. } M).$$

Bemerkung: Vereinigungen und Durchschnitte lassen sich auch für beliebige Familien $(A_i)_{i \in I}$ von Mengen erklären:

$$\bigcap_{i \in I} A_i := \{a : a \in A_i \text{ für alle } i \in I\},$$

$$\bigcup_{i \in I} A_i := \{a : a \in A_i \text{ für mindestens ein } i \in I\}.$$

Tupel und kartesisches Produkt

Geordnete Paare (a, b) unterscheiden sich von Paarmengen $\{a, b\}$ darin, dass es auf die Reihenfolge ankommt, d.h.

$$(a, b) = (c, d) :\Leftrightarrow a = c \text{ und } b = d$$

Insbesondere: $(a, b) = (b, a)$ gdw.$a = b$, während stets $\{a, b\} = \{b, a\}$.

Analog: **n-Tupel:** $(a_1, \ldots, a_n)$ mit n Komponenten.

Kartesisches Produkt von Mengen $A_1, \ldots, A_n$:

$$A_1 \times \ldots \times A_n := \{(a_1, \ldots, a_n) : a_i \in A_i \text{ für } 1 \leq i \leq n\}.$$

Falls $A = A_1 = A_2 = \ldots = A_n$ schreibt man auch A^n für die Menge aller n-Tupel über A.

Alphabete, Wörter, Sprachen

In vielen Anwendungen in der Informatik werden endliche, nicht-leere Mengen Σ als **Alphabet** bezeichnet und die Elemente $a \in \Sigma$ als **Buchstaben** oder **Zeichen**.

Wörter über einem Alphabet Σ sind endliche Folgen

$$w = a_1 \dots a_n$$

von Buchstaben $a_i \in \Sigma$.

n ist die **Länge** $|w|$ von w.
Wörter der Länge n werden mit n-Tupeln aus Σ^n identifiziert.
Wörter der Länge 1 werden mit Buchstaben identifiziert.

Menge aller Wörter über Σ:

$$\Sigma^* := \bigcup_{n \in \mathbb{N}} \Sigma^n.$$

Leeres Wort: $\varepsilon \in \Sigma^*$.

Σ-Sprache: Teilmenge $L \subseteq \Sigma^*$, d.h. Menge von Σ-Wörtern.

Menge der nicht-leeren Wörter:

$$\Sigma^+ := \Sigma^* \setminus \{\varepsilon\} = \{w \in \Sigma^* : |w| \geq 1\} = \bigcup_{n \geq 1} \Sigma^n.$$

Relationen

n-stellige Relationen: Teilmenge $R \subseteq A_1 \times \ldots \times A_n$.

Falls $R \subseteq A^n$ spricht man von einer **Relation über A**.

Beispiele:

- Ordnungsrelationen, z.B. $\leq := \{(n,m) \in \mathbb{N}^2 : n \leq m\}$ 2-stell.
- Funktionsrelationen $\{(n,m,k) \in \mathbb{R}^3 : k = n \cdot m\}$. (3-stell.)
- Kantenrelationen E in Graph G : $\{(u,v) \in G^2 : u \xrightarrow{E} v\}$, d.h. $(u,v) \in E$ gdw. von u eine E-Kante nach v ausgeht.
- Präfixrelation auf Σ^* :

$$\preccurlyeq := \{(u, uw) : u, w \in \Sigma^*\} \subseteq \Sigma^* \times \Sigma^*,$$

d.h. $(u,v) \in \preccurlyeq$ gdw. u Anfangsstück ('Präfix') von v (2-stell.).

Äquivalenzrelationen

Bei 2-stelligen (oder ,,binären") Relationen R schreiben wir meistens ,,xRy" statt ,,$(x, y) \in R$", also $x \leq y, x \preccurlyeq x$ etc. (,,infixe" Notation).

Mögliche Eigenschaften binärer Relationen über A: z.B.

- **Reflexivität:** für alle $a \in A$ gilt: aRa.
- **Symmetrie:** für alle $a, b \in A$ gilt: aRb gdw. bRa.
- **Transitivität:** für alle $a, b, c \in A$ gilt: wenn aRb und bRc, dann auch aRc.

Beispiele: $\leq$ auf $\mathbb{N}$ und $\preccurlyeq$ auf Σ^* sind reflexiv und transitiv, aber nicht symmetrisch. $<$ auf $\mathbb{N}$ ist transitiv, aber weder reflexiv noch symmetrisch.

Definition: Eine reflexive, symmetrische und transitive Relation $R \subseteq A^2$ heißt **Äquivalenzrelation**.
Wir schreiben oft $\sim$ statt R.

Beispiele:

- Die Gleichheit '$=$' von Mengen ist eine Äquivalenzrelation.
- Auf $\mathbb{N}^2$ wird durch
  $(n_1, m_1) \sim (n_2, m_2) :\Leftrightarrow n_1 - m_1 = n_2 - m_2$ eine Äquivalenzrelation definiert. Reflexivität und Symmetrie sind trivial. Transitivität:

  $$n_1 - m_1 = n_2 - m_2 \text{ und } n_2 - m_2 = n_3 - m_3 \Rightarrow$$

  $$n_1 - m_1 = n_3 - m_3.$$

- Auf Σ^* ist $w_1 \sim w_2 :\Leftrightarrow |w_1| = |w_2|$ eine Äquivalenzrelation.

Äquivalenzklassen

Definition: Sei A eine Menge und $\sim$ eine Äquivalenzrelation auf A. Für jedes $a \in A$ def. wir die **Äquivalenzklasse** von a durch

$$[a]_\sim := \{b \in A : a \sim b\} \ (\subseteq A).$$

Satz:

1) $a \in [a]_\sim$ und somit $A = \bigcup_{a \in A} [a]_\sim$.

2) $a \sim b \Leftrightarrow [a]_\sim = [b]_\sim$ ($\sim$ also verallgemeinerte Gleichheit).

3) $a \nsim b \Leftrightarrow [a]_\sim \cap [b]_\sim = \emptyset$.

Also: durch **Quotient** $A/\sim := \{[a]_\sim : a \in A\}$ Zerlegung von A in disjunkte Teilmengen.

Falls $A/\sim$ endlich: $\mathrm{index}(\sim) := |A/\sim|$ **Index von** $\sim$.

Funktionen

Definition: Eine Relation $R \subseteq A \times B$ heißt **Funktion von A nach B**, falls es zu jedem $a \in A$ genau ein $b \in B$ gibt mit $(a, b) \in R$.

Funktionen werden üblicherweise mit f, g, h, etc. bezeichnet und man schreibt $f : A \to B$.

Das eindeutig bestimmte $b \in B$ mit $(a, b) \in f$ wird mit $f(a)$ bezeichnet und **Bild von a unter f** genannt.

A heißt **Definitionsbereich** und B **Zielbereich** von f.

$$f[A] := \{f(a) : a \in A\} \subseteq B \textbf{ Bild von } f.$$

Wir schreiben auch $f(A)$ statt $f[A]$.

Für $B' \subseteq B$ heißt

$$f^{-1}[B'] := \{a \in A : f(a) \in B'\}$$

Urbild von B'.

Beachte: Urbilder können auch leer sein.

Beispiel: Natürliche Projektion

$$\pi_\sim : A \to P(A),\ a \mapsto [a]_\sim,$$

wobei $\sim$ eine Äquivalenzrelation auf A ist.
$\pi_\sim[A] = A/\sim$.

Definition: Sei $f : A \to B$ eine Funktion. f heißt

(i) **surjektiv**, falls $f[A] = B$, d.h. zu jedem $b \in B$ gibt es **mindestens ein** $a \in A$ mit $f(a) = b$, d.h. $f^{-1}[\{b\}] \neq \emptyset$.

(ii) **injektiv**, falls stets

$$a_1 \neq a_2 \;\Rightarrow\; f(a_1) \neq f(a_2),$$

d.h. zu jedem $b \in B$ gibt es **höchstens ein** $a \in A$ mit $f(a) = b$, d.h. $f^{-1}[\{b\}] = \emptyset$ oder $= \{a\}$ (Einermenge).

(iii) **bijektiv**, falls f surjektiv und injektiv ist, d.h. zu jedem $b \in B$ gibt es **genau ein** $a \in A$ mit $f(a) = b$, d.h. $f^{-1}[\{b\}]$ ist für jedes b eine Einermenge.

Komposition von Funktionen: Seien $f : A \to B$ und $g : B \to C$ Funktionen. Dann wird durch Komposition von f und g eine neue Funktion erklärt

$$g \circ f : A \to C, \quad a \longmapsto g(f(a)).$$

Definition: Sei $f : A \to B$ eine Funktion. Eine Funktion $g : B \to A$ heißt **Umkehrfunktion** von f, falls

$$g \circ f = \mathrm{id}_A \text{ und } f \circ g = \mathrm{id}_B,$$

mit den Identitätsfunktionen $\mathrm{id}_A(a) := a$, $\mathrm{id}_B(b) := b$ auf A, B.

Satz: Eine Funktion $f : A \to B$ ist genau dann eine Bijektion, wenn f eine Umkehrfunktion $f^{-1} : B \to A$ besitzt:

$$f^{-1}(b) := \text{ das eindeutig bestimmte } a \in A \text{ mit } f(a) = b.$$

Operationen

n-stellige Funktion auf A: Funktion $f : A^n \to B$.

n-stellige Operation auf A: Funktion $f : A^n \to A$.

- Addition und Multiplikation sind 2-stellige Operationen auf $\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}$. Ebenso ist Division eine 2-stellige Operation auf $\mathbb{Q} \setminus \{0\}$ und $\mathbb{R} \setminus \{0\}$.
- Konkatenation $\cdot$ oder $*$ auf Σ^*

  $(a_1 \dots a_n, b_1 \dots b_m) \mapsto a_1 \dots a_n \cdot b_1 \dots b_m = a_1 \dots a_n b_1 \dots b_m$

  ist eine 2-stellige Operation.

Mögliche Eigenschaften von 2-stell. Operationen

- **Assoziativität:** $(a * b) * c = a * (b * c)$ für alle $a, b, c \in A$.
- **Kommutivität:** $a * b = b * a$ für alle $a, b \in A$.
- **Neutrales Element $e \in A$:** $a * e = e * a = a$ für alle $a \in A$.
- **Inverses Element (falls e existent) $a' \in A$ zu $a \in A$:** $a * a' = a' * a = e$.

Bemerkung: Das neutrale Element e und (falls e existent und $*$ assoziativ) zu a das inverse Element a' sind eindeutig bestimmt.

Beispiel: Die Konkatenation auf Σ^* ist assoziativ (aber nicht kommutativ), hat ε als neutrales Element, aber für $w \neq \varepsilon$ kein inverses Element.

Strukturen $\mathcal{M}$

Eine **Struktur** ist ein Tupel bestehend aus einer (i.a. nicht-leeren) Trägermenge M und gewissen Konstanten („0-stellige" Operationen), Operationen und Relationen.

Beispiele:

- **Standardstrukturen der Algebra** wie Gruppen $(\mathbb{Z}, 0, +)$, Ringe $(\mathbb{Z}, 0, 1, +, \cdot)$, Körper $(\mathbb{Q}, 0, 1, +, \cdot)$, geordnete Körper $(\mathbb{R}, 0, 1, +, \cdot, <)$.
- **Graphen** (V, E) mit Knotenmenge V und Kantenmenge E.
- **Monoide** (assoziative 2-stellige Operation mit neutralem Element, z.B. $(\mathbb{N}, 0, +)$):

  Wort-Monoid: $(\Sigma^*, \varepsilon, \cdot)$, mit Konkatenation $\cdot$.

Boolesche Algebren $(B, 0, 1, +, \cdot, ')$

BA1: $+$ und $\cdot$ assoziativ und kommutativ, d.h. für alle $x, y, z \in B$:

$$(x + y) + z = x + (y + z), \qquad x + y = y + x$$

$$(x \cdot y) \cdot z = x \cdot (y \cdot z), \qquad x \cdot y = y \cdot x.$$

BA2: $+$ und $\cdot$ distributiv, d.h. für alle $x, y, z \in B$

$$x \cdot (y + z) = (x \cdot y) + (x \cdot z), \quad x + (y \cdot z) = (x + y) \cdot (x + z).$$

BA3: 0 und 1 neutrale Elemente bzgl. $+$ und $\cdot$:

$$x + 0 = x \qquad x \cdot 1 = x \text{ für alle } x \in B.$$

BA4: Komplement: $0 \neq 1$ und $x \cdot x' = 0$ und $x + x' = 1$ für alle $x \in B$.

Beispiele:

$\big(\mathcal{P}(M), \emptyset, M, \cup, \cap, \bar{}\big)$ für $M \neq \emptyset$ und $\big(\mathbb{B}, 0, 1, \vee, \wedge, \neg\big)$.

Homomorphismen zwischen Strukturen $\mathcal{A}, \mathcal{B}$

Eine **strukturerhaltende Abbildung** f zwischen Strukturen desselben Typs z.B. $\mathcal{A} = (A, e^{\mathcal{A}}, *^{\mathcal{A}}, R^{\mathcal{A}})$ und $\mathcal{B} = (B, e^{\mathcal{B}}, *^{\mathcal{B}}, R^{\mathcal{B}})$ mit je einer zweistelligen Operation und Relation und einer Konstanten

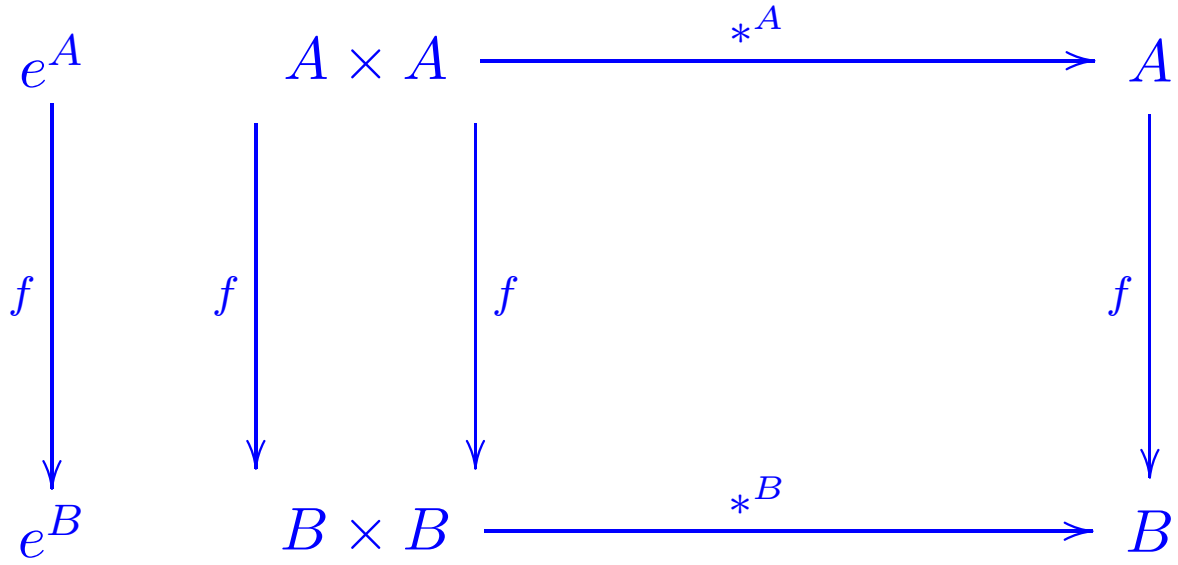
$$f : A \longrightarrow B, \qquad a \longmapsto f(a).$$

heißt **Homomorphismus**, falls:

(i) $f(e^{\mathcal{A}}) = e^{\mathcal{B}}$

(ii) $f(a_1 *^{\mathcal{A}} a_2) = f(a_1) *^{\mathcal{B}} f(a_2)$

(iii) $(a_1, a_2) \in R^{\mathcal{A}} \Rightarrow (f(a_1), f(a_2)) \in R^{\mathcal{B}}$.


e^A
$A \times A$
$*^A$
A
f
f
f
f
e^B
$B \times B$
$*^B$
B

Beispiele für Homomorphismen

1) $f : \Sigma^* \longrightarrow \mathbb{N}$

$w \longmapsto |w|$

Homomorphismus von $(\Sigma^*, \varepsilon, \cdot)$ nach $(\mathbb{N}, 0, +)$.

2) Sei $a \in \Sigma$

$f : \mathbb{N} \longrightarrow \Sigma^*$

$n \longmapsto \underbrace{aaa \ldots a}_{n-times}$

Homomorphismus von $(\mathbb{N}, 0, +)$ nach $(\Sigma^*, \varepsilon, \cdot)$.

3) Sei $f : \Sigma_1 \to \Sigma_2$ eine Abbildung zwischen Alphabeten.
Dann existiert genau ein **Homomorphismus**
$\widehat{f} : (\Sigma_1^*, \varepsilon, \cdot) \to (\Sigma_2^*, \varepsilon, \cdot)$, der f fortsetzt, d.h. mit $\widehat{f}(a) = f(a)$ für alle $a \in \Sigma_1$:

$$\begin{array}{rcl} \hat{f} \colon \Sigma_1^* & \longrightarrow & \Sigma_2^* \\ w = a_1 \dots a_n & \longmapsto & a_1' \dots a_n' \quad \text{mit } a_i' = f(a_i). \end{array}$$

Isomorphismen

Definition: Ein bijektiver Homomorphismus $f : \mathcal{A} \to \mathcal{B}$ zwischen Strukturen $\mathcal{A}, \mathcal{B}$ heißt **Isomomorphismus**, falls auch die Umkehrabbildung f^{-1} ein Homomorphismus $f^{-1} : \mathcal{B} \to \mathcal{A}$. ist.
$\mathcal{A}$ und $\mathcal{B}$ heißen **isomorph**, falls es einen Isomorphismus $f : \mathcal{A} \to \mathcal{B}$ gibt. In Zeichen: $\mathcal{A} \simeq \mathcal{B}$.

Bemerkung: Ein Homomorphismus ist ein Isomorphismus gdw. er eine Bijektion der Trägermengen ist und für Relationen auch die umgekehrte Implikation erfüllt

$$(a_1, a_2) \in R^{\mathcal{A}} \Leftrightarrow (f(a_1), f(a_2)) \in R^{\mathcal{B}}.$$

Bei algebraischen Strukturen (d.h. ohne Relationen) folgt also bereits, dass f^{-1} ebenfalls ein Homomorphismus ist.

Beispiele

- Beispiele ,,1)" und ,,2)" sind nur dann Isomorphismen, wenn $\Sigma = \{a\}$ aus einem Buchstaben besteht.
- Beispiel ,,3)" ist genau dann ein Isomorphismus, wenn Σ_1 und Σ_2 gleich viele Elemente haben und f eine Bijektion ist.

Homomorphismen (bzw. Isomorphismen) $\mathcal{A} \to \mathcal{A}$ heißen auch **Endomorphismen** (bzw. **Automorphismen**).

Aussagenlogische Verknüpfungen

„1" und „0" stehen für die Wahrheitswerte **„wahr"** und **„falsch"**.

Aussagenlogische Operatoren:

- Konjunktion: $\wedge$ **(und)**
- Disjunktion: $\vee$ **(oder)** (nicht ausschließend)
- Negation: $\neg$ **(nicht)**
- Implikation: $\rightarrow$ **(wenn, dann)**
- Äquivalenz: $\leftrightarrow$ **(genau dann, wenn).**

Aussagenlogisch komplexe Sätze

Seien p, q, r etc. Symbole für (atomare) Sätze.

Mit $\wedge, \vee, \neg, \rightarrow, \leftrightarrow, (,)$ werden hieraus komplexe Sätze gebildet.

Konvention: $\neg$ bindet stärker als $\wedge, \vee$, die ihrerseits stärker binden als $\rightarrow, \leftrightarrow$.

Beispiel:

$$p \wedge (\neg q \vee r) \rightarrow (p \wedge \neg q) \vee (p \wedge r)$$

steht für

$$(p \wedge ((\neg q) \vee r)) \rightarrow ((p \wedge (\neg q)) \vee (p \wedge r)).$$

Die Wahrheitswerte komplexer Sätze sind Funktionen der Wahrheitswerte ihrer atomaren Bestandteile:

$\wedge : \mathbb{B} \times \mathbb{B} \to \mathbb{B} \qquad (p, q) \mapsto (p \wedge q)$

$\wedge$	$q = 0$	$q = 1$
$p = 0$	0	0
$p = 1$	0	1

$\vee : \mathbb{B} \times \mathbb{B} \to \mathbb{B} \qquad (p, q) \mapsto (p \vee q)$

$\vee$	$q = 0$	$q = 1$
$p = 0$	0	1
$p = 1$	1	1

$\neg : \mathbb{B} \to \mathbb{B} \qquad p \mapsto (\neg p)$

p	0	1
$\neg p$	1	0

$\rightarrow : \mathbb{B} \times \mathbb{B} \rightarrow \mathbb{B} \qquad (p, q) \mapsto (p \rightarrow q)$

$\rightarrow$	$q = 0$	$q = 1$
$p = 0$	1	1
$p = 1$	0	1

$\leftrightarrow : \mathbb{B} \times \mathbb{B} \rightarrow \mathbb{B} \qquad (p, q) \mapsto (p \leftrightarrow q)$

$\leftrightarrow$	$q = 0$	$q = 1$
$p = 0$	1	0
$p = 1$	0	1

Definition: Zwei komplexe Sätze S_1, S_2, die aus atomaren Sätzen $p_1, \ldots, p_n$ mit $\wedge, \vee, \neg, \rightarrow, \leftrightarrow$ aufgebaut sind, heißen **logisch äquivalent** („$S_1 \equiv S_2$"), falls unter jeder Belegung

$$\mathcal{B} : \{p_1, \ldots, p_n\} \rightarrow \mathbb{B}$$

S_1 und S_2 denselben Wahrheitswert erhalten.

Satz:

1) $p \vee q \equiv \neg(\neg p \wedge \neg q)$ und $p \wedge q \equiv \neg(\neg p \vee \neg q)$.

2) $p \rightarrow q \equiv \neg p \vee q \equiv \neg(p \wedge \neg q)$.

3) $p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p)$.

4) $\neg p \rightarrow 0 \equiv p$ (**Beweis durch Widerspruch**)

5) $p \rightarrow q \equiv \neg q \rightarrow \neg p$ (**Beweis durch Kontraposition**).

6) $p \vee \neg p \equiv 1$ (**Satz vom ausgeschlossenen Dritten**).

7) $p \wedge \neg p \equiv 0$ (**Satz vom Widerspruch**).

Insbesondere: $\vee, \rightarrow, \leftrightarrow$ sind bereits aus $\wedge, \neg$ definierbar.
Tatsächlich sind **alle** $2^{(2^n)}$ vielen n-stelligen Booleschen Operationen $B : \mathbb{B}^n \rightarrow \mathbb{B}$ aus $\wedge, \neg$ definierbar!

Quantorenlogik

Durch Relationen, Konstanten, Funktionen und Variablen lassen sich Sätze wie $x \geq 5 \rightarrow 2x \geq 7$ oder $x^2 = x$ bilden, die ihrerseits mit aussagenlogischen Operatoren verbunden werden können. Zu wahrheitsfähigen Aussagen werden diese, wenn die Variablen durch Quantoren $(\forall x \in A)$ („für alle x in A gilt") und $(\exists y \in B)$ („es existiert mindestens ein y in B mit") gebunden werden:

$$(1)\ (\forall x \in \mathbb{N})\,(x \geq 5 \rightarrow 2x \geq 7),$$

$$(2)\ (\exists x \in \mathbb{N})\,(x^2 = x),$$

$$(3)\ (\forall x \in \mathbb{N})(\exists y \in \mathbb{N})\,(y > x),$$

$$(4)\ (\exists y \in \mathbb{N})(\forall x \in \mathbb{N})\,(y > x).$$

$(1), (2), (3)$ sind über $\mathbb{N}$ wahr, während (4) falsch ist.

Satz: $\forall$ und $\exists$ sind mittels $\neg$ wechselseitig definierbar:

1) $(\forall x \in M)\, A(x)$ gilt genau dann, wenn $\neg(\exists x \in M)\neg A(x)$ gilt.

2) $(\exists x \in M)\, A(x)$ gilt genau dann, wenn $\neg(\forall x \in M)\neg A(x)$.

Sätze der Form $(\exists x \in M)\, A(x)$ kann man durch Angabe eines Beispiels x nachweisen.

All-Sätze $(\forall x \in M) A(x)$ kann man durch Angabe eines Beispiels widerlegen aber niemals durch Beispiele beweisen.

Beweise durch Induktion

Um All-Sätze $(\forall n \in \mathbb{N})A(n)$ über den natürlichen Zahlen zu beweisen, kann man oft **(vollständige) Induktion** verwenden:

Satz: Um $(\forall n \in \mathbb{N})A(n)$ zu beweisen, reicht es zu zeigen:

(i) $A(0)$ ist wahr (**Induktionsanfang**) **und**

(ii) Für jedes $n \in \mathbb{N}$ gilt $A(n) \rightarrow A(n+1)$ (**Induktionsschritt**).

Beispiel eines Induktionsbeweises

Satz: Für alle $n \in \mathbb{N}$ ist $n^3 - n$ durch 6 teilbar, in Zeichen: $6|(n^3 - n)$.

Beweis: Induktion nach n mit $A(n) :\equiv 6|(n^3 - n)$.

Induktionsanfang: Sei $n = 0$. $6|(0^3 - 0)$ ist klar.

Induktionsschritt: Sei $n \in \mathbb{N}$ beliebig.

Wir nehmen an, dass $6|(n^3 - n)$ gilt (**Induktionsannahme**).

Zu zeigen

$$6|\big((n+1)^3 - (n+1)\big).$$

Dies folgt aber aus:

$$\begin{aligned}(n+1)^3-(n+1) &= n^3+3n^2+3n+1-n-1\\ &= (n^3-n)+(3n^2+3n)\\ &= (n^3-n)+3n(n+1),\end{aligned}$$

da nach Induktionsannahme $6|(n^3-n)$ und auch $6|(3n(n+1))$, da $n(n+1)$ stets gerade ist. $\square$

Variationen+Warnung

Manchmal gilt eine All-Aussage erst von einem $n_0 \in \mathbb{N}$ an, d.h. $(\forall n \geq n_0)\, A(n)$. Um dies durch Induktion zu beweisen, genügt es zu zeigen

$$A(n_0) \textbf{ and } (\forall n \geq n_0)\ (A(n) \rightarrow A(n+1)).$$

Warnung: Entscheidend ist, dass der Induktionsschritt wirklich von n_0 angefangen funktioniert. Falls der Induktionsschritt erst ab z.B. $n_0 + 2$ funktioniert, muss man $A(n_0 + 1)$ und $A(n_0 + 2)$ zusätzlich verifizieren!

Beispiel eines inkorrekten Induktionsbeweises

„Satz": Jede Gruppe von n Personen besteht aus gleichaltrigen Personen.

„Beweis": Für $n = 0$ (leere Gruppe) und $n = 1$ ist dies klar. Induktionschritt: wir nehmen an, dass die Behauptung für jede Gruppe von $n > 0$ Personen gilt. Sei nun P eine Gruppe von $n + 1$ Personen. Wähle $p_1, p_2 \in P$ mit $p_1 \neq p_2$ und betrachte

$$P_1 := P \setminus \{p_1\} \text{ und } P_2 := P \setminus \{p_2\}.$$

Nach Induktionsannahme bestehen beide Gruppen aus gleichaltrigen Personen. Sei nun $p \in P_1 \cap P_2$. Dann haben alle Personen in P_1 sowie alle Personen in P_2 dasselbe Alter wie p. Also haben alle Personen in $P = P_1 \cup P_2$ dasselbe Alter.

Wertverlaufsinduktion

Satz: Um $(\forall n \in \mathbb{N})\, A(n)$ zu beweisen, reicht es zu zeigen, dass für jedes $n \in \mathbb{N}$ aus $(\forall m < n)\ A(m)$ stets folgt $A(n)$, d.h.

$$(\forall n \in \mathbb{N})\ [(\forall m < n)\, A(m)\ \rightarrow A(n)].$$

Beweis: Übungsaufgabe!

Strukturelle Induktion

M werde, ausgehend von $M_0 \subset M$,
durch Operationen $F \in \mathcal{F}$ erzeugt. Dann lässt sich

$$(\forall x \in M) A(x)$$

beweisen anhand von

(i) **Induktionsanfang**: $A(x)$ gilt für alle $x \in M_0$.

(ii) **Induktionsschritt(e)** für $F \in \mathcal{F}$ (n-stellig):

aus $A(x_i)$ für $i = 1, \ldots, n$ folgt, dass auch $A(F(x_1, \ldots, x_n))$.

Beispiele

Bereich M	$M_0 \subset M$	erzeugende Operationen
$\mathbb{N}$	$\{0\}$	S: $n \longmapsto n+1$
Σ^*	$\{\varepsilon\}$	$(w \longmapsto wa)$ für $a \in \Sigma$
$\{*, c\}$-Terme	$\{c\}$	$(t_1, t_2) \longmapsto (t_1 * t_2)$
endl. Teilmengen von A	$\{\emptyset\}$	$(B \longmapsto B \cup \{a\})$ für $a \in A$

Operationen auf Sprachen

Seien L, L_1, L_2 Sprachen $\subseteq \Sigma^*$ über einem Alphabet Σ.

Boolesche Operationen:

- **Vereinigung:** $(L_1, L_2) \mapsto L_1 \cup L_2$.
- **Durchschnitt:** $(L_1, L_2) \mapsto L_1 \cap L_2$.
- **Komplement:** $L \mapsto \overline{L} := \Sigma^* \setminus L$.

Konkatenation von Sprachen:

$$(L_1, L_2) \mapsto L_1 \cdot L_2 := \{u \cdot v \,:\, u \in L_1,\, v \in L_2\}.$$

Stern-Operation:

$$L \mapsto L^* := \{u_1 \cdot \ldots \cdot u_n \,:\, u_1, \ldots, u_n \in L, n \in \mathbb{N}\}.$$

Reguläre Ausdrücke

Die Menge **REG(Σ)** der **regulären Ausdrücke** über Σ ist induktiv erzeugt durch:

(i) $\emptyset$ ist ein regulärer Ausdruck.

(ii) Für jedes $a \in \Sigma$ ist $\mathbf{a}$ ein regulärer Ausdruck.

(iii) Mit $\alpha, \beta \in \mathrm{REG}(\Sigma)$ ist auch $(\alpha + \beta) \in \mathrm{REG}(\Sigma)$.

(iv) Mit $\alpha, \beta \in \mathrm{REG}(\Sigma)$ ist auch $(\alpha\beta) \in \mathrm{REG}(\Sigma)$.

(v) Mit $\alpha \in \mathrm{REG}(\Sigma)$ ist auch $\alpha^* \in \mathrm{REG}(\Sigma)$.

Reguläre Sprachen

Jedem regulären Ausdruck (Syntax) wird induktiv eine Sprache $L(\alpha) \subseteq \Sigma^*$ zugeordnet (Semantik).

$L(\alpha)$ heißt die **durch α erzeugte reguläre Sprache**:

(i) $L(\emptyset) := \emptyset$.

(ii) $L(a) := \{a\}$.

(iii) $L(\alpha + \beta) := L(\alpha) \cup L(\beta)$.

(iv) $L(\alpha\beta) := L(\alpha) \cdot L(\beta)$.

(v) $L(\alpha^*) := \big(L(\alpha)\big)^*$.

Definition: Eine Sprache $L \subseteq \Sigma^*$ heißt **regulär**, wenn es ein $\alpha \in \mathsf{REG}(\Sigma)$ gibt mit $L = L(\alpha)$.

Beispiele regulärer Sprachen I

Beispiel 1: Sei $\Sigma := \{0, 1\}$. Dann ist die Sprache L der Wörter über Σ, mit genau einer 1 regulär, da

$$L = L(0^*10^*).$$

Das Komplement $\overline{L}$ von L ist ebenfalls regulär, da

$$\overline{L} = \underbrace{0^*}_{\text{,,keine 1"}} + \underbrace{(0+1)^*1(0+1)^*1(0+1)^*}_{\text{,,mindestens zwei 1"}}.$$

Beispiele regulärer Sprachen II

Beispiel 2: Sei $\Sigma := \{a_1, \ldots, a_n\}$. Dann sind die folgenden Sprachen regulär:

(i) $L(\emptyset^*) = \{\varepsilon\}$.

(ii) $\Sigma = L(a_1 + \ldots + a_n)$.

(iii) $\Sigma^* = L((a_1 + \ldots + a_n)^*)$.

(iv) $\Sigma^+ = L((a_1 + \ldots + a_n)(a_1 + \ldots + a_n)^*)$.

Zusammenfassung

Die regulären Sprachen werden ausgehend von den Sprachen $\emptyset$ und $\{a\}$ durch die Operationen Vereinigung, Konkatenation und Stern erzeugt.

Als Folgerung eines Satzes von Kleene werden wir später einsehen, dass die regulären Sprachen auch unter Durchschnitt und Komplement abgeschlossen sind.

Transitionssysteme

Transitionssysteme (mit endl. Zustandsmenge)

$\mathcal{S} = (\Sigma, Q, \Delta)$ mit den Komponenten:

Σ: Alphabet (Kantenbeschriftungen)

Q: Zustandsmenge, endlich, $\neq \emptyset$

$\Delta \subset Q \times \Sigma \times Q$: Transitionsrelation

$(q, a, q') \in \Delta$ steht für die Transition $q \xrightarrow{a} q'$.

Deterministische endliche Automaten: DFA

Transitionssysteme zur **Erkennung von Sprachen**

Deterministic finite automata: DFA

Transitionssystem bei dem die

Transitionsrelation $\Delta \subset Q \times \Sigma \times Q$ eine

Transitionsfunktion

$$\begin{aligned} \delta \colon Q \times \Sigma &\longrightarrow Q \\ (q, a) &\longmapsto \delta(q, a) \in Q \end{aligned}$$

ist, d.h. stets ein **eindeutig bestimmter** Nachfolgezustand.

Also: **kein deadlock, keine Auswahl!**

Zusatzstruktur

Ein endlicher deterministischer Automat $\mathcal{A}$ besitzt einen ausgezeichneten **Angangszustand** $q_0 \in Q$ und eine **Menge** $A \subseteq Q$ **akzeptierender Zustände**.

Also

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

Q	endliche, nicht-leere **Zustandsmenge**
$q_0 \in Q$	**Anfangszustand**
$A \subseteq Q$	Menge der **akzeptierenden Zustände**
$\delta\colon Q \times \Sigma \to Q$	**Übergangsfunktion.**

Berechnung von $\mathcal{A}$ auf $w = a_1 \dots a_n \in \Sigma^*$

Zustandsfolge $q_0, \dots, q_n$ mit $q_{i+1} = \delta(q_i, a_{i+1})$ für $0 \leq i < n$

$$q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots \; q_{n-1} \xrightarrow{a_n} q_n$$

analog: *Lauf* von $q \in Q$ aus (nicht notw. von q_0 aus)

führt zu eindeutiger Fortsetzung $\hat{\delta}$ von δ:

$$\begin{aligned} \hat{\delta}\colon Q \times \Sigma^* &\longrightarrow Q \\ (q, w) &\longmapsto \hat{\delta}(q, w) \in Q \end{aligned}$$

der (!) Endzustand des Laufs auf w von q aus

Induktiv definiert:

$$\hat{\delta}(q, \varepsilon) := q, \quad \hat{\delta}(q, wa) := \delta(\hat{\delta}(q, w), a).$$

DFA: von $\mathcal{A}$ erkannte/akzeptierte Sprache

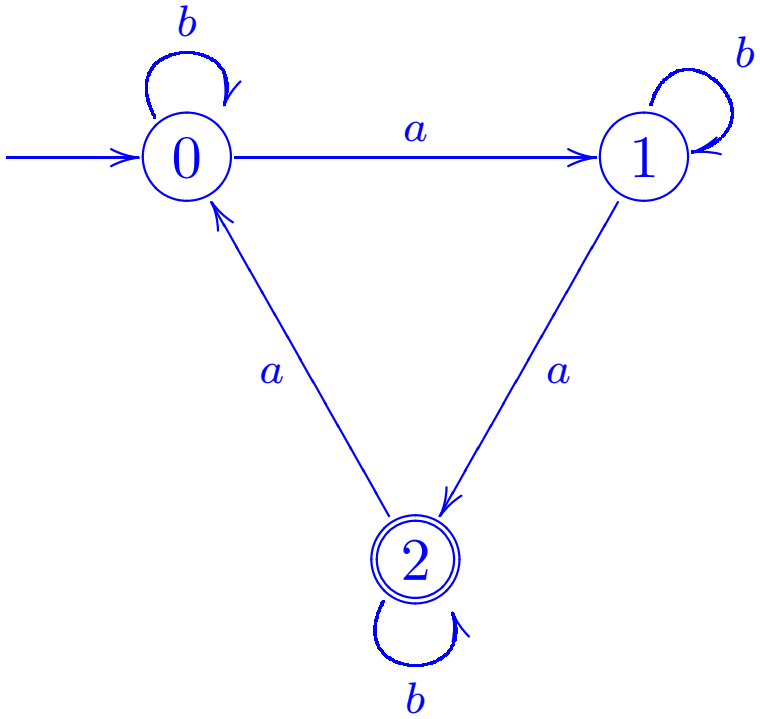
$w = a_1 \dots a_n$ mit Berechnung $q_0, \dots, q_n$ $\qquad q_n = \hat{\delta}(q_0, w)$

$\mathcal{A}$ $\begin{cases} \textbf{akzeptiert } w & \text{falls } q_n \in A \\ \textbf{verwirft } w & \text{falls } q_n \notin A \end{cases}$

die von $\mathcal{A}$ **akzeptierte/erkannte Sprache**:

$$\begin{aligned} L(\mathcal{A}) &:= \{w \in \Sigma^* : \ \mathcal{A} \text{ akzeptiert } w \ \} \\ &= \{w \in \Sigma^* : \hat{\delta}(q_0, w) \in A\} \end{aligned}$$

Beispiel: DFA für $|w|_a$ mod 3



modulo-3 Zähler für a, $|w|_a \bmod 3$.

Zusatzstruktur:

$\rightarrow$: Initialisierung; $\textcircled{\textcircled{2}}$: Endzustand für $|w|_a \equiv 2 \pmod 3$

Nicht-deterministische endliche Automaten, NFA

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

Q	endliche, nicht-leere Zustandsmenge
$q_0 \in Q$	Anfangszustand
$A \subseteq Q$	Menge der akzeptierenden Zust"ande
$\Delta \subseteq Q \times \Sigma \times Q$	"Ubergangs*relation*.

Berechnung von $\mathcal{A}$ auf $w = a_1 \dots a_n \in \Sigma^*$

jede (!) Zustandsfolge $q_0, \dots, q_n$ mit $(q_i, a_{i+1}, q_{i+1}) \in \Delta$

$$q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots \; q_{n-1} \xrightarrow{a_n} q_n$$

Warnung: i.a. nicht eindeutig, nicht notwendig existent!

NFA: von $\mathcal{A}$ erkannte/akzeptierte Sprache

Definition: Eine Berechnung $q_0, \ldots, q_n$ von $\mathcal{A}$ auf $w = a_1 \ldots a_n$ ist eine **akzeptierende Berechnung** auf w falls $q_n \in A$

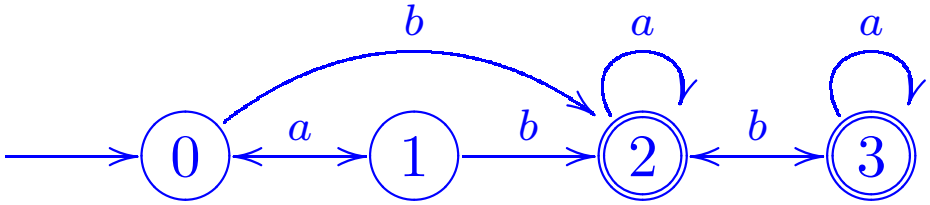
die von $\mathcal{A}$ **akzeptierte/erkannte Sprache**:

$L(\mathcal{A}) := \{w \in \Sigma^* : \ \mathcal{A}$ **hat** akzeptierende Berechnung auf $w \}$

Also: Existenz mindestens einer akzeptierenden Berechnung verlangt, d.h. Asymmetrie bzgl. akzeptieren/verwerfen.

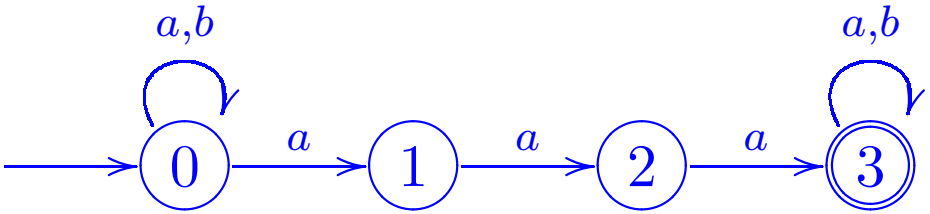
Beispiele mit $\Sigma = \{a, b\}$

DFA $\mathcal{A}_1$



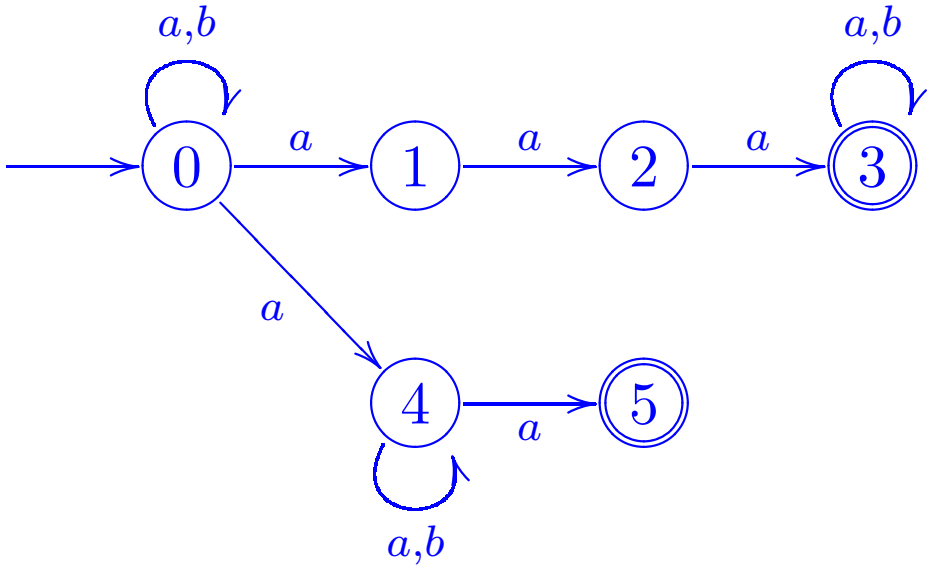
$L(\mathcal{A}_1) = L(a^*b(a+b)^*)$

NFA $\mathcal{A}_2$



$L(\mathcal{A}_2) = L\big((a+b)^*aaa(a+b)^*\big)$

NFA $\mathcal{A}_3$



$$L(\mathcal{A}_3) = L\big((a+b)^*\big(aaa(a+b)^* + a(a+b)^*a\big)\big)$$

Reduktion von NFA zu äquivalentem DFA; Determinisierung

Der **Potenzmengen-Trick**

Satz: Zu NFA $\mathcal{A}$ lässt sich effektiv ein DFA $\mathcal{A}^{\mathsf{det}}$ konstruieren mit

$$L(\mathcal{A}) = L(\mathcal{A}^{\mathsf{det}}).$$

Idee: Zustände von $\mathcal{A}^{\mathsf{det}}$ sind die Mengen der möglichen Zustände von $\mathcal{A}$.

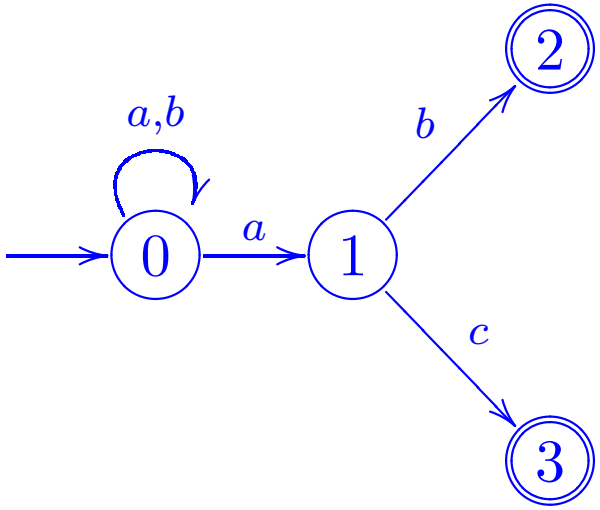
	$\mathcal{A} = (\Sigma, Q, q_0, \Delta, A)$	$\mathcal{A}^{\text{det}} = (\Sigma, \hat{Q}, \hat{q}_0, \delta, \hat{A})$
Zust.	$q \in Q$	$S \subseteq Q$: $\quad \hat{Q} := \mathcal{P}(Q)$
Start	q_0	$\hat{q}_0 := \{q_0\}$
akz.	A	$\hat{A} := \{S \colon S \cap A \neq \emptyset\}$
Trans.	$\Delta \subseteq Q \times \Sigma \times Q$	$\delta \colon \hat{Q} \times \Sigma \to \hat{Q}$

$\delta(S, a) = \{q' \in Q \colon (q, a, q') \in \Delta$ für mindestens ein $q \in S\}$

(Erinnerung: $\mathcal{P}(Q) = \{S : S \subseteq Q\}$)

Beispiel: Determinisierung durch Potenzmengen-Automat

NFA $\mathcal{A}$ mit $\Sigma = \{a, b, c\}$



$L(\mathcal{A}) = L((a + b)^* a(b + c))$.

DFA $\mathcal{A}^{\mathsf{det}}$ mit $L(\mathcal{A}^{\mathsf{det}}) = L(\mathcal{A})$:

δ	a	b	c
$\{0\}$	$\{0,1\}$	$\{0\}$	$\emptyset$
$\{0,1\}$	$\{0,1\}$	$\{0,2\}$	$\{3\}$
$\{0,2\}$	$\{0,1\}$	$\{0\}$	$\emptyset$
$\{3\}$	$\emptyset$	$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

(aktive) Zustände: $\{0\}, \{0,1\}, \{0,2\}, \{3\}, \emptyset$

akzeptierende Zustände: $\{0,2\}$ und $\{3\}$

Abschlusseigenschaften für NFA/DFA erkennbare Sprachen

Vereinigung: zu DFA $\mathcal{A}_1, \mathcal{A}_2$ existiert DFA $\mathcal{A}$ mit $L(\mathcal{A}) = L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$.

Durchschnitt: zu DFA $\mathcal{A}_1, \mathcal{A}_2$ existiert DFA $\mathcal{A}$ mit $L(\mathcal{A}) = L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$.

Komplement: zu DFA $\mathcal{A}_1$ existiert DFA $\mathcal{A}$ mit $L(\mathcal{A}) = \overline{L(\mathcal{A}_1)}$.

Konkatenation: zu NFA $\mathcal{A}_1, \mathcal{A}_2$ existiert NFA $\mathcal{A}$ mit $L(\mathcal{A}) = L(\mathcal{A}_1) \cdot L(\mathcal{A}_2)$. Mit obiger Reduktion gilt dies auch für DFA.

Stern-Operation: Zu NFA $\mathcal{A}_1$ existiert NFA $\mathcal{A}$ mit $L(\mathcal{A}) = \big(L(\mathcal{A}_1)\big)^*$.

Durchschnitt und Vereinigung (für DFA)

zu $\mathcal{A}_1 = \left(\Sigma, Q^{(1)}, q_0^{(1)}, \delta^{(1)}, A^{(1)}\right)$ und $\mathcal{A}_2 = \left(\Sigma, Q^{(2)}, q_0^{(2)}, \delta^{(2)}, A^{(2)}\right)$ konstruiere **Produktautomat** $\mathcal{A} = \left(\Sigma, Q, q_0, \delta, A\right)$ mit

$$Q := Q^{(1)} \times Q^{(2)}$$

$$q_0 := (q_0^{(1)}, q_0^{(2)})$$

$$\delta\big((q_1, q_2), a\big) := \big(\delta^{(1)}(q_1, a), \delta^{(2)}(q_2, a)\big)$$

$$A := \begin{cases} A^{(1)} \times A^{(2)} & \text{für Durchschnitt} \\ (A^{(1)} \times Q^{(2)}) \cup (Q^{(1)} \times A^{(2)}) & \text{für Vereinigung} \end{cases}$$

$\mathcal{A}$ simuliert $\mathcal{A}_1/\mathcal{A}_2$ **parallel** in erster/zweiter Komponente der Zustände.

Komplement für DFA

Sei

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

ein DFA.

Definiere

$$\mathcal{A}_1 := (\Sigma, Q, q_0, \delta, Q \setminus A).$$

Dann gilt

$$L(\mathcal{A}_1) = \overline{L(\mathcal{A})}.$$

Konkatenation (für NFA)

Seien NFA $\mathcal{A}_1 = \left(\Sigma, Q^{(1)}, q_0^{(1)}, \Delta^{(1)}, A^{(1)}\right)$

$\mathcal{A}_2 = \left(\Sigma, Q^{(2)}, q_0^{(2)}, \Delta^{(2)}, A^{(2)}\right)$

mit o.B.d.A. $Q^{(1)} \cap Q^{(2)} = \emptyset$ und $q_0^{(1)} \notin A^{(1)}$. Dann erhält man die **Hintereinanderschaltung** als NFA $\mathcal{A} = \left(\Sigma, Q, q_0, \delta, A\right)$

$$Q := Q^{(1)} \cup Q^{(2)}$$

$$q_0 := q_0^{(1)}$$

$$A := A^{(2)}$$

$$\Delta := \Delta^{(1)} \cup \Delta^{(2)} \cup \Delta^{(1)\to(2)}$$

$$\Delta^{(1)\to(2)} := \left\{(q, a, q_0^{(2)}) \colon q \in Q^{(1)}, (q, a, q') \in \Delta^{(1)} \text{ für ein } q' \in A^{(1)}\right\}$$

Sternoperation (für NFA)

Sei NFA $\mathcal{A}_1 = \big(\Sigma, Q^{(1)}, q_0^{(1)}, \Delta^{(1)}, A^{(1)}\big)$.

Definiere $\mathcal{A}^+ := \big(\Sigma, Q^{(1)}, q_0^{(1)}, \Delta^+, A^{(1)}\big)$, wobei

$$\Delta^+ := \Delta^{(1)} \cup \big\{(q, a, q_0^{(1)}) : q \in Q^{(1)}, (q, a, q') \in \Delta^{(1)} \text{ für ein } q' \in A^{(1)}\big\}.$$

Falls $q_0^{(1)} \in A^{(1)}$ gilt

$$L(\mathcal{A}^+) = L(\mathcal{A}_1)^*.$$

Sonst

$$L(\mathcal{A}^+) = L(\mathcal{A}_1)^* \setminus \{\varepsilon\}.$$

Theorem: Sei L eine reguläre Sprache. Dann wird L von einem geeigneten DFA erkannt.

Beweis: Durch Induktion über reguläre Ausdrücke zeigt man:

$\big(\forall \alpha \in \mathrm{REG}(\Sigma)\big)$ $(L(\alpha)$ Automaten-erkennbar).

Induktionsanfang: $\alpha = \emptyset$ und $\alpha = \mathbf{a}$ für $a \in \Sigma$.
$L(\emptyset) = \emptyset$ und $L(\mathbf{a}) = \{a\}$ Automaten-erkennbar.

Induktionsschritte: von α_1, α_2 zu $\alpha_1 + \alpha_2$, $\alpha_1\alpha_2$, ${\alpha_1}^*$:
Wenn $L(\alpha_1)$, $L(\alpha_2)$ Automaten-erkennbar sind, so auch

$L(\alpha_1 + \alpha_2) = L(\alpha_1) \cup L(\alpha_2)$, $L(\alpha_1\alpha_2) = L(\alpha_1) \cdot L(\alpha_2)$,
$L({\alpha_1}^*) = \big(L(\alpha_1)\big)^*$. Dies aber haben wir bereits bewiesen. □

Bislang bewiesene Abschlusseigenschaften

$L(\alpha)$: $\alpha \in \mathrm{REG}(\Sigma)$

$L(\emptyset) = \emptyset$, $L(a) = \{a\}$, ...

$L(\mathcal{A})$: Σ-NFA/DFA $\mathcal{A}$

$\emptyset$, $\{a\}$, ...

abgeschlossen unter		
Vereinigung	$\cup$	ja (triv)
Konkatenation	$\cdot$	ja (triv)
Stern-Operation	*	ja (triv)
Durchschnitt	$\cap$	**?**
Komplement	$^-$	**?**

abgeschlossen unter		
Vereinigung	$\cup$	ja
Konkatenation	$\cdot$	ja
Stern-Operation	*	ja
Durchschnitt	$\cap$	ja
Komplement	$^-$	ja

Satz von Kleene:

Eine Σ-Sprache L ist genau dann DFA/NFA-erkennbar, wenn L regulär ist.

Zum Beweis des Satzes von Kleene: Dass jede reguläre Sprache DFA/NFA-erkennbar ist, haben wir bereits bewiesen.

Die Umkehrung wird bewiesen, in dem wir zu jeder DFA-erkennbaren Sprache L einen regulären Ausdruck α mit $L = L(\alpha)$ konstruieren:

Idee: Rekursive Berechnung von α ausgehend von endlichen und daher regulären Sprachen.

Betrachte DFA $\mathcal{A} = (\Sigma, Q, q_0, \delta, A)$ mit $Q = \{1, \ldots, n\}$.

Zu $0 \leq k \leq n$ und $1 \leq \ell, m \leq n$ sei

$$L^k_{\ell,m} := \left\{ w \in \Sigma^* : \begin{array}{l} \mathcal{A} \text{ hat Lauf von Zustand } \ell \text{ nach Zustand } m \\ \text{auf } w \text{ über Zwischenzustände } q \in \{1, \ldots, k\} \end{array} \right\}$$

$$L^0_{\ell,m} = \begin{cases} \{a \in \Sigma : \delta(\ell, a) = m\} & \text{falls } \ell \neq m \\ \{\varepsilon\} \cup \{a \in \Sigma : \delta(\ell, a) = \ell\} & \text{falls } \ell = m \end{cases}$$

(endlich)!

$$L^{k+1}_{\ell,m} = \underbrace{L^k_{\ell,m}}_{(1)} \quad \cup \quad \underbrace{L^k_{\ell,k+1}}_{(2)} \cdot \underbrace{(L^k_{k+1,k+1})^*}_{(3)} \cdot \underbrace{L^k_{k+1,m}}_{(4)}$$

(1) Läufe ohne Zustand $k+1$;

(2) Läufe von Zustand ℓ zum ersten $k+1$;

(3) Schleifen durch Zustand $k+1$;

(4) Läufe vom letzten $k+1$ nach m.

Aus den nach Induktionshypothese bereits konstruierten regulären Ausdrücken für der Teilsprachen $L^k_{\ldots}$ kann ein regulärer Ausdruck für $L^{k+1}_{\ell,m}$ wie folgt gebildet werden:

$$\alpha^{k+1}_{\ell,m} = \underbrace{\alpha^k_{\ell,m}}_{(1)} \quad + \quad \underbrace{\alpha^k_{\ell,k+1}}_{(2)} \underbrace{(\alpha^k_{k+1,k+1})^*}_{(3)} \underbrace{\alpha^k_{k+1,m}}_{(4)} .$$

Definiere: $\alpha := \sum_{q \in A} \alpha^n_{q_0,q}$. Dann: $L(\mathcal{A}) = L(\alpha)$. □

Folgerungen aus dem Satz von Kleene

- Die Klasse der regulären Sprachen ist abgeschlossen unter allen Booleschen Operationen sowie Konkatenation und Stern.
- Alle durch endliche Automaten erkennbaren Sprachen lassen sich allein mit Vereinigung, Konkatenation und Stern aus (einfachsten) endlichen Sprachen gewinnen.

Wieviele Zustände sind notwendig?

Zustandszahlen von DFA $\mathcal{A}$ mit $L = L(\mathcal{A})$
als **Maß für die Komplexität** von L.

Grundidee zur Konstruktion von **minimalem** DFA für L:

jeder Zustand beschreibt notwendige Information; verschiedene Zustände korrespondieren zu notwendigen Unterscheidungen.

Methode: definiere geeignete **Äquivalenzrelationen** auf Σ^*

- $\sim_L$ zu gegebenem L $\quad w \not\sim_L w'$: "notwendige Unterscheidung"
- $\sim_{\mathcal{A}}$ zu gegebenem $\mathcal{A}$ $\quad w \not\sim_{\mathcal{A}} w'$: "verschiedene Berechnungen"

Die Äquivalenzrelation $\sim_L$

Sei $L \subseteq \Sigma^*$.

Definition:

$$w \sim_L w' \quad \text{gdw} \quad (\forall x \in \Sigma^*)\,(wx \in L \Leftrightarrow w'x \in L).$$

Eigenschaften:

1) $\sim_L$ ist **Äquivalenzrelation auf** Σ^*.
2) $\sim_L$ ist **rechts-invariant**: $w \sim_L w' \Rightarrow wu \sim_L w'u$.
3) $L \sim_L$ **abgeschl. unter** $\sim_L$: $w \in L \wedge w \sim_L w' \rightarrow w' \in L$.
4) $L = \bigcup_{w \in L} [w]_{\sim_L}$.

Gegeben sei ein DFA $\mathcal{A} = (\Sigma, Q, q_0, \delta, A)$.

Definition:

$$w \sim_{\mathcal{A}} w' \quad \text{gdw} \quad \hat{\delta}(q_0, w) = \hat{\delta}(q_0, w').$$

Eigenschaften:

1) $\sim_{\mathcal{A}}$ ist **Äquivalenzrelation auf Σ^*.**
2) $\sim_{\mathcal{A}}$ ist **rechts-invariant**: $w \sim_{\mathcal{A}} w' \Rightarrow wu \sim_{\mathcal{A}} w'u$.
3) $\sim_{\mathcal{A}}$ hat **endlichen Index**: $\text{index}(\sim_{\mathcal{A}}) \leq |Q|$.
4) Für $L = L(\mathcal{A})$ ist $\sim_{\mathcal{A}}$ eine **Verfeinerung** von $\sim_L$:
   $(\forall w, w' \in \Sigma^*)\ w \sim_{\mathcal{A}} w' \Rightarrow w \sim_L w'$.

Korollar: Sei $L = L(\mathcal{A})$ für DFA $\mathcal{A} = (\Sigma, Q, q_0, \delta, A)$.
Dann gilt: $\text{index}(\sim_L) \leq \text{index}(\sim_{\mathcal{A}}) \leq |Q|$.

Zusammenfassung der bisherigen Ergebnisse

- L regulär $\Rightarrow$ $\sim_L$ hat endlichen Index
- Für reguläres L: jeder DFA, der L erkennt, hat mindestens $\mathrm{index}(\sim_L)$ viele Zustände

Ziel: Satz von Myhill-Nerode

- $\sim_L$ hat endlichen Index $\Leftrightarrow$ L regulär.
- Für reguläres L existiert DFA mit genau $\mathrm{index}(\sim_L)$-vielen Zuständen für L.

Der Äquivalenzklassen-Automat

Gegeben: Sei $L \subseteq \Sigma^*$ mit $\text{index}(\sim_L)$ **endlich**.

Idee: Konstruktion eines minimalen DFA $\mathcal{A}$, der L erkennt.

$[w]_L := \{v \in \Sigma^* : v \sim_L w\}$ ($\sim_L$-Äquivalenzklasse von w).

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

$$Q := \Sigma^*/\sim_L$$

$$q_0 := [\varepsilon]_L$$

$$\delta([w]_L, a) := [wa]_L \text{ (wohldefiniert, da } \sim_L \text{ rechtsinvariant!)}$$

$$A := \{[w]_L : w \in L\}$$

$L = L(\mathcal{A})$ folgt aus: $(\forall w \in \Sigma^*)\hat{\delta}(q_0, w) = [w]_L$ (Induktion!)

Insbesondere ist also L regulär.

Satz von Myhill-Nerode

Theorem (Myhill-Nerode): Sei L eine Σ-Sprache. Dann sind die folgenden Aussagen äquivalent:

1) $\sim_L$ hat endlichen Index.
2) L ist regulär.

Korollar zum Beweis: Für jede **reguläre** Sprache L gibt es **einen** kleinsten DFA, der L erkennt, d.h. einen mit genau $\text{index}(\sim_L)$ vielen Zuständen.

Korollar zum Satz von Myhill-Nerode: Eine Σ-Sprache L ist genau dann nicht-regulär, wenn es eine Folge $(w_n)_{n\in\mathbb{N}}$ in Σ^* gibt mit $w_n \not\sim_L w_m$ für $n \neq m$.

Anwendung: $L = \{a^n b^n : n \in \mathbb{N}\} \subset \{a,b\}^*$ ist nicht regulär.

Äquivalenzautomat als *der* Minimalautomat

In folgenden zeigen wir, dass der L-Äquivalenzklassenautomat bis auf Isomorphie der einzige DFA ist, der L mit index($\sim_L$)-vielen, d.h. mit minimal vielen, Zuständen erkennt.

Definition: Ein Isomorphismus zwischen DFA's $\mathcal{A}^{(1)} = (\Sigma, Q^{(1)}, q_0^{(1)}, \delta^{(1)}, A^{(1)})$, $\mathcal{A}^{(2)} = (\Sigma, Q^{(2)}, q_0^{(2)}, \delta^{(2)}, A^{(2)})$ ist eine Bijektion $f : Q^{(1)} \to Q^{(2)}$ mit

1) $f(q_0^{(1)}) = q_0^{(2)}$,

2) $f(\delta^{(1)}(q, a)) = \delta^{(2)}(f(q), a)$ für alle $q \in Q^{(1)}, a \in \Sigma$,

3) $f[A^{(1)}] = A^{(2)}$.

Theorem: Jeder DFA mit $\mathrm{index}(\sim_L)$-vielen Zuständen, der L erkennt, ist isomorph zum Äquivalenzklassen-Automat zu $\sim_L$.

Beweis: Wir zeigen zunächst das folgende

Lemma:

Sei $\mathcal{A} = (\Sigma, Q, q_0, \delta, A)$ ein DFA ohne unerreichbare Zustände.

Dann ist **$\mathcal{A}$ isomorph zum Äquivalenzautomaten für $\sim_{\mathcal{A}}$.**

Beweis des Lemmas

Sei $\mathcal{A}' = (\Sigma, Q', q_0', \delta', A')$ mit

$$Q' := \Sigma^*/\sim_{\mathcal{A}} = \{[w]_{\mathcal{A}} : w \in \Sigma^*\},$$

$$q_0' := [\varepsilon]_{\mathcal{A}}$$

$$\delta'([w]_{\mathcal{A}}, a) := [wa]_{\mathcal{A}} \text{ (wohldefiniert, da } \sim_{\mathcal{A}} \text{ rechtsinvariant)}$$

$$A' := \{[w]_{\mathcal{A}} : \widehat{\delta}(q_0, w) \in A\},$$

wobei $[w]_{\mathcal{A}}$ die Äquivalenzklasse von w bzgl. $\sim_{\mathcal{A}}$ ist.

$$f(q) := \{w \in \Sigma^* : \widehat{\delta}(q_0, w) = q\}$$

ist ein Isomorphismus zwischen $\mathcal{A}$ und $\mathcal{A}'$. □

Beweis des Theorems

Da $\mathcal{A}$ die minimale Anzahl index($\sim_L$) von Zuständen hat, sind alle Zustände von $\mathcal{A}$ erreichbar. Nach dem Lemma ist $\mathcal{A}$ daher isomorph zu dem Äquivalenzklassenautomaten zu $\sim_{\mathcal{A}}$.
Aus den $\sim_{\mathcal{A}}, \sim_L$-Eigenschaften folgt, dass index($\sim_{\mathcal{A}}$) $\leq |Q|$ =index($\sim_L$) aber auch, da $\sim_{\mathcal{A}}$ Verfeinerung von $\sim_L$ ist, index($\sim_L$) $\leq$index($\sim_{\mathcal{A}}$). Also index($\sim_{\mathcal{A}}$) =index($\sim_L$) und $\sim_{\mathcal{A}} = \sim_L$. Also ist $\mathcal{A}$ identisch mit dem Äquivalenzklassenautomaten zu $\sim_L$. □

Minimierung eines gegebenen DFA

Wie kann man redundante Zustände erkennen/eliminieren?

- Eliminiere alle nicht-erreichbaren Zustände.
- Dann: $q \not\sim q'$ **wesentlich verschieden**, wenn für ein $x \in \Sigma^*$ **nicht** $(\hat{\delta}(q,x) \in A) \leftrightarrow (\widehat{\delta}(q',x) \in A)$.
- Zusammenfassung von $\sim$-Klassen von Zuständen liefert minimierten DFA, isomorph zum Minimalautomat.

Minimierung algorithmisch: induktive Verfeinerung

Tests für x der Länge $i = 0, 1, \ldots$

$q \not\sim_0 q'$ **gdw.** **nicht**$(q \in A) \Leftrightarrow (q' \in A)$

$q \not\sim_{i+1} q'$ **gdw.** $q \not\sim_i q'$ oder $(\exists a \in \Sigma)\ \delta(q,a) \not\sim_i \delta(q',a)$

Sobald $\sim_{i+1} = \sim_i$ $(=: \sim)$, fasse Zustände in $\sim$-Klassen zusammen.

Das Pumping Lemma

Theorem: Sei $L \subseteq \Sigma^*$ regulär. Dann existiert $n \in \mathbb{N}$ derart, dass sich jedes $x \in L$ mit $|x| \geq n$ zerlegen lässt in $x = uvw$, $v \neq \varepsilon$, $|uv| \leq n$ und für alle $m \in \mathbb{N}$

$$u \cdot v^m \cdot w = u \cdot \underbrace{v \cdots v}_{m \text{ mal}} \cdot w \in L.$$

Beweis: $L = L(\mathcal{A})$, DFA $\mathcal{A} = (\Sigma, Q, q_0, \delta, A)$, $n := |Q|$

Nach Schubfachprinzip (da $|x| \geq n$): $q_i = q_j = q$ im Lauf auf x:

$$x = \underbrace{a_1 \quad \cdots \quad a_i}_{u} \underbrace{a_{i+1} \quad \cdots \quad a_j}_{v} \underbrace{a_{j+1} \quad \cdots \quad a_\ell}_{w}.$$

q_0 q q

□

Beispiele nicht-regulärer Sprachen

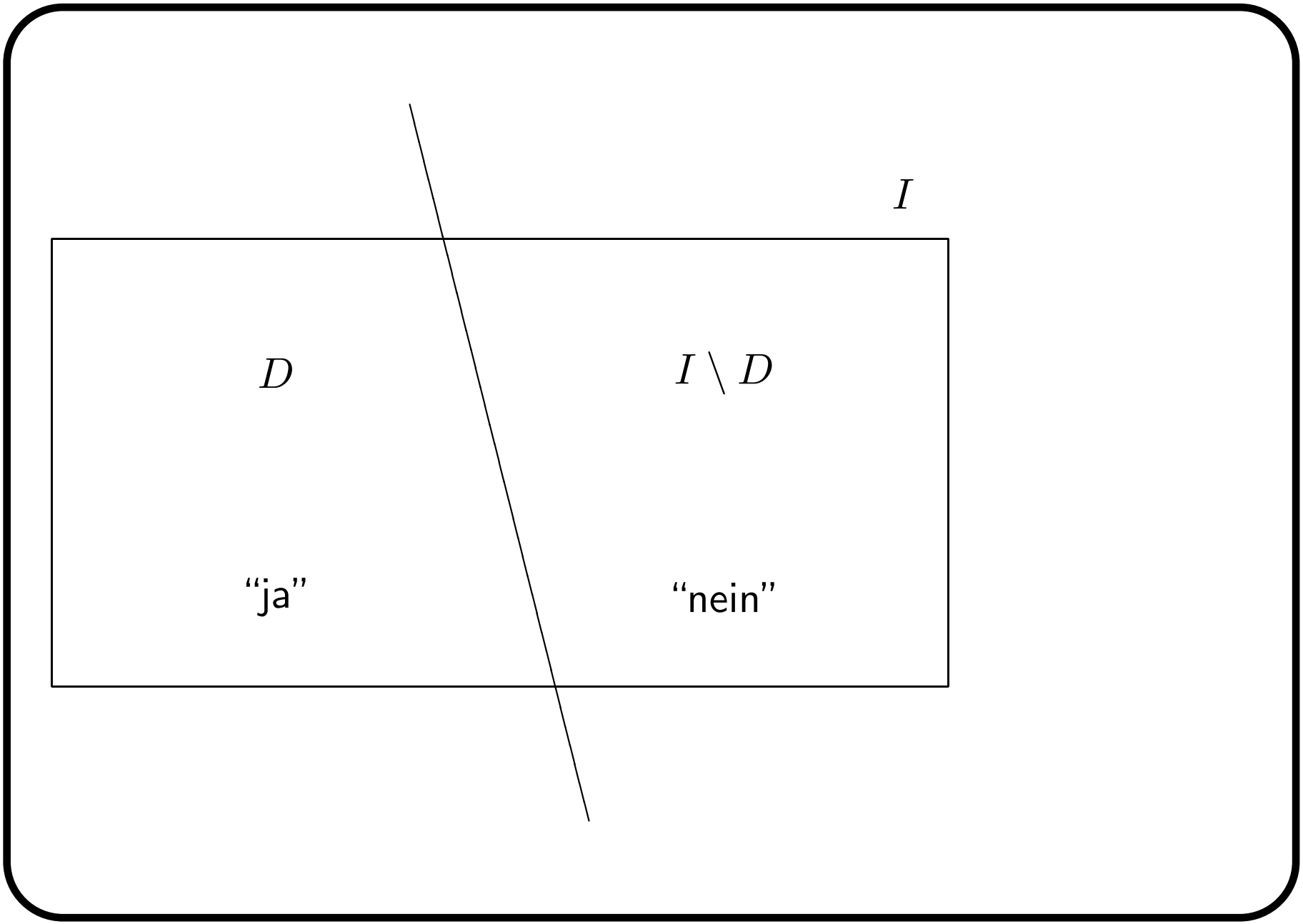
- $L = \{a^n b^n : n \in \mathbb{N}\}$ für $a, b \in \Sigma, a \neq b$.
- $L = \{w \in \{(,)\}^* : w$ **korrekte Klammerschachtelung** $\}$.
- **Palindrom** $= \{w \in \Sigma^* : w = w^{-1}\}$ für $|\Sigma| \geq 2$.
- $L = \{w \in \{1\}^* \ : \ |w|$ ist Primzahl $\}$.

Entscheidungsprobleme

Entscheidungsproblem: (ja/nein Problem) spezifiert durch
Menge von zugelassenen Eingaben $x \in I$ (Instanzen)
Teilmenge $D \subseteq I$ der positiven Instanzen (Antwort "ja")

d.h.: **wohldefinierte** ja/nein Antwort für alle $x \in I$.

Aufgabe: gegeben $x \in I$, entscheide ob $x \in D$.


I
D
$I \setminus D$
“ja”
“nein”

Wichtige Entscheidungsprobleme für reguläre Sprachen

- **Wortproblem:** Sei $L \subseteq \Sigma^*$: für $w \in \Sigma^*$ entscheide, ob $w \in L$ (d.h. $I = \Sigma^*, D = L$).

  Entscheidbar für reguläres L, da $L = L(\mathcal{A})$ durch einen DFA $\mathcal{A}$ erkannt wird:

  $\mathcal{A}$ liefert (lineares) Entscheidungsverfahren für „$w \in L$".

- **Leerheitsproblem für reguläre Sprachen:** für DFA $\mathcal{A}$ entscheide, ob $L(\mathcal{A}) = \emptyset$, d.h.

  $I = \{\mathcal{A}\colon \mathcal{A} \text{ DFA }\}, \quad D = \{\mathcal{A}\colon L(\mathcal{A}) = \emptyset\}$.

  Analog mit $I = \mathrm{REG}(\Sigma)$: zu $\alpha \in$REG konstruiere $\mathcal{A}$ mit $L(\alpha) = L(\mathcal{A})$.

Leerheitsproblem ist **entscheidbar**: $L \neq \emptyset$ gdw. mindestens ein akzeptierender Zustand vom Anfangszustand aus erreichbar (entscheidbar duch Suche im Zustandsgraphen von $\mathcal{A}$).

- **Sprachgleichheit/Automatenäquivalenz**:

  für $\mathcal{A}, \mathcal{B}$ DFAs über Σ. Entscheide, ob $L(\mathcal{A}) = L(\mathcal{B})$.

  $I = \{(\mathcal{A}, \mathcal{B}) : \mathcal{A}, \mathcal{B}$ DFAs über $\Sigma \}$,

  $D = \{(\mathcal{A}, \mathcal{B}) \in I \ : \ L(\mathcal{A}) = L(\mathcal{B}).\}$

  analog für $\alpha, \beta \in \text{Reg}(\Sigma)$. Entscheide, ob $L(\alpha) = L(\beta)$.

Entscheidbarkeit der Sprachgleichheit

Sei $L_1 := L(\mathcal{A}),\ L_2 := L(\mathcal{B})$. Es gilt

$$L_1 = L_2 \Leftrightarrow (L_1 \setminus L_2 = \emptyset = L_2 \setminus L_1).$$

Da

$$L_1 \setminus L_2 = L_1 \cap \overline{L_2} \text{ und } L_2 \setminus L_1 = L_2 \cap \overline{L_1}$$

und DFAs für $L_1 \cap \overline{L_2}$ und $L_2 \cap \overline{L_1}$ in $\mathcal{A}, \mathcal{B}$ berechenbar sind, folgt die Entscheidbarkeit aus der Entscheidbarkeit des Leerheitsproblems.

Grammatiken

Idee: Spezifikation einer Sprache durch **Erzeugungsprozesse**

$G = (\Sigma, V, P, X_0)$

Σ **Terminalalphabet,**

V endliche Menge von **Variablen**, $V \cap \Sigma = \emptyset$,

$X_0 \in V$ **Startvariable/Startsymbol,**

P endliche Menge von **Produktionen/Regeln**

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

Produktionen

$(v, v') \in P$ Produktion/Regel von G: $\boxed{v \to v'}$

linke Seite: $v \in (V \cup \Sigma)^+ = (V \cup \Sigma)^* \setminus \{\varepsilon\}$,

rechte Seite: $v' \in (V \cup \Sigma)^*$.

Erlaubt in $(V \cup \Sigma)$-Wörtern **Ersetzung von v durch v':**

$uvw \to_G uv'w$ (direkter Ableitungsschritt in G).

Ableitbarkeit

Zwei Ableitbarkeitsrelationen: 2-stellig über $(\Sigma \cup V)^*$:

1-Schritt Ableitbarkeit $\to_G$:

$x \to_G x' \;:\Leftrightarrow\; x = uvw,\ x' = uv'w$ wobei $(v, v') \in P$

Ableitbarkeit $\to_G^*$:

$$x \to_G^* x' \;:\Leftrightarrow\; \begin{cases} \text{es existiert ein } \to_G\text{-Pfad von } x \text{ nach } x'\text{:} \\ x = x_1 \to_G x_2 \to_G \ldots \to_G x_n = x'. \end{cases}$$

„endliche Iteration von $\to_G$": reflexiver, transitiver Abschluss

- $w \in (\Sigma \cup V)^*$ **ableitbar in** G gdw. $X_0 \to_G^* w$.
- $L(G) = \{w \in \Sigma^* : X_0 \to_G^* w\}$ die von G erzeugte Σ-Sprache.

Beispiele

1) Palindrom: Sei z.B. $\Sigma = \{0, 1\}$,

$G = (\Sigma, V, P, X)$ mit $V = \{X\}$ und Produktionen

P : $X \to \varepsilon$

$X \to a$ für jedes $a \in \Sigma$

$X \to aXa$ für jedes $a \in \Sigma$

erzeugt die Σ-Sprache der **Σ-Palindrome** ($w \in \Sigma^* : w = w^{-1}$).

2) Klammerungen:

$\Sigma = \{(,)\}$, $G = (\Sigma, V, P, X)$ mit $V = \{X\}$ und Produktionen

$$P: \quad X \to (\ ) \qquad X \to (X) \qquad X \to XX$$

Durch Induktion zeigt man: diese Grammatik erzeugt die **Sprache der korrekt geschachtelten nicht-leeren Klammerwörter**.

Beachte: Die Sprachen in 1) und 2) sind **nicht regulär!**

3) Reguläre Sprachen:

Satz: Jede reguläre Sprache wird von einer Grammatik erzeugt.

Beweis: Zu $L = L(\mathcal{A})$ für NFA $\mathcal{A} = (\Sigma, Q, q_0, \Delta, A)$:

$G_{\mathcal{A}} := (\Sigma, V, P, X_0)$

$$
\begin{aligned}
V &:= \{X_q : q \in Q\}, \\
X_0 &:= X_{q_0}, \\
P &: \quad X_q \to aX_{q'} \quad \text{für } (q, a, q') \in \Delta \\
&\quad\; X_q \to \varepsilon \quad\;\; \text{für } q \in A
\end{aligned}
$$

$L = L(\mathcal{A}) = L(G_{\mathcal{A}})$:

Ableitungen in G simulieren akzeptierende Berechnungen von $\mathcal{A}$.

□

Beispiel $a^n b^n$:

$\Sigma = \{a, b\}$, $G = (\Sigma, V, P, X)$, $V = \{X\}$,

$$P: \quad X \to \varepsilon \qquad X \to aXb$$

erzeugt die Sprache $L = \{a^n b^n : n \in \mathbb{N}\}$. (nicht regulär)

Beispiel $a^n b^n c^n$:

$\Sigma = \{a, b, c\}$, $G = (\Sigma, V, P, X)$, $V = \{X, Y, Z\}$, $X_0 = X$,

$$
\begin{array}{rrl}
P: & X & \to \varepsilon \\
 & X & \to aXYZ \\
 & ZY & \to YZ \\
 & aY & \to ab \\
 & bY & \to bb \\
 & bZ & \to bc \\
 & cZ & \to cc
\end{array}
$$

erzeugt die Sprache $L = \{a^n b^n c^n : n \in \mathbb{N}\}$. (nicht regulär)

Baumdarstellung von Ableitungsschritten

$\Sigma = \{+, \cdot, (,), 0, 1\}$

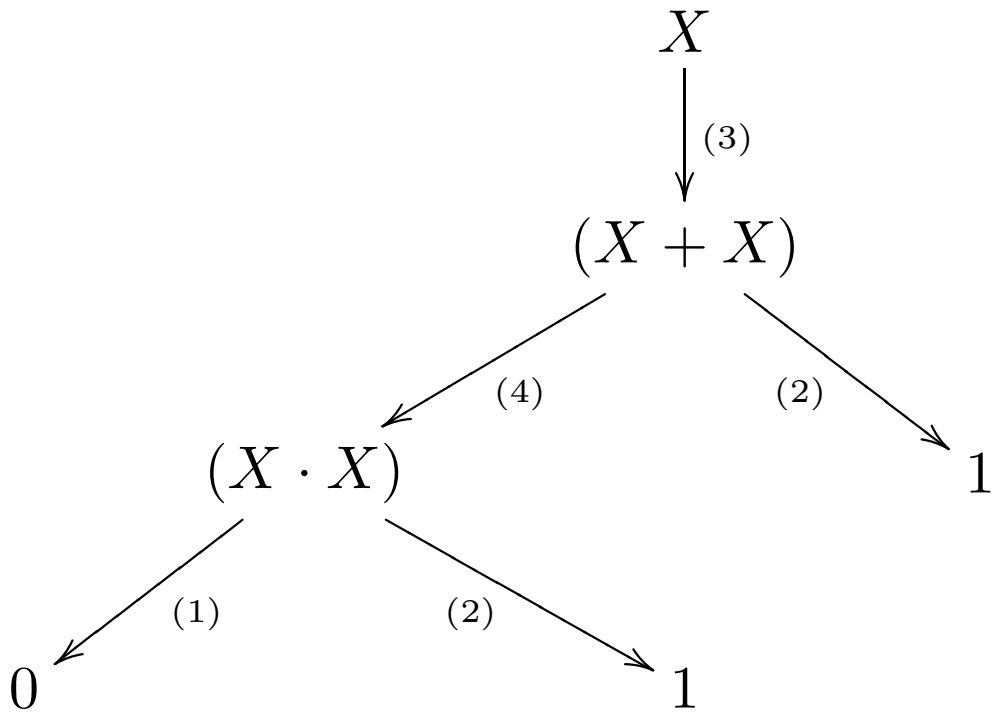
$G = (\Sigma, \{X\}, P, X)$: Baum zu $((0 \cdot 1) + 1) \in L(G)$:

$X \rightarrow 0 \quad (1)$

$X \rightarrow 1 \quad (2)$

$X \rightarrow (X + X) \quad (3)$

$X \rightarrow (X \cdot X) \quad (4)$



Backus-Naur Form, BNF

Kompaktschreibweise

$v \rightarrow v'_1 \mid v'_2 \mid \ldots \mid v'_n$ anstelle von $v \rightarrow v'_1$, $\vdots$, $v \rightarrow v'_n$

lies "|" als "oder"; oft auch ::= anstelle von $\rightarrow$

Erweiterte BNF, EBNF:

Weitere eliminierbare Abkürzungen

$X \rightarrow u[v]w$ für $X \rightarrow uw \mid uvw$

„v **kann** zwischen u und w eingefügt werden"

$X \to u\{v\}w$ für $X \to uw \mid uZw$ [Z **neu!**]

$Z \to ZZ \mid v$

„jedes $v' \in \{v\}^*$ **kann** zwischen u und w eingefügt werden"

Definition:

Wir sagen, dass in einer Grammatik $G = (\Sigma, V, P, X_0)$ die Variable X_0 **nur als Startsymbol auftritt**, wenn X_0 in keiner Produktion $\in P$ auf der rechten Seite vorkommt.

In diesem Fall heißen ε-Produktionen der Form $X_0 \to \varepsilon$ **harmlose ε-Produktionen**.

Chomsky-Hierarchie von Grammatiken G

X, Y Variablen in V, $a \in \Sigma$, $v, v' \in (\Sigma \cup V)^*$

G regulär oder Typ 3

Produktionen **rechtslinear**: $X \to \varepsilon$, $X \to a$, $X \to aY$

G kontextfrei oder Typ 2

alle Produktionen haben als linke Seite

eine **einzelne Variable**: $X \to v$

G kontextsensitiv oder Typ 1

alle Produktionen **nicht verkürzend**: $v \to v'$, $|v'| \geq |v|$

bis auf ggf. eine *harmlose* ε-Produktion

G allgemein oder Typ 0: keine Einschränkungen.

Einfache Beobachtungen

- Reguläre Grammatiken sind insbesondere kontexfrei.
- Bei der Definition regulärer Grammatiken kann man sogar auf Produktionen der Form $X \to a$ verzichten:
  ersetzte $X \to a$ durch $X \to aZ$ und $Z \to \varepsilon$, wobei Z neu ist.

Definition: Eine Sprache $L \subseteq \Sigma^*$ heißt vom **Typ 0**, **kontextsensitiv (Typ 1)** bzw. **kontextfrei (Typ 2)**, wenn es eine Grammatik G mit dieser Eigenschaft gibt und $L = L(G)$.

Wir zeigen nun, dass eine Spache L in dem zuvor definierten Sinne regulär ist gdw. es eine reguläre (Typ 3) Grammatik G gibt mit $L = L(G)$:

Charakterisierung regulärer Sprachen durch reguläre Grammatiken

Theorem: $L \subseteq \Sigma^*$ regulär gdw. $L = L(G)$ für eine reguläre Grammatik G.

Beweis:

"$\Rightarrow$": aus $L = L(\mathcal{A})$ für NFA $\mathcal{A}$ gewinne reguläre Grammatik $G_{\mathcal{A}}$ mit $L(G) = L$:

$$X_q \to aX_{q'} \quad \text{für } (q, a, q') \in \Delta,$$

$$X_q \to \varepsilon \quad \text{für } q \in A.$$

"$\Leftarrow$": konstruiere NFA zu Typ 3 Grammatik $G = (\Sigma, V, P, X_0)$ mit Produktionen $X \to aY$ und $X \to \varepsilon$.

Ableitungen in G

$$X_0 \to_G a_1 X_1 \to_G \cdots \to_G a_1 \dots a_n X_n \to_G a_1 \dots a_n$$

werden simuliert durch $\mathcal{A} := \big(\Sigma, V, X_0, \Delta, A\big)$

mit $\Delta = \big\{(X, a, X') \colon X \to aX' \in P\big\}$,

$A = \big\{X \colon X \to \varepsilon \in P\big\}$.

□

Kontextfrei $\Rightarrow$ kontextsensitiv

Lemma: Jede kontextfreie Grammatik ist äquivalent zu einer kontextfreien Grammatik, die höchstens harmlose ε-Produktionen enthält.

Beweis: Sei $G = (\Sigma, V, P, X_0)$ kontextfrei. Wir können stets zu einer äquivalenten kontextfreien Grammatik G' übergehen, in der X_0 nur als Startsymbol auftritt: führe eine neue Variable X_0' ein und definiere

$$G' := (\Sigma, V \cup \{X_0'\}, P', X_0') \text{ mit } P' := P \cup \{X_0' \to X_0\}.$$

Sei nun $V_\varepsilon \subseteq V$ die Menge der Variablen Y mit $Y \to_G^* \varepsilon$.
Betrachte eine Produktion $X \to uYv$ mit $Y \in V_\varepsilon$. Füge zu P die Produktion $X \to uv$ hinzu.

Hierdurch ändert sich die erzeugte Sprache nicht. Man schließt unter diesen Erweiterungen ab und streicht alle Produktionen der Form $Y \to \varepsilon$ außer (falls $\varepsilon \in L(G)$) $X_0 \to \varepsilon$. □

Korollar: Zu jeder kontextfreien Grammatik gibt es eine äquivalente kontextsensitive Grammatik.

Zusammenfassung:

Typ 3 $\subseteq$ Typ 2 $\subseteq$ Typ 1 $\subseteq$ Typ 0 $\subseteq$ beliebige Sprachen.

Ziel (Chomsky-Hierarchy): alle Inklusionen sind echt, d.h.
Typ 3 $\subset$ Typ 2 $\subset$ Typ 1 $\subset$ Typ 0 $\subset$ beliebige Sprachen.

Chomsky-Hierarchie: Trennung der Klassen

Bereits gezeigt: $\{a^n b^n : n \in \mathbb{N}\}$ **nicht regulär**, aber **kontextfrei (Typ 2)**. Also

$$\text{Typ 3} \subset \text{Typ 2}.$$

Da es nur abzählbar viele Typ 0 Sprachen gibt, aber für jedes Σ überabzählbar viele Σ-Sprachen, folgt

$$\text{Typ 0} \subset \text{beliebige Sprachen}.$$

Im folgenden zeigen wir

$$\{a^n b^n c^n : n \in \mathbb{N}\} \text{ kontextsensitiv, aber nicht kontextfrei,}$$

also

$$\text{Typ 2} \subset \text{Typ 1.}$$

Die folgende Grammatik G' erzeugt $L := \{a^n b^n c^n : n \in \mathbb{N}\}$ und ist kontextsensitiv, also $L \in$ Typ 1:

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

$V = \{X, Y, Z\}$

$$
\begin{array}{rrl}
P: & X & \to \varepsilon \\
& X & \to aXYZ \\
& ZY & \to YZ \\
& aY & \to ab \\
& bY & \to bb \\
& bZ & \to bc \\
& cZ & \to cc
\end{array}
$$

$G' = (\Sigma, V, P, X_0)$

$V = \{X_0, X, Y, Z\}$

$$
\begin{array}{rrl}
P: & X_0 & \to X \mid \varepsilon \\
& X & \to aXYZ \mid aYZ \\
& ZY & \to YZ \\
& aY & \to ab \\
& bY & \to bb \\
& bZ & \to bc \\
& cZ & \to cc
\end{array}
$$

G' ist kontextsensitiv (nur harmlose ε-Produktion!).

Kontextfreie Sprachen (Typ 2)

Ziel: Pumping Lemma für kontextfreie Sprachen (liefert, dass $\{a^n b^n c^n : n \in \mathbb{N}\}$ nicht kontextfrei ist).

Zum Beweis wird benötigt: **Chomsky-Normalform.**

Zulässige Produktionen bei Typ 2: $\varepsilon \notin L$: $X \to v,\ v \neq \varepsilon$

Verschärfung bei Chomsky-Normalform: $X \to YZ$ und $X \to a$

Satz: Jede Typ 2 Grammatik ohne ε-Produktionen ist äquivalent zu einer Gammatik in Chomsky-Normalform (CNF Grammatik). Also wird jede kontextfreie Sprache L mit $\varepsilon \notin L$ von einer CNF Grammatik erzeugt.

Beweis des Chomsky-Normalformsatzes

1. Schritt: Für jedes $a \in \Sigma$ füge neue Variable Z_a zu V hinzu, d.h. $V' := V \cup \{Z_a : a \in \Sigma\}$, und ersetze P durch P' :

- In allen P-Produktionen ersetze $a \in \Sigma$ in rechter Seite durch Z_a,
- Füge $Z_a \to a$ hinzu.

Resultat: Äquivalente Grammatik mit Produktion nur von der Form $X \to v$, wobei $\varepsilon \neq v \in (V')^*$ oder $v \in \Sigma$.

Schritt 2 (Elimination von Produktionen $X \to Y$): Sei G gemäß Schritt 1 erhalten und sei (X, Y) mit $X \to_G^* Y$. Betrachte alle Produktionen $Y \to v_i$ $(i = 1, \ldots, k)$ in P für die v_i keine Variable ist. Füge dann $X \to v_i$ $(i = 1, \ldots, n)$ hinzu. Führe dies für alle derartigen Paare (X, Y) aus. Nun streiche alle Produktionen der Form $X \to Y$. Die erzeugt Sprache hat sich nicht geändert.

Schritt 3 (Elimination von Produktionen $X \to Y_1 \dots Y_k$ für $k \geq 3$): Produktionen $X \to Y_1 \dots Y_k$ mit $k \geq 3$ können äquivalent ersetzt werden durch

$$X \to Y_1 \dots Y_{k-2} Z,$$
$$Z \to Y_{k-1} Y_k,$$

wobei Z eine neue Variable ist.
Iteriere bis nur noch Produktionen der Form $X \to Y_1 Y_2$ übrig sind.

Die nach Ausführung der Schritte 1-3 aus G erhaltene Grammatik ist zu G äquivalent und in Chomsky-Normalform. □

Satz: Für eine CNF Grammatik G ist die Länge einer Ableitung von $w \in L(G)$ stets $2|w| - 1$.

Beweis: Betrachte $w \in (\Sigma \cup V)^*$ mit $X_0 \to_G^* w$, wobei l die Länge der Ableitung sei. Man zeigt durch Induktion über l, dass

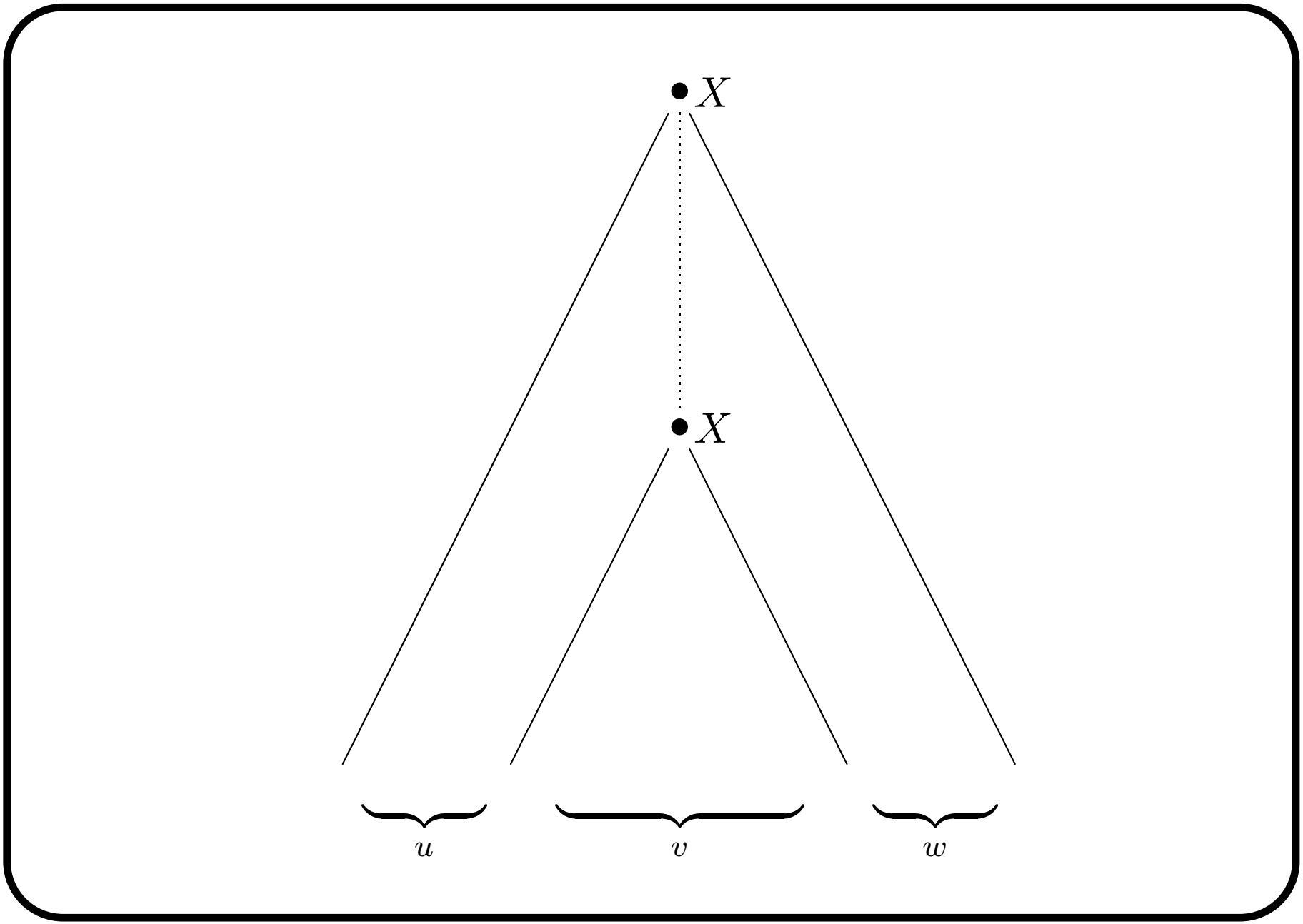
$$|w|_V + 2|w|_\Sigma = l + 1.$$

Falls w also ein ableitbares Wort in Σ^* ist, so hat jede Ableitung die Länge $l = 2|w| - 1$. □

Pumping Lemma für kontextfreie Sprachen

Theorem: Für jede kontextfreie Sprache $L \subseteq \Sigma^*$ existiert ein $n \in \mathbb{N}$, so dass sich jedes $x \in L$ mit $|x| \geq n$ zerlegen lässt in $x = yuvwz$, wobei $uw \neq \varepsilon$ und $yu^m vw^m z \in L$ für alle $m \in \mathbb{N}$. Man kann dabei u, v, w so wählen, dass $|uvw| \leq n$.

Beweiskizze: Nehme an, dass G in CNF ist mit $L(G) = L \setminus \{\varepsilon\}$. Setze $n := 2^{|V|+2}$, wobei $|V|$ die Anzahl der Variablen von G ist. Da der Ableitungsbaum eines Wortes x in G ein binärer Baum T_x ist, gilt falls $|x| \geq n$, dass T_x einen Pfad der Länge $|V| + 2$ hat, in dem $|V| + 1$ Variablen vorkommen, also eine Variable X mindestens zweimal. Konstruiere nun uvw wie in der Skizze:


X
X
u
v
w

Korollar: $L := \{a^n b^n c^n : n \in \mathbb{N}\}$ ist nicht kontextfrei (aber kontextsensitiv). Insbesondere also

$$\text{Typ 2} \subset \text{Typ 1}.$$

Abschlusseigenschaften kontextfreier Sprachen

Theorem:

1) Die Klasse der kontextfreien Sprachen ist unter den Operationen Vereinigung, Konkatenation und Stern abgeschlossen, d.h. mit L_1, L_2, L sind auch $L_1 \cup L_2, L_1 \cdot L_2$ und L^* kontextfrei.

2) Die Klasse der kontextfreien Sprachen ist weder unter Durchschnitt noch unter Komplement abgeschlossen.

Beweis: 1)(i) **Vereinigung** und **Konkatenation**: zu Typ 2 Grammatiken $G^{(i)} = (\Sigma, V^{(i)}, P^{(i)}, X_0^{(i)})$ mit $L(G^{(i)}) = L_i$ konstruieren wir Typ 2 Grammatiken G für $L_1 \cup L_2$ und $L_1 \cdot L_2$:
O.B.d.A. sei $V^{(1)} \cap V^{(2)} = \emptyset$.

Für G wählen wir eine neue Startvariable X_0.

Definiere $G = (\Sigma, V, P, X_0)$ mit $L(G) = L(G^{(1)}) \cup L(G^{(2)})$:

$$V := V^{(1)} \cup V^{(2)} \cup \{X_0\},$$

$$P := \{X_0 \to X_0^{(1)}, X_0 \to X_0^{(2)}\} \cup P^{(1)} \cup P^{(2)}.$$

Definiere: $G = (\Sigma, V, P, X_0)$ mit $L(G) = L(G^{(1)}) \cdot L(G^{(2)})$:

$$\begin{aligned} V &:= V^{(1)} \cup V^{(2)} \cup \{X_0\}, \\ P &:= \{X_0 \to X_0^{(1)} X_0^{(2)}\} \cup P^{(1)} \cup P^{(2)}. \end{aligned}$$

(ii) **Sternoperation**: zu Typ-2 Grammatik $G = (\Sigma, V, P, X_0)$ für L definiere Typ-2 Grammatik $G' = (\Sigma, V', P', X_0')$ für L^* durch

$$\begin{aligned} V' &:= V \cup \{X_0'\}, \\ P' &:= \{X_0' \to \varepsilon, X_0' \to X_0' X_0\} \cup P. \end{aligned}$$

2)(i) **Kein Abschluss unter Durchschnitt:**

Nach 1)(ii) sind

$$L_1 := \{a^n b^n : n \in \mathbb{N}\} \cdot \{c\}^*, \quad L_2 := \{a\}^* \cdot \{b^n c^n : n \in \mathbb{N}\}$$

kontextfrei, nicht aber $L_1 \cap L_2 = \{a^n b^n c^n : n \in \mathbb{N}\}$!

(ii) **Kein Abschluss unter Komplement:**

Wegen

$$L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$$

würde aus der Abgeschlossenheit unter Komplement zusammen mit der in 1) bewiesenen Abgeschlossenheit unter Vereinigung die Angeschlossenheit unter Durchschnitt folgen, was (i) widerspricht.

□

Entscheidbarkeit des Wortproblems für kontextfreie Sprachen

Satz: Sei $G = (\Sigma, V, P, X_0)$ eine Typ-2 Grammatik in Chomsky NF (d.h. nur Produktionen $X \to a$ und $X \to YZ$). Für $w = a_1 \dots a_n \in \Sigma^n$ und $1 \leq i \leq j \leq n$ sei $w_{i,j} = a_i \dots a_j$. Dann gilt für alle $1 \leq i \leq j \leq n$ und $X \in V$:

$(*)$

$$X \to_G^* w_{i,j}$$

gdw.

$$i = j \text{ und } (X \to a_i) \in P \text{ oder}$$

für ein k mit $i \leq k < j$ und ein $X \to YZ$ ist

$$Y \to_G^* w_{i,k} \text{ und } Z \to_G^* w_{k+1,j}.$$

Korollar (Cocke, Younger, Kasami): Der obige Satz liefert ein Verfahren (CYK-Algorithmus), das für eine durch eine CNF-Grammatik G erzeugte kontextfreie Sprache $L(G)$ das Wortproblem effizient entscheidet.

Beweis: Sei Σ das Alphabet von L und $w \in \Sigma^*$. Mittels $(*)$ kann man sukzessive für immer längere Teilwörter $w_{i,j}$ von w die Menge $V_{i,j}$ derjenigen Variablen X von G bestimmen mit $X \to_G^* w_{i,j}$. Für $i = 1$ und $j = |w|$ liefert der Test '$X_0 \in V_{1,|w|}$?' die Anwort auf die Frage, ob $X_0 \to_G^* w$, d.h. ob $w \in L(G)$. $\square$

Kellerautomaten (PDA): Ein charakterisierendes Berechnungsmodell für Typ 2

PDA = NFA + Kellerspeicher (stack, push-down storage)

Zwei Alphabete: Σ (Eingabe) und Γ (Keller).

Konfiguration jeweils bestimmt durch

- Zustand ($q \in Q$)
- Position in der Eingabe
- Kellerinhalt ($\alpha \in \Gamma^*$).

Erlaubte Übergänge abhängig von (**nichtdeterministisch**):

- Zustand
- oberstem Kellersymbol (pop)
- nächstem Eingabesymbol

Übergang resultiert in

- Zustandswechsel
- (optional) Vorrücken in Eingabe
- pop und push im Keller:
  - Entfernen des obersten Kellersymbols (pop)
  - Einschreiben eines Wortes in Keller (push)

PDA $\mathcal{P} = (\Sigma, Q, q_0, \Delta, A, \Gamma, \#)$:

Σ	Eingabealphabet
Γ	Kelleralphabet
$\# \in \Gamma$	Anfangs-Kellersymbol
Q	Zustandsmenge
$q_0 \in Q$	Anfangszustand
$A \subseteq Q$	akzeptierende Zustände
$\Delta \subseteq Q \times \Gamma \times (\Sigma \cup \{\varepsilon\}) \times \Gamma^* \times Q$	Übergangsrelation (endlich!)

Konfigurationen

$C = (q, v, \alpha) \in Q \times \Sigma^* \times \Gamma^*$:

$q \in Q$ aktueller Zustand,

$v \in \Sigma^*$ Restabschnitt des Eingabewortes,

$\alpha \in \Gamma^*$ aktueller Kellerinhalt.

Startkonfiguration auf Eingabe w: $C_0[w] = (q_0, w, \#)$

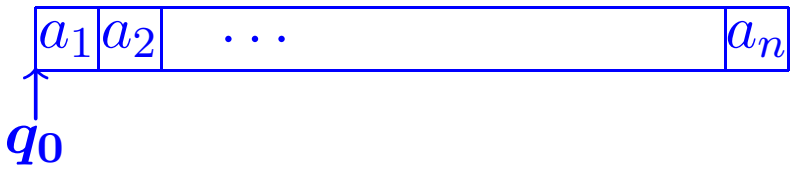
Nachfolgekonfigurationen zu $C = (q, v, \alpha)$, $\alpha = \gamma\,\alpha_{\text{rest}}$,
$\gamma \in \Gamma$ oberstes Kellersymbol, Keller nicht leer:

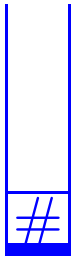
$$C' = (q', v', \alpha') \text{ mit } \left.\begin{array}{l} v = xv' \\ \alpha' = \beta\,\alpha_{\text{rest}} \end{array}\right\} \text{ für ein } (q, \gamma, x, \beta, q') \in \Delta.$$

PDA Berechnungen I

Startkonfiguration auf $w = a_1 \dots a_n$

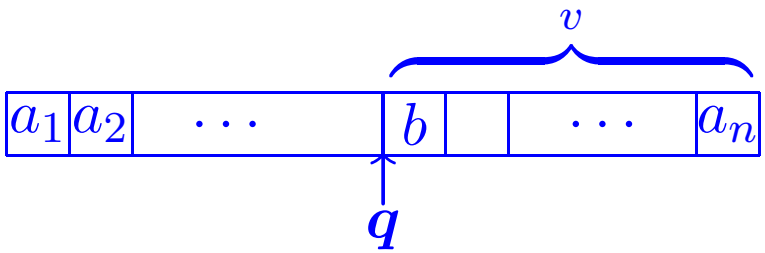
$C_0[w]$

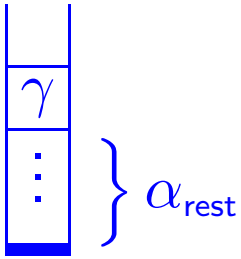




Typische Konfiguration C auf $w = a_1 \dots a_n$

C



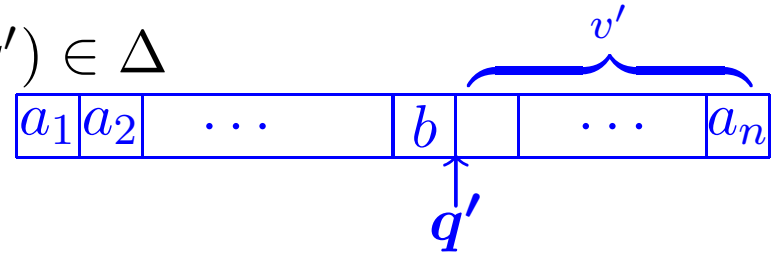


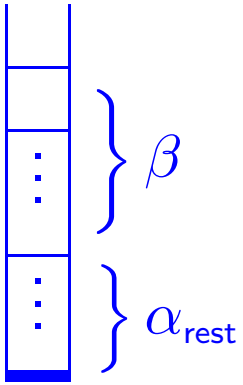
PDA Berechnungen II

Nachfolgekonfiguration von C

zu $(q, \gamma, b, \beta, q') \in \Delta$

C'

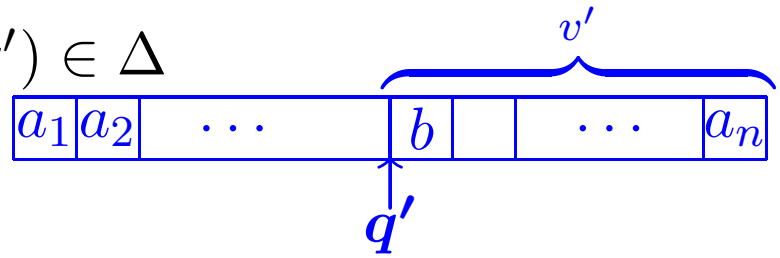


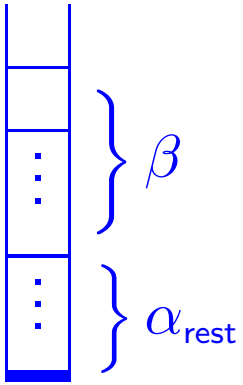


Nachfolgekonfiguration von C

zu $(q, \gamma, \varepsilon, \beta, q') \in \Delta$

C'





PDA Berechnungen III

(Terminierende) Berechnung von $\mathcal{P}$ auf Eingabe $w \in \Sigma^*$:

Konfigurationsfolge $C_0 \dots C_f$, wobei

$C_0 = C_0[w] = (q_0, w, \#)$,

C_{i+1} eine Nachfolgekonfiguration von C_i, $0 \leq i < f$,

C_f **Endkonfiguration** ohne anwendbare Transition

$C_0[w] \xrightarrow{\mathcal{P}} C_f$

Akzeptierende Berechnung: $C_f = (q, \varepsilon, \varepsilon)$ mit $q \in A$.

Von $\mathcal{P}$ akzeptierte Sprache:

$$L(\mathcal{P}) = \left\{ w \in \Sigma^* : C_0[w] \xrightarrow{\mathcal{P}} (q, \varepsilon, \varepsilon) \text{ für ein } q \in A \right\}.$$

Beispiel: PDA für Klammersprache

$$\mathcal{P} = \big(\{(,)\}, Q, q_0, \Delta, \{q_0\}, \Gamma, \#\big),$$

$$Q = A = \{q\}, \text{ ein Zustand } q = q_0, \ \Gamma = \{|, \#\}.$$

Transitionen:

$(q\,,\#\,,(\,,\|\#\,,q)$	verarbeitet "(" und addiert "|" im Keller
$(q\,,\,\|\,,(\,,\,\|\|\,,q)$	verarbeitet "(" und addiert "|" im Keller
$(q\,,\,\|\,,)\,,\,\varepsilon\,,q)$	verarbeitet ")" und löscht ein "|" im Keller
$(q\,,\#\,,\varepsilon\,,\,\varepsilon\,,q)$	ε-Transition, die # löscht

Idee: Kellerspeicher als Zähler für $|u|_(- |u|_)$!

Kontextfrei (Typ 2) = PDA-erkennbar

Theorem: Für $L \subseteq \Sigma^*$ sind äquivalent:

1) L kontextfrei.

2) $L = L(\mathcal{P})$ für einen PDA $\mathcal{P}$.

Beweisidee: '1) $\Rightarrow$ 2)': Sei $L(G)$ durch eine kontextfreie Grammatik $G = (\Sigma, V, P, X_0)$ erzeugt.
Wir konstruieren einen PDA $\mathcal{P} = (\Sigma, Q, q_0, \Delta, A, \Gamma, \#)$ der $L(G)$ erkennt:

$\mathcal{P}$ hat nur einen Zustand q, der sowohl Start- wie auch akzeptierender Zustand ist: $Q := A := \{q\}, q_0 := q$.

Ferner: $\Gamma := V \cup \Sigma, \# := X_0$.

Transitionsrelation Δ :

$$\{(q, X, \varepsilon, \alpha, q) \ : \ X \to \alpha \in P\} \cup \{(q, a, a, \varepsilon, q) \ : \ a \in \Sigma\}.$$

Man zeigt durch Induktion über die Ableitung, dass $\mathcal{P}$ ausgehend von $C_0[w] = (q_0, w, \#)$ gerade die Konfigurationen $C = (q, v, \alpha)$ annehmen kann, für die $w = uv$ und $X_0 \to_G^* u\alpha$. Die (einzige) akzeptierende Endkonfiguration $C = (q, \varepsilon, \varepsilon)$ wird also genau dann erreicht, wenn $X_0 \to_G^* w$, d.h. wenn $w \in L(G)$.

'2) $\Rightarrow$ 1)': Sei $L = L(\mathcal{P})$ eine durch einen PDA $\mathcal{P} = (\Sigma, Q, q_0, \Delta, A, \Gamma, \#)$ akzeptierte Sprache.

Wir konstruieren eine kontextfreie Grammatik $G = (\Sigma, V, P, X_0)$ wie folgt: $V := (Q \times \Gamma \times Q) \cup \{X_0\}$.

Für alle $q, q', q_i \in Q, \gamma, \gamma_i \in \Gamma, x \in \Sigma \cup \{\varepsilon\}$ haben wir die folgenden Produktionen P :

$X_0 \to (q_0, \#, q)$ für $q \in A$,

$(q, \gamma, q') \to x$ für $(q, \gamma, x, \varepsilon, q') \in \Delta$,

$(q, \gamma, q') \to x(q_1, \gamma_1, q')$ für $(q, \gamma, x, \gamma_1, q_1) \in \Delta$,

$(q, \gamma, q') \to x(q_1, \gamma_1, q_2)(q_2, \gamma_2, q_3) \dots (q_l, \gamma_l, q')$
für $(q, \gamma, x, \gamma_1 \dots \gamma_l, q_1) \in \Delta$.

Durch Induktion zeigt man (Übung), dass für alle $w \in \Sigma^*$:

$$(q, \gamma, q') \to_G^* w \text{ gdw. } (q, w, \gamma) \xrightarrow{\mathcal{P}} (q', \varepsilon, \varepsilon).$$

Insbesondere also

$$X_0 \to_G^* w \;\Leftrightarrow\; (\exists q' \in A)\,[\,(q_0, \#, q') \to_G^* w\,]$$

$$\Leftrightarrow (\exists q' \in A)\,[\,\underbrace{(q_0, w, \#)}_{C_0=} \xrightarrow{\mathcal{P}} (q', \varepsilon, \varepsilon)\,].$$

Also $L(G) = L(\mathcal{P})$. □

Ergänzungen

Man kann auch eine deterministische Variante von PDA'a – mit DPDA bezeichnet – definieren. Die von DPDA akzeptierten Sprachen heißen **deterministisch kontextfrei** und bilden eine echte Teiklasse der kontextfreien Sprachen.

Satz: Die Klasse der deterministisch kontextfreien Sprachen ist unter Komplement abgeschlossen, aber weder unter Schnitt noch unter Vereinigung abgeschlossen.

Turingmaschinen, DTM: ein universelles Berechnungsmodell

Hier: deterministische Variante DTM

DTM = DFA + unbeschränkter Lese/Schreibzugriff.

Eingabe-/Arbeitsspeicher: unbeschränkte Folge von Zellen

als "Band" mit Lese/Schreibkopf

Konfiguration bestimmt durch

- Zustand ($q \in Q$)
- Position auf dem Band
- (relevante) Bandbeschriftung

Übergang in **Nachfolgekonfiguration** abhängig von

- Zustand
- aktuell gelesenem Bandsymbol

Ein Übergang resultiert in

- Zustandswechsel
- Schreiben
- Kopfbewegung $(<, \circ, >)$

DTM $\mathcal{M} = (\Sigma, Q, q_0, \delta, q^+, q^-)$

Q	Zustandsmenge
$q_0 \in Q$	Anfangszustand
$q^+ \in Q$	akzeptierender Endzustand
$q^- \in Q$	verwerfender Endzustand, $q^- \neq q^+$
δ	Übergangsfunktion

$$\delta : Q \times (\Sigma \cup \{\Box\}) \to (\Sigma \cup \{\Box\}) \times \{<, \circ, >\} \times Q$$

Konfigurationen:

$$C = (\alpha, q, x, \beta) \in (\Sigma \cup \{\Box\})^* \times Q \times (\Sigma \cup \{\Box\}) \times (\Sigma \cup \{\Box\})^*$$

α: (relevanter) Bandinhalt links vom Kopf (links von α nur $\Box$)

x: Bandinhalt in Kopfposition

β: (relevanter) Bandinhalt rechts vom Kopf (rechts von β nur $\Box$)

q: aktueller Zustand

Startkonfiguration auf Eingabe w: $C_0[w] := (\varepsilon, q_0, \Box, w)$

Nachfolgekonfiguration: $C \longmapsto C'$ gemäß δ :

Für $C = (\alpha, q, x, \beta)$ mit $q \neq q^+, q^-$ ist C' definiert durch

$$C' := \begin{cases} (\alpha, q', x', \beta), & \text{falls } d = \circ, \\ (\alpha_0, q', a, x'\beta), & \text{falls } d = <, \\ (\alpha x', q', b, \beta_0), & \text{falls } d = >, \end{cases}$$

wobei $\delta(q, x) = (x', d, q')$ und $\alpha = \alpha_0 a, \beta = b\beta_0$ mit $a, b \in \Sigma \cup \{\Box\}$.

Endkonfigurationen: $q \in \{q^+, q^-\}$, akzeptierend/verwerfend.

Falls $\mathcal{M}$ ausgehend von $C_0[w]$ über endliche viele Nachfolgekonfigurationen, die keine Endkonfigurationen sind, in eine Konfiguration mit Zustand q kommt, so schreiben wir

$$w \xrightarrow{\mathcal{M}} q.$$

Definition:

1) Wir sagen, dass (die Berechnung von) $\mathcal{M}$ angesetzt auf das Wort w **stoppt**, falls $w \xrightarrow{\mathcal{M}} q$ mit $q = q^+$ oder $q = q^-$. Wir schreiben dann $w \xrightarrow{\mathcal{M}}$ STOP.
   Falls dies nicht gilt, so sagen wir, dass $\mathcal{M}$ angesetzt auf w divergiert: $w \xrightarrow{\mathcal{M}} \infty$.
2) Eine DTM $\mathcal{M}$ über dem Alphabet Σ heißt **total**, falls $\mathcal{M}$ angesetzt auf jedes Wort $w \in \Sigma^*$ stoppt.

Definition:

1) Die von DTM $\mathcal{M}$ **akzeptierte Sprache** ist definiert durch
$L(\mathcal{M}) = \{w \in \Sigma^* : \mathcal{M} \text{ akzeptiert } w \} = \{w \in \Sigma^* : w \xrightarrow{\mathcal{M}} q^+\}$.

2) Eine Sprache L heißt **rekursiv aufzählbar (semi-entscheidbar)**, falls es eine DTM $\mathcal{M}$ gibt, die L akzeptiert, d.h. $L = L(\mathcal{M})$.

3) $\mathcal{M}$ **entscheidet** (das Wortproblem für) eine Sprache L falls für alle $w \in \Sigma^*$:

$$w \xrightarrow{\mathcal{M}} \begin{cases} q^+ & \text{für } w \in L \\ q^- & \text{für } w \notin L \end{cases}$$

4) Eine Sprache L heißt **entscheidbar (rekursiv)**, falls es eine DTM $\mathcal{M}$ gibt, die L entscheidet.

Einfache Eigenschaften

1) Falls $\mathcal{M}$ die Sprache L entscheidet, so akzeptiert $\mathcal{M}$ die Sprache auch.
2) Für eine **totale** DTM $\mathcal{M}$ gilt, dass $\mathcal{M}$ eine Sprache L genau dann akzeptiert, wenn $\mathcal{M}$ die Sprache L entscheidet.

Satz (E. Post): Für $L \subseteq \Sigma^*$ sind äquivalent:

1) L ist entscheidbar.
2) Sowohl L als auch $\overline{L}$ sind aufzählbar.

Berechenbare Funktionen

Sei $\mathcal{M}$ eine DTM. Falls $\mathcal{M}$ angesetzt auf ein Input-Wort w stoppt in einer Endkonfiguration (α, q, x, β), so bezeichnet $M(w)$ das längste Präfix von β ohne $\Box$.

Definition: Eine partielle Funktion $f : \Sigma^* \to \Sigma^*$ heißt **berechenbar**, falls es eine DTM $\mathcal{M}$ gibt mit

$$(\forall w \in \Sigma^*)\ (f(w) = M(w)).$$

Beispiel: DTM für Palindrom

δ	$\square$	0	1
q_0	$(\square, >, q^?)$		
$q^?$	$(\square, \circ, q^+)$	$(\square, >, q^{\rightarrow 0})$	$(\square, >, q^{\rightarrow 1})$
$q^{\rightarrow 0}$	$(\square, <, q^{\leftarrow 0})$	$(0, >, q^{\rightarrow 0})$	$(1, >, q^{\rightarrow 0})$
$q^{\rightarrow 1}$	$(\square, <, q^{\leftarrow 1})$	$(0, >, q^{\rightarrow 1})$	$(1, >, q^{\rightarrow 1})$
$q^{\leftarrow 0}$	$(\square, \circ, q^+)$	$(\square, <, q^{\leftarrow})$	$(\square, \circ, q^-)$
$q^{\leftarrow 1}$	$(\square, \circ, q^+)$	$(\square, \circ, q^-)$	$(\square, <, q^{\leftarrow})$
$q^{\leftarrow}$	$(\square, >, q^?)$	$(0, <, q^{\leftarrow})$	$(1, <, q^{\leftarrow})$

Intendierte Rolle der Zustände:

q_0	:	Startzustand	$q^?$	:	Anfang abfragen
$q^{\rightarrow 0}$	:	zum Ende, merke 0	$q^{\leftarrow 0}$	:	vergleiche Ende mit 0
$q^{\rightarrow 1}$	:	zum Ende, merke 1	$q^{\leftarrow 1}$	:	vergleiche Ende mit 1
$q^{\leftarrow}$	:	zum Anfang	q^+/q^-	:	akzeptiere/verwerfe

Church Turing These

algorithmische Entscheidbarkeit = Turing-Entscheidbarkeit
algorithmische Erzeugbarkeit = Turing-Aufzählbarkeit
Berechenbarkeit = Turing-Berechenbarkeit

Belege:

- Erfahrung: alle akzeptierten Algorithmen lassen sich im Prinzip mit DTM simulieren
- Robustheit des TM Modells
- bewiesene Äquivalenz mit ganz unterschiedlichen alternativen Charakterisierungen

Eine unentscheidbare semi-entscheidbare Sprache

DTM's $\mathcal{M}$ über einem Alphabet Σ lassen sich durch Wörter in Σ^* kodieren. Meistens verwendet man hier z.B. $\Sigma := \{0, 1\}$ oder $\Sigma := \{|\}$ und interpretiert Wörter über Σ als natürliche Zahlen n in binärer Darstellung oder repräsentiert durch $||\ldots|$ (n-mal). Mittels üblicher Kodierungen endlicher Folgen durch natürliche Zahlen lässt sich $(\Sigma, Q, q_0, \delta, q^+, q^-)$ vollständig kodieren.

Die Details von $\mathcal{M} \mapsto \langle\mathcal{M}\rangle$ sind unerheblich. Wir verlangen nur, dass es eine DTM $\mathcal{U}$ gibt mit $\mathcal{U}$ angewendet auf ein Wort

$$\langle\mathcal{M}\rangle \,\square\, w$$

führt $\mathcal{M}$ angewendet auf $w \in \Sigma^*$ aus.

Universelle DTM

Für $\mathcal{U}$ gilt

$$L(\mathcal{U}) = \{\langle \mathcal{M} \rangle \Box w : w \in L(\mathcal{M})\}.$$

$\mathcal{U}$ kann also jede DTM $\mathcal{M}$ simulieren und heißt daher **universelle Turingmachine**.

Das (spezielle) **Halteproblem (HP)** für Turingmaschinen ist definiert als die Sprache

$$\text{HP} := \{\langle \mathcal{M} \rangle : \mathcal{M} \text{ angesetzt auf } \langle \mathcal{M} \rangle \text{ stoppt}\}.$$

Zu $\mathcal{M}$ konstruiert man leicht eine DTM $\mathcal{M}'$ mit

$$L(\mathcal{M}') = \{w \in \Sigma^* : \mathcal{M} \text{ angesetzt auf } w \text{ stoppt}\}.$$

HP wird erkannt durch DTM $\mathcal{M}_{\mathrm{HP}}$, die $\langle \mathcal{M} \rangle$ genau dann akzeptiert, wenn $\langle \mathcal{M}' \rangle \square \langle \mathcal{M} \rangle$ von $\mathcal{U}$ akzeptiert wird.

Korollar: HP ist rekursiv aufzählbar (semi-entscheidbar).

Satz: HP ist unentscheidbar.

Beweis: Angenommen es gäbe eine DTM $\mathcal{M}_0$, die HP entscheidet. Dann gilt für jede Eingabe $w = \langle \mathcal{M} \rangle$:

$$\langle \mathcal{M} \rangle \overset{\mathcal{M}_0}{\to} q^+ \Leftrightarrow \langle \mathcal{M} \rangle \overset{\mathcal{M}}{\to} \text{STOPP}$$
$$\langle \mathcal{M} \rangle \overset{\mathcal{M}_0}{\to} q^- \Leftrightarrow \langle \mathcal{M} \rangle \overset{\mathcal{M}}{\to} \infty \; (\equiv \neg\text{-STOPP}).$$

Wir schreiben $\mathcal{M}_0$ zu einer neuen DTM $\mathcal{M}_1$ um mit

$$w \overset{\mathcal{M}_1}{\to} \infty \Leftrightarrow w \overset{\mathcal{M}_0}{\to} q^+$$
$$w \overset{\mathcal{M}_1}{\to} q^- \Leftrightarrow w \overset{\mathcal{M}_0}{\to} q^-.$$

Zusammengenommen ergibt sich

$$\langle \mathcal{M}_1 \rangle \overset{\mathcal{M}_1}{\to} \infty \Leftrightarrow \langle \mathcal{M}_1 \rangle \overset{\mathcal{M}_1}{\to} \text{ STOPP},$$

was einen Widerspruch darstellt. □

Korollar: $\overline{\text{HP}}$ ist nicht rekursiv aufzählbar. Insbesondere ist also die Klasse der rekursiv aufzählbaren Sprachen nicht unter Komplement abgeschlossen.

Das Postsche Korrespondenzproblem

Die Unentscheidbarkeit vieler anderer Probleme P läßt sich auf die Unentscheidbarkeit des Halteproblems HP reduzieren (indem man zeigt, dass sich eine P-entscheidende DTM $\mathcal{M}$ in eine HP-entscheidende DTM $\mathcal{M}'$ umschreiben ließe). Beispiel:

Gegeben: endliche Folge von Wortpaaren
$(x_1, y_1), (x_2, y_2), \ldots, (x_k, y_k)$ mit $x_i, y_i \in \Sigma^+$.
Gefragt: Gibt es eine Folge von Indizes
$i_1, i_2, \ldots, i_n \in \{1, 2, \ldots, k\}$ $(n \geq 1)$ mit

$$x_{i_1} x_{i_2} \ldots x_{i_n} = y_{i_1} y_{i_2} \ldots y_{i_n}?$$

$(i_1, i_2, \ldots, i_n)$ heißt dann eine Lösung des Korrespondenzproblems $(x_1, y_1), (x_2, y_2), \ldots, (x_k, y_k)$.

Für ein gegebenes Alphabet Σ besteht das Postsche Korrespondenzproblem PCP_Σ gerade in der Menge endlicher Folgen $(x_1, y_1), (x_2, y_2), \ldots, (x_k, y_k)$, deren Korrespondenzproblem lösbar ist.

Satz: PCP_Σ ist stets rekursiv aufzählbar, aber bereits für $\Sigma := \{0, 1\}$ nicht rekursiv entscheidbar.

Charakterisierung der Typ-0 Sprachen

Satz: Eine Sprache L ist genau dann vom Typ 0 (d.h. von einer Typ-0 Grammatik erzeugt) wenn L rekursiv aufzählbar ist (d.h. von einer DTM $\mathcal{M}$ akzeptiert wird).

Korollar:

1) Es gibt von Typ-0 Grammatiken erzeugte Sprachen, deren Wortproblem unentscheidbar ist.
2) HP ist Typ 0, während $\overline{\mathrm{HP}}$ nicht vom Typ 0 ist. Insbesondere ist die Klasse der Typ-0 Sprachen also nicht unter Komplement abgeschlossen.

Nichtdeterministische Turingmaschinen NTM

Eine **nichtdeterministische Turingmaschine NTM** hat statt einer Übergangsfunktion δ eine Übergangsrelation Δ, d.h. jedem Paar $(q, x) \in Q \times (\Sigma \cup \{\Box\})$ wird eine endliche Menge von Tripeln in $(\Sigma \cup \{\Box\}) \times \{<, \circ, >\} \times Q$ zugeordnet.

Satz: Zu jeder NTM $\mathcal{N}$ gibt es eine DTM $\mathcal{M}$, die dieselbe Sprache akzeptiert, d.h. $L(\mathcal{N}) = L(\mathcal{M})$.

Beweisidee: Probiere alle nichtdeterministischen Möglichkeiten durch (Details siehe: Hopcroft/Motwani/Ullman, S. 350-520). $\Box$

Problem:

Im allgemeinen braucht $\mathcal{M}$ exponentiell mehr Zeit als $\mathcal{N}$.

Charakterisierung der Typ 1 Sprachen

Eine nicht-deterministische Turingmaschine (NTM) $\mathcal{M}$ heißt **linear platzbeschränkt**, falls es eine Konstante $c \in \mathbb{N}$ gibt, so dass sich $\mathcal{M}$ angesetzt auf ein Wort $w \in \Sigma^*$ mit dem Lese/Schreibkopf höchstens $c \cdot |w|$ Felder vom Startfeld entfernt.

Satz (Landweber-Kuroda): Eine Sprache $L \subseteq \Sigma^*$ ist genau dann kontextsensitiv (d.h. von einer Typ-1 Grammatik erzeugt), wenn es eine linear platzbeschränkte NTM $\mathcal{M}$ gibt mit $L = L(\mathcal{M})$.

Beweis: Siehe M.A. Davis, R. Sigal, E.J. Weyuker: Computability, Complexity, and Languages, S. 330-337.

Satz: Das Wortproblem für kontextsensitive Sprachen ist entscheidbar.

Beweisidee: Da Ableitungen in kontextsensitiven Grammatiken nicht längenverkürzend sind, kann man den Suchraum beschränken: es reicht bei Input w alle Ableitungsfolgen, deren Zwischenschritte v alle $|v| \leq |w|$ erfüllen, zu betrachten. Nach der Elimination von Zyklen sind das nur endlich viele. □

Korollar:

HP ist keine kontextsentsitive Sprache (d.h. nicht Typ 1).

Also erhalten wir die **echte Inklusion** Typ 1 $\subset$ Typ 0.

Bemerkung:

Nicht jede entscheidbare Sprache ist kontextsensitiv.

Also:

$$\text{Typ 1} \subset \text{entscheidbare Sprachen} \subset \text{Typ 0}.$$

Abschlusseigenschaften für Typ 1 und Typ 0

Man zeigt ähnlich zu schon bewiesenen Abschlusseigenschaften, dass die Typ 1 Sprachen unter $\cup, \cap, \cdot, *$ abgeschlossen sind. Ein nicht-triviales Ergebnis ist der folgende

Satz (Immerman, Szelepcsenyi 1987): Die Klasse der kontextsensitiven Sprachen (also vom Typ 1) ist unter Komplement abgeschlossen.

Zusammenfassung der Abschlusseigenschaften in der Chomsky-Hierarchie

Typ	abgeschlossen unter $\cup$	$\cap$	$\overline{}$	$\cdot$	$*$
3	+	+	+	+	+
2	+	−	−	+	+
1	+	+	+	+	+
0	+	+	−	+	+
bel. Σ-Sprachen	+	+	+	+	+

Zusammenfassung

Äquivalenzrelationen

Bei 2-stelligen (oder ,,binären") Relationen R schreiben wir meistens ,,xRy" statt ,,$(x, y) \in R$", also $x \leq y, x$ 4 x etc. (,,infixe" Notation).

Mögliche Eigenschaften binärer Relationen über A: z.B.

- **Reflexivität:** für alle $a \in A$ gilt: aRa.
- **Symmetrie:** für alle $a, b \in A$ gilt: aRb gdw. bRa.
- **Transitivität:** für alle $a, b, c \in A$ gilt: wenn aRb und bRc, dann auch aRc.

Definition: Eine reflexive, symmetrische und transitive Relation $R \subseteq A^2$ heißt **Äquivalenzrelation**.
Wir schreiben oft $\sim$ statt R.

Beispiele:

- Die Gleichheit '$=$' von Mengen ist eine Äquivalenzrelation.
- Auf $\mathbb{N}^2$ wird durch
  $(n_1, m_1) \sim (n_2, m_2) :\Leftrightarrow n_1 - m_1 = n_2 - m_2$ eine Äquivalenzrelation definiert. Reflexivität und Symmetrie sind trivial. Transitivität:

  $$n_1 - m_1 = n_2 - m_2 \text{ und } n_2 - m_2 = n_3 - m_3 \Rightarrow$$

  $$n_1 - m_1 = n_3 - m_3.$$

- Auf Σ^* ist $w_1 \sim w_2 :\Leftrightarrow |w_1| = |w_2|$ eine Äquivalenzrelation.

Äquivalenzklassen

Definition: Sei A eine Menge und $\sim$ eine Äquivalenzrelation auf A. Für jedes $a \in A$ def. wir die **Äquivalenzklasse** von a durch

$$[a]_\sim := \{b \in A : a \sim b\} \ (\subseteq A).$$

Satz:

1) $a \in [a]_\sim$ und somit $A = \bigcup_{a \in A} [a]_\sim$.

2) $a \sim b \Leftrightarrow [a]_\sim = [b]_\sim$ ($\sim$ also verallgemeinerte Gleichheit).

3) $a \not\sim b \Leftrightarrow [a]_\sim \cap [b]_\sim = \emptyset$.

Also: durch **Quotient** $A/\sim := \{[a]_\sim : a \in A\}$ Zerlegung von A in disjunkte Teilmengen.

Falls $A/\sim$ endlich: $\mathrm{index}(\sim) := |A/\sim|$ **Index von** $\sim$.

Funktionen

Definition: Eine Relation $R \subseteq A \times B$ heißt **Funktion von A nach B**, falls es zu jedem $a \in A$ genau ein $b \in B$ gibt mit $(a, b) \in R$.

Funktionen werden üblicherweise mit f, g, h, etc. bezeichnet und man schreibt $f : A \to B$.

Das eindeutig bestimmte $b \in B$ mit $(a, b) \in f$ wird mit $f(a)$ bezeichnet und **Bild von a unter f** genannt.

A heißt **Definitionsbereich** und B **Zielbereich** von f.

$$f[A] := \{f(a) : a \in A\} \subseteq B \textbf{ Bild von } f.$$

Wir schreiben auch $f(A)$ statt $f[A]$.

Für $B' \subseteq B$ heißt

$$f^{-1}[B'] := \{a \in A : f(a) \in B'\}$$

Urbild von B'.

Beachte: Urbilder können auch leer sein.

Beispiel: Natürliche Projektion

$$\pi_\sim : A \to P(A),\ a \mapsto [a]_\sim,$$

wobei $\sim$ eine Äquivalenzrelation auf A ist.
$\pi_\sim[A] = A/\sim$.

Definition: Sei $f : A \to B$ eine Funktion. f heißt

(i) **surjektiv**, falls $f[A] = B$, d.h. zu jedem $b \in B$ gibt es **mindestens ein** $a \in A$ mit $f(a) = b$, d.h. $f^{-1}[\{b\}] \neq \emptyset$.

(ii) **injektiv**, falls stets

$$a_1 \neq a_2 \;\Rightarrow\; f(a_1) \neq f(a_2),$$

d.h. zu jedem $b \in B$ gibt es **höchstens ein** $a \in A$ mit $f(a) = b$, d.h. $f^{-1}[\{b\}] = \emptyset$ oder $= \{a\}$ (Einermenge).

(iii) **bijektiv**, falls f surjektiv und injektiv ist, d.h. zu jedem $b \in B$ gibt es **genau ein** $a \in A$ mit $f(a) = b$, d.h. $f^{-1}[\{b\}]$ ist für jedes b eine Einermenge.

Komposition von Funktionen: Seien $f : A \to B$ und $g : B \to C$ Funktionen. Dann wird durch Komposition von f und g eine neue Funktion erklärt

$$g \circ f : A \to C, \quad a \longmapsto g(f(a)).$$

Definition: Sei $f : A \to B$ eine Funktion. Eine Funktion $g : B \to A$ heißt **Umkehrfunktion** von f, falls

$$g \circ f = \mathrm{id}_A \text{ und } f \circ g = \mathrm{id}_B,$$

mit den Identitätsfunktionen $\mathrm{id}_A(a) := a$, $\mathrm{id}_B(b) := b$ auf A, B.

Satz: Eine Funktion $f : A \to B$ ist genau dann eine Bijektion, wenn f eine Umkehrfunktion $f^{-1} : B \to A$ besitzt:

$$f^{-1}(b) := \text{ das eindeutig bestimmte } a \in A \text{ mit } f(a) = b.$$

Beweise durch vollständige Induktion

Satz: Um $(\forall n \in \mathbb{N})A(n)$ zu beweisen, reicht es zu zeigen:

(i) $A(0)$ ist wahr (**Induktionsanfang**) **und**

(ii) Für jedes $n \in \mathbb{N}$ gilt $A(n) \to A(n+1)$ (**Induktionsschritt**).

Variationen:

Um $(\forall n \geq n_0)\, A(n)$ zu beweisen, genügt es zu zeigen

$$A(n_0) \textbf{ and } (\forall n \geq n_0)\ (A(n) \to A(n+1))$$

oder auch nur

$$A(n_0) \textbf{ and } (\forall n \geq n_0)\ ((\forall m)(n_0 \leq m \leq n \to A(m)) \to A(n+1)).$$

Alphabete, Wörter, Sprachen

In vielen Anwendungen in der Informatik werden endliche, nicht-leere Mengen Σ als **Alphabet** bezeichnet und die Elemente $a \in \Sigma$ als **Buchstaben** oder **Zeichen**.

Wörter über einem Alphabet Σ sind endliche Folgen

$$w = a_1 \dots a_n$$

von Buchstaben $a_i \in \Sigma$.

n ist die **Länge** $|w|$ von w.
Wörter der Länge n werden mit n-Tupeln aus Σ^n identifiziert.
Wörter der Länge 1 werden mit Buchstaben identifiziert.

Menge aller Wörter über Σ:

$$\Sigma^* := \bigcup_{n \in \mathbb{N}} \Sigma^n.$$

Leeres Wort: $\varepsilon \in \Sigma^*$.

Σ-Sprache: Teilmenge $L \subseteq \Sigma^*$, d.h. Menge von Σ-Wörtern.

Menge der nicht-leeren Wörter:

$$\Sigma^+ := \Sigma^* \setminus \{\varepsilon\} = \{w \in \Sigma^* : |w| \geq 1\} = \bigcup_{n \geq 1} \Sigma^n.$$

Strukturelle Induktion

M werde, ausgehend von $M_0 \subset M$,
durch Operationen $F \in \mathcal{F}$ erzeugt. Dann lässt sich

$$(\forall x \in M) A(x)$$

beweisen anhand von

(i) **Induktionsanfang**: $A(x)$ gilt für alle $x \in M_0$.

(ii) **Induktionsschritt(e)** für $F \in \mathcal{F}$ (n-stellig):

aus $A(x_i)$ für $i = 1, \ldots, n$ folgt, dass auch $A(F(x_1, \ldots, x_n))$.

Beispiele

Bereich M	$M_0 \subset M$	erzeugende Operationen
$\mathbb{N}$	$\{0\}$	$\mathsf{S}\colon n \longmapsto n+1$
Σ^*	$\{\varepsilon\}$	$(\ w \longmapsto wa\ )$ für $a \in \Sigma$
$\{*, c\}$-Terme	$\{c\}$	$(t_1, t_2) \longmapsto (t_1 * t_2)$
endl. Teilmengen von A	$\{\emptyset\}$	$(\ B \longmapsto B \cup \{a\}\ )$ für $a \in A$

Operationen auf Sprachen

Seien L, L_1, L_2 Sprachen $\subseteq \Sigma^*$ über einem Alphabet Σ.

Boolesche Operationen:

- **Vereinigung:** $(L_1, L_2) \mapsto L_1 \cup L_2$.
- **Durchschnitt:** $(L_1, L_2) \mapsto L_1 \cap L_2$.
- **Komplement:** $L \mapsto \overline{L} := \Sigma^* \setminus L$.

Konkatenation von Sprachen:

$$(L_1, L_2) \mapsto L_1 \cdot L_2 := \{u \cdot v \,:\, u \in L_1,\, v \in L_2\}.$$

Stern-Operation:

$$L \mapsto L^* := \{u_1 \cdot \ldots \cdot u_n \,:\, u_1, \ldots, u_n \in L, n \in \mathbb{N}\}.$$

Reguläre Ausdrücke

Die Menge **REG(Σ)** der **regulären Ausdrücke** über Σ ist induktiv erzeugt durch:

(i) $\emptyset$ ist ein regulärer Ausdruck.

(ii) Für jedes $a \in \Sigma$ ist $\mathbf{a}$ ein regulärer Ausdruck.

(iii) Mit $\alpha, \beta \in \mathrm{REG}(\Sigma)$ ist auch $(\alpha + \beta) \in \mathrm{REG}(\Sigma)$.

(iv) Mit $\alpha, \beta \in \mathrm{REG}(\Sigma)$ ist auch $(\alpha\beta) \in \mathrm{REG}(\Sigma)$.

(v) Mit $\alpha \in \mathrm{REG}(\Sigma)$ ist auch $\alpha^* \in \mathrm{REG}(\Sigma)$.

Reguläre Sprachen

Jedem regulären Ausdruck (Syntax) wird induktiv eine Sprache $L(\alpha) \subseteq \Sigma^*$ zugeordnet (Semantik).

$L(\alpha)$ heißt die **durch α erzeugte reguläre Sprache**:

(i) $L(\emptyset) := \emptyset$.

(ii) $L(a) := \{a\}$.

(iii) $L(\alpha + \beta) := L(\alpha) \cup L(\beta)$.

(iv) $L(\alpha\beta) := L(\alpha) \cdot L(\beta)$.

(v) $L(\alpha^*) := \big(L(\alpha)\big)^*$.

Definition: Eine Sprache $L \subseteq \Sigma^*$ heißt **regulär**, wenn es ein $\alpha \in \mathsf{REG}(\Sigma)$ gibt mit $L = L(\alpha)$.

Beispiele regulärer Sprachen I

Beispiel 1: Sei $\Sigma := \{0, 1\}$. Dann ist die Sprache L der Wörter über Σ, mit genau einer 1 regulär, da

$$L = L(0^*10^*).$$

Das Komplement $\overline{L}$ von L ist ebenfalls regulär, da

$$\overline{L} = \underbrace{0^*}_{\text{,,keine 1"}} + \underbrace{(0+1)^*1(0+1)^*1(0+1)^*}_{\text{,,mindestens zwei 1"}}.$$

Beispiele regulärer Sprachen II

Beispiel 2: Sei $\Sigma := \{a_1, \ldots, a_n\}$. Dann sind die folgenden Sprachen regulär:

(i) $L(\emptyset^*) = \{\varepsilon\}$.

(ii) $\Sigma = L(a_1 + \ldots + a_n)$.

(iii) $\Sigma^* = L((a_1 + \ldots + a_n)^*)$.

(iv) $\Sigma^+ = L((a_1 + \ldots + a_n)(a_1 + \ldots + a_n)^*)$.

Zusammenfassung

Die regulären Sprachen werden ausgehend von den Sprachen $\emptyset$ und $\{a\}$ durch die Operationen Vereinigung, Konkatenation und Stern erzeugt.

Als Folgerung des Satzes von Kleene (siehe unten) ergibt sich, dass die regulären Sprachen auch unter Durchschnitt und Komplement abgeschlossen sind.

Deterministische endliche Automaten: DFA

Ein endlicher deterministischer Automat $\mathcal{A}$ besitzt einen ausgezeichneten **Angangszustand** $q_0 \in Q$ und eine **Menge** $A \subseteq Q$ **akzeptierender Zustände**.

Also

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

Q	endliche, nicht-leere **Zustandsmenge**
$q_0 \in Q$	**Anfangszustand**
$A \subseteq Q$	Menge der **akzeptierenden Zustände**
$\delta : Q \times \Sigma \to Q$	**Übergangsfunktion.**

Berechnung von $\mathcal{A}$ auf $w = a_1 \dots a_n \in \Sigma^*$

Zustandsfolge $q_0, \dots, q_n$ mit $q_{i+1} = \delta(q_i, a_{i+1})$ für $0 \leq i < n$

$$q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots \; q_{n-1} \xrightarrow{a_n} q_n$$

analog: *Lauf* von $q \in Q$ aus (nicht notw. von q_0 aus)

führt zu eindeutiger Fortsetzung $\hat{\delta}$ von δ:

$$\begin{aligned} \hat{\delta}\colon Q \times \Sigma^* &\longrightarrow Q \\ (q, w) &\longmapsto \hat{\delta}(q, w) \in Q \end{aligned}$$

$\hat{\delta}(q, w)$ der (!) Endzustand des Laufs auf w von q aus

induktiv definiert

DFA: von $\mathcal{A}$ erkannte/akzeptierte Sprache

$w = a_1 \dots a_n$ mit Berechnung $q_0, \dots, q_n$ $\qquad q_n = \hat{\delta}(q_0, w)$

$$\mathcal{A} \quad \begin{cases} \textbf{akzeptiert } w & \text{falls } q_n \in A \\ \textbf{verwirft } w & \text{falls } q_n \notin A \end{cases}$$

die von $\mathcal{A}$ **akzeptierte/erkannte Sprache**:

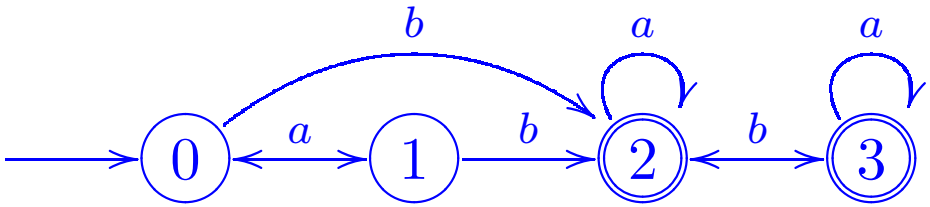
$$\begin{aligned} L(\mathcal{A}) &:= \{w \in \Sigma^* : \ \mathcal{A} \text{ akzeptiert } w \ \} \\ &= \{w \in \Sigma^* : \hat{\delta}(q_0, w) \in A\} \end{aligned}$$

Nicht-deterministische endliche Automaten, NFA

- statt Transitionsfunktion δ: **Transitionsrelation Δ**.
- NFA $\mathcal{A}$ akzeptiert Wort w, gdw. **mindestens eine** akzeptierende Berechnung auf w existiert.
- Kann durch DFA simuliert werden: **Potenzmengentrick!**

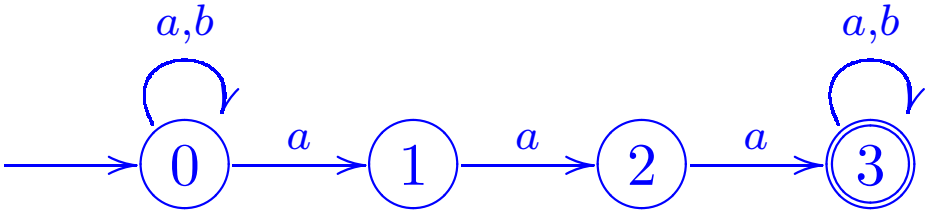
Beispiele mit $\Sigma = \{a, b\}$

DFA $\mathcal{A}_1$



$L(\mathcal{A}_1) = L(a^*b(a+b)^*)$

NFA $\mathcal{A}_2$



$L(\mathcal{A}_2) = L\big((a+b)^*aaa(a+b)^*\big)$

Satz von Kleene:

Eine Σ-Sprache L ist genau dann DFA/NFA-erkennbar, wenn L regulär ist.

Minimierung von DFA's

- Zu jedem DFA $\mathcal{A}$ gibt es einen bis auf Isomorphie eindeutig bestimmten äquivalenten DFA $\mathcal{A}'$ (d.h. $L(\mathcal{A}) = L(\mathcal{A}')$) mit minimaler Zustandsmenge: **Äquivalenzautomat!**
- Minimisierung eines gegebenen DFA $\mathcal{A}$ durch **induktive Verfeinerung von Äquivalenzrelationen** auf Zustandsmenge Q so lange dies erzwungen wird. Neue minimale Zustandsmenge: Äquivalenzklassen der so erhaltenen Äquivalenzelation.

Minimierung algorithmisch: induktive Verfeinerung

Tests für x der Länge $i = 0, 1, \ldots$

$q \quad \not\sim_0 \quad q'$ **gdw.** **nicht**$(q \in A) \Leftrightarrow (q' \in A)$

$q \quad \not\sim_{i+1} \quad q'$ **gdw.** $q \not\sim_i q'$ oder $(\exists a \in \Sigma)\ \delta(q,a) \not\sim_i \delta(q',a)$

Sobald $\sim_{i+1} = \sim_i$ $(=: \sim)$, fasse Zustände in $\sim$-Klassen zusammen.

Das Pumping Lemma für reguläre Sprachen

Theorem: Sei $L \subseteq \Sigma^*$ regulär. Dann existiert $n \in \mathbb{N}$ derart, dass sich jedes $x \in L$ mit $|x| \geq n$ zerlegen lässt in $x = uvw$, $v \neq \varepsilon$, $|uv| \leq n$ und für alle $m \in \mathbb{N}$

$$u \cdot v^m \cdot w = u \cdot \underbrace{v \cdots v}_{m \text{ mal}} \cdot w \in L.$$

Beweis: Schubfachprinzip!

Korollar: $L = \{a^n b^n : n \in \mathbb{N}\}$ für $a, b \in \Sigma, a \neq b$ ist nicht-regulär.

Grammatiken

Idee: Spezifikation einer Sprache durch **Erzeugungsprozesse**

$G = (\Sigma, V, P, X_0)$

Σ **Terminalalphabet,**

V endliche Menge von **Variablen**, $V \cap \Sigma = \emptyset$,

$X_0 \in V$ **Startvariable/Startsymbol,**

P endliche Menge von **Produktionen/Regeln**

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

Produktionen

$(v, v') \in P$ Produktion/Regel von G: $\boxed{v \to v'}$

linke Seite: $v \in (V \cup \Sigma)^+ = (V \cup \Sigma)^* \setminus \{\varepsilon\}$,

rechte Seite: $v' \in (V \cup \Sigma)^*$.

Erlaubt in $(V \cup \Sigma)$-Wörtern **Ersetzung von v durch v':**

$uvw \to_G uv'w$ (direkter Ableitungsschritt in G).

$\to_G^*$:$\equiv$ reflexiver, transitiver Abschluss von $\to_G$.

- $w \in (\Sigma \cup V)^*$ **ableitbar in G** gdw. $X_0 \to_G^* w$.
- $L(G) = \{w \in \Sigma^* : X_0 \to_G^* w\}$ die von G erzeugte Σ-Sprache.

Chomsky-Hierarchie von Grammatiken G

X, Y Variablen in V, $a \in \Sigma$, $v, v' \in (\Sigma \cup V)^*$

G regulär oder Typ 3

Produktionen **rechtslinear**: $X \to \varepsilon$, $X \to a$, $X \to aY$

G kontextfrei oder Typ 2

alle Produktionen haben als linke Seite

eine **einzelne Variable**: $X \to v$

G kontextsensitiv oder Typ 1

alle Produktionen **nicht verkürzend**: $v \to v'$, $|v'| \geq |v|$

bis auf ggf. eine *harmlose* ε-Produktion

G allgemein oder Typ 0: keine Einschränkungen.

Pumping Lemma für kontextfreie Sprachen

Theorem: Für jede kontextfreie Sprache $L \subseteq \Sigma^*$ existiert ein $n \in \mathbb{N}$, so dass sich jedes $x \in L$ mit $|x| \geq n$ zerlegen lässt in $x = yuvwz$, wobei $uw \neq \varepsilon$ und $yu^m vw^m z \in L$ für alle $m \in \mathbb{N}$. Man kann dabei u, v, w so wählen, dass $|uvw| \leq n$.

Im Beweis verwendet: Typ-2-Grammatiken können (bis auf harmlose ε-Produktionen) effektiv auf **Chomsky-Normalform gebracht werden**, d.h. nur Produktionen der Form

$$X \to YZ, \quad X \to a.$$

Hierarchiesatz

Typ 3 $\subset$ Typ 2 $\subset$ Typ 1 $\subset$ Typ 0 $\subset$ beliebige Sprache.

Alle Inklusionen sind **echt!**

Trennende Beispiele:

- $L = \{a^n b^n : n \in \mathbb{N}\} \in$ Typ 2 $\setminus$ Typ 3.
- $L = \{a^n b^n c^n : n \in \mathbb{N}\} \in$ Typ 1 $\setminus$ Typ 2.
- HP $\in$ Typ 0 $\setminus$ Typ 1, da Wortproblem für Typ-1-Sprachen entscheidbar, aber HP (Halteproblem) unentscheidbar ist.
- Die Sprache $\overline{\mathrm{HP}}$ ist nicht in Typ 0.

Charakterisierungen

- **Regulär (Typ 3):** durch reg. Ausdrücke definiert=von DFA/NFA erkannt= von regulärer (Typ-3-)Grammtik erzeugt.
- **Kontextfrei (Typ 2):** von kontextfreier (Typ-2-)Grammtik erzeugt=von PDA erkannt, wobei: PDA=NFA+Kellerspeicher.
- **Kontextsensitiv (Typ 1):** von kontextsensitiver (Typ-1-)Grammtik erzeugt=von linear platzbeschränkter nichtdeterministischer Turingmaschine erkannt.
- **Rekursiv aufzählbar (Typ 0):** von beliebiger Grammatik erzeugt=von DTM erkannt, wobei DTM=DFA+unbeschränktem Lese/Schreibzugriff.

Wortprobleme

Das Wortproblem für

- **reguläre Sprachen** ist **entscheidbar** (folgt aus Charakterisierung durch DFA's)
- **kontextfreie Sprachen** ist **entscheidbar** (folgt aus dem CYK-Algorithmus für Grammatiken in Chomsky-Normalform).
- für **kontextsensitive Sprachen** ist **entscheidbar** (da wegen $|v'| \geq |v|$ wenn $v \rightarrow v'$, nur endlich viele Ableitungen in Frage kommen).
- für **rekursiv aufzählbare Sprachen** ist im allgemeinen **unentscheidbar** (siehe HP).

Zusammenfassung der Abschlusseigenschaften in der Chomsky-Hierarchie

Typ	abgeschlossen unter				
	$\cup$	$\cap$	$\overline{}$	$\cdot$	$*$
3	+	+	+	+	+
2	+	−	−	+	+
1	+	+	+	+	+
0	+	+	−	+	+
bel. Σ-Sprachen	+	+	+	+	+