Formale Grundlagen der Informatik I

Zusammenfassung

http://public.phoenixsystems.de/fgi1

Erstellt von

Konstantin Simon Mar. Möllers

17. April 2012

Hamburg

Inhaltsverzeichnis

Vorwort

Diese Zusammenfassung wurde verfasst, um einen guten Überblick zu auftretenden **Definitionen** und **Regeln** zu liefern (nicht mehr, aber auch nicht weniger). Nur mit dieser Zusammenfassung zu lernen gibt nicht genug Stoff her, um die Klausur zu bestehen, aber um sich das Lernen zum Bestehen zu erleichtern. Ich freue mich jederzeit über Rückmeldungen im Facebook-Post oder als PM direkt an mich!

Diese Zusammenfassung ist zu finden unter `http://public.phoenixsystems.de/fgi1`

Dieses Dokument wurde vollständig in der Dokumentensprache LaTeX geschrieben und programmiert. Quelltextanfragen bitte via Mail an 1kmoelle@informatik.uni-hamburg.de.

Quellen & Ressourcen

- FGI-1-Skript zu **Regulären Mengen**: `http://www.informatik.uni-hamburg.de/WSV/teaching/vorlesungen/FGI1SoSe12/FGI1-12-14-Regulaere-Mengen.pdf` (Abruf am 8. April 2012)
- FGI-1-Skript zu **Kontextfreier Grammatik**: `http://www.informatik.uni-hamburg.de/WSV/teaching/vorlesungen/FGI1SoSe12/FGI1-15-17-CFL.pdf` (Abruf am 9. April 2012)
- Wikipedia-Artikel zu **Endlichen Automaten**: `http://de.wikipedia.org/wiki/Endlicher_Automat` (Abruf am 10. April 2012)
- Wikipedia-Artikel zum **Deterministischen endlichen Automaten**: `http://de.wikipedia.org/wiki/Deterministischer_endlicher_Automat` (Abruf am 10. April 2012)
- Wikipedia-Artikel zum **Pumping Lemma**: `http://de.wikipedia.org/wiki/Deterministischer_endlicher_Automat` (Abruf am 10. April 2012)
- Präsentation von Berndt Farwer *(TGI)* zum **Pumping Lemma**: `http://www.informatik.uni-hamburg.de/TGI/lehre/vl/SS01/F2/Files/uvw-web-Dateien/uvw-web1.html`

Kapitel 1: Grundlagen

Gehalten am 2. April, Folien 1 bis 33, Skript „Reguläre Mengen“

Teil I. Einführung & Motivation

Grundlagenkiste: *Etwas mehr als nur ein Blick in diese Kiste ermöglicht Einblick in Zusammenhänge und bessere Lösungen in den Anwendungen!*

I.1 Motivation

- **Formale Sprachen**
  - Zeichenvorrat Σ
  - Menge aller Zeichenketten Σ^*
  - Sprache $\mathcal{L}(A)$: eine Menge von Zeichenketten
- **Sprachfamilien** angeordnet nach Komplexität (endl. Mengen am simpelsten, abzählbare Mengen am kompliziertesten)

Teil II. Notationen

Wir unterscheiden grundsätzlich **Mengen** (Ansammlungen von Elementen) und **Aussagen** (logische Schlussfolgerungen, Ausdrücke, Definitionen usw.). Im Folgenden sind einige Regeln und Techniken zu Mengen und Aussagen notiert.

II.1 Aussagenlogik

Seien A und B Aussagen (die entweder wahr oder falsch sein können).

- $A \wedge B$: Sowohl A als auch B sind wahr
- $A \vee B$: Entweder A oder B ist wahr – oder beide.
- $A \Rightarrow B$: Wenn A wahr ist, dann auch B
- $A \Leftrightarrow B$: Wenn A wahr ist, dann auch B – und umgekehrt.
- $\forall x.A$: Für alle x gilt A. (Wobei der Wahrheitswert von A von x abhängen kann).
- $\exists x.A$: Für mindestens ein x gilt A (Wobei der Wahrheitswert von A von x abhängen kann).

II.2 Mengenoperationen

- **Vereinigung** $A \cup B := \{x \mid x \in A \text{ oder } x \in B\}$
- **Schnitt** $A \cap B := \{x \mid x \in A \text{ und } x \in B\}$

- **Komplement A \ B** $:= \{x \in A \mid x \in A \text{ und } x \notin B\}$
- **Potenzmenge** $\mathcal{P}(\mathrm{A}) = \{B \mid B \subseteq A\}$
  - Es gilt: $|\mathcal{P}(A)| = 2^{|A|}$
  - Die Relation „$\subseteq$“ ist eine *partielle Ordnung* auf der Potenzmenge $\mathcal{P}(A)$. *(keine totale!)*
- **Kartesisches Produkt A × B** $:= \{(a, b) \mid a \in A, b \in B\}$
- **Komplexprodukt A · B** $:= \{a \odot b \mid a \in A, b \in B\}$
- **Potenzen**
  - Menge aller *Tupel* (Paare): $A^2 := A \times A$
  - Menge aller *Tripel*: $A^3 := A \times A \times A$
  - Menge aller *Quadrupel*: $A^4 := A \times A \times A \times A$
  - Menge aller *Quintupel*: $A^5 := A \times A \times A \times A \times A$
- **Kardinalität |A|** ist die Anzahl der Elemente der Menge A.
- **Von A erzeugte Unterhalbgruppe** $\mathrm{A}^+ := \bigcup\limits_{i \geq 1} A^i$ mit $A^1 := A$ und $A^{i+1} := A^i \cdot A$
- **Von A erzeugtes Untermonoid** $\mathrm{A}^* := A^+ \cup \{e\}$

II.3 Notwendige und hinreichende Bedingung

Für zwei Aussagen A und B gilt:

- **notwendige Bedingung**: Damit B gelten kann ist es *notwendig*, dass auch A gilt. Es muss aber nicht A wegen B gelten!
  $\boxed{A \Rightarrow B}$
- **hinreichende Bedingung**: Gilt B, dann ist dies *hinreichend* (oder ausreichend) dafür, dass auch A gilt. Es muss aber nicht B aus A gelten!
  $\boxed{A \Leftarrow B}$
- **Notwendige und zugleich hinreichende Bedingung**: Gilt A, dann gilt B, gilt B, dann gilt auch A.
  $\boxed{A \Leftrightarrow B}$

Teil III. Beweistechniken

- **Direkter Beweis**

  Beweisen einer Behauptung durch eine Folge von Schlussfolgerungen.

  $\boxed{\text{Zeige } A \Rightarrow B}$
- **Indirekter Beweis**

Zeigen, dass unter Annahme des Gegenteils auch das Gegenteil der Behauptung bewiesen wird.

$\boxed{\text{Aus } \neg B \Rightarrow \neg A \text{ folgt } A \Rightarrow B}$

- **Beweis durch Widerspruch**

  Annahme des Gegenteils und Behauptung zu einem Widerspruch führen.

  $\boxed{\text{Aus } (A \wedge \neg B)\lightning \text{ folgt } A \Rightarrow B}$

- **Beweis durch Inklusion**

  Durch die Beweise, dass Menge A in Menge B und B auch in A enthalten ist, wird gezeigt, dass die Mengen identisch sind.

  $\boxed{A \subseteq B \wedge A \supseteq B \Rightarrow A = B}$

- **Induktionsbeweis** *(vollständige Induktion)*

  Eine Aussage $A_n, n \in \mathbb{N}$ wird beweisen, indem man zunächst im *Induktionsanfang* A_1 zeigt und dann, im *Induktionsschritt*, beweist, dass jede Aussage auch für A_{n+1} gilt.

  $\boxed{A_1 \wedge A_{n+1} \Rightarrow A_n}$

Teil IV. Wörter & Sprachen

IV.1 Grundsätzliche Begriffe

- Ein **Alphabet** Σ ist eine (total geordnete) endliche Menge von unterschiedlichen **Zeichen**.
- Die **Konkatenation** $\circ$ ist der Operator des freien Monoids.
- Das **freie Monoid** *(oder* Kleene*sche Hülle)* $(\Sigma^*, \circ)$ definiert die Hintereinanderschreibung der einzelnen Zeichen als *Monoid-Operation*.
- Das **leere Wort** ϵ ist das *neutrale Element*.
- $w^k := \underbrace{ww\cdots w}_{k}$ mit $w^0 := \epsilon$

IV.2 Halbgruppe

Für **Halbgruppen** $H = (M, *)$ gilt:

- *Assoziativgesetz*
  $(a * b) * c = a * (b * c)$
- Eine binäre (2-stellige) Operation $*$
- *Abgeschlossenheit* bezüglich ihrer Elemente, also:
  $\forall x, y \in M \Rightarrow x * y \in M$

IV.3 Monoid

Für **Monoide** $X = (M, *)$ gelten:

- Die *Halbgruppenaxiome*
- *Existenz eines neutralen Elements* ϵ bezüglich $*$, für das gilt:
  $x * \epsilon = x$ mit $x, \epsilon \in M$ (linksneutral)
  $\epsilon * x = x$ mit $x, \epsilon \in M$ (rechtsneutral)
- **Feststellung:** Ist $(M, *)$ ein Monoid, so besitzt jedes $a \in M$ höchstens ein Inverses. (Im DM-Skript Seite 99)

Teil V. Formale Sprachen

- Eine Sprache muss durch eine *endliche Repräsentation* (endliche Mengen etc.) dargestellt werden.
- Die Beschreibung einer Sprache muss *eindeutig* sein.
- Es kann *unterschiedliche Beschreibungen* für dieselbe Sprache geben.
- **Einfache Konzepte**:
  - endliche Automaten
  - reguläre (eig. rationale) Ausdrücke
  - Typ-3 Grammatiken (rechts- und linkslineare)

Kapitel 2: Endlicher Automat, Potenzautomat

Gehalten am 3. April, Folien 34 bis 91

Teil I. Deterministischer Endlicher Automat (DFA)

- DFA: ***d**eterministic **f**inite **a**utomaton*
- **deterministisch**: eindeutiger Ablauf, *vorhersehbare* Zustandswechsel.
- **endlich**: endliche Steuerung durch eine endliche *Zustandsmenge*.
- **Automat**: eine selbstausführende Maschine.

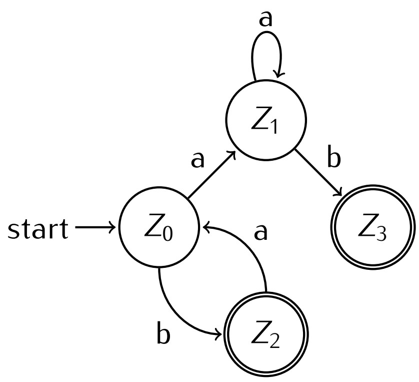
I.1 Definition

Beschrieben durch das *Quintupel*

$$A := (Q, \Sigma, \delta, q_0, F)$$

- Q ist eine endliche Menge von **Zuständen**
- Σ ist ein endliches **Alphabet** von Eingabesymbolen
- $\delta\colon Q \times \Sigma \to Q$ ist eine nicht notwendigerweise totale **Überführungsfunktion**
- $q_0 \in Q$ ist der **Startzustand** *(keine Menge!)*
- $F \subseteq Q$ ist die Menge der **Endzustände**

I.2 Beispiel



- Er ist **deterministisch**, da an keinem Zustand ein Zeichen mehrfach auftritt
- Menge der Zustände $Q = \{Z_0, Z_1, Z_2, Z_3\}$
- Alphabet $\Sigma = \{a, b\}$
- Übergangsfunktion δ:

$Z \in Q$	$\sigma \in \Sigma$	$\delta(Z, \sigma) \in Q$
Z_0	a	Z_1
Z_0	b	Z_2
Z_1	a	Z_1
Z_1	b	Z_3
Z_2	a	Z_0

- Anfangszustand Z_0
- Menge der Endzustände $F = \{Z_2, Z_3\}$

I.3 Erweiterte Übergangsfunktion

Definiert durch $\delta^*\colon Q \times \Sigma^* \Rightarrow Q$ mit der Zustandsmenge Q, dem Alphabet Σ und der Überführungsfunktion δ, so dass gilt:

$$\delta^*\big(z, xw\big) := \delta^*\big(\delta(z, x), w\big)$$

für alle Zustände $z \in Q$, alle Symbole $x \in \Sigma$ und alle Wörter $w \in \Sigma^*$, sowie

$$\forall q \in Q\colon \delta^*(q, \epsilon) := q$$

I.4 Vollständiger DFA und andere DFA

Ein DFA $A := (Q, \Sigma, \delta, q_0, F)$ heißt:

- **vollständig** *(vDFA)*
  genau dann, wenn für jedes $(p, x) \in Q \times \Sigma$ ein $q \in Q$ existiert, so dass $q = \delta(p, x)$ ist.
- **initial zusammenhängend** *(izDFA)*
  genau dann, wenn zu jedem Zustand $p \in Q$ ein Wort $w \in \Sigma^*$ existiert, mit $p = \delta^*(z_0, w)$.
- Zwei DFA A und B heißen **äquivalent**
  genau dann, wenn sie die gleiche Sprache akzeptieren: $\mathcal{L}(A) = \mathcal{L}(B)$

Teil II. Die Familie $\mathcal{REG}$ der regulären Sprachen

- beschrieben durch die **Menge der von einem DFA akzeptierten Wörter**
- Wir bezeichnen diese Menge mit $\mathcal{REG}$
- DFA_Σ bezeichnet die **Familie aller Sprachen** $\mathcal{L} \subseteq \Sigma^*$, welche von DFAs mit dem Alphabet Σ akzeptiert werden können.

$$\text{(I)} \quad \text{DFA}_\Sigma = \{\mathcal{L} \subseteq \Sigma^* \mid \mathcal{L} = (A) \text{ für einen DFA } A\}$$
$$\text{(II)} \quad \mathcal{REG}_\Sigma := \text{DFA}_\Sigma \text{ sowie } \mathcal{REG} = \bigcup_{\substack{\Sigma \text{ ist endl.} \\ \text{Alphabet}}} \mathcal{REG}_\Sigma$$

II.1 Akzeptierte Sprache des DFA

Die von dem DFA A **akzeptierte Sprache** ist definiert als die Menge:

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

II.2 Kochrezept: Wie beweist man $\mathcal{L}(B) = \mathcal{L}$?

Wir führen einen Inklusionsbeweis durch, also wir zeigen:

$$\mathcal{L}(B) \subseteq \mathcal{L} \wedge \mathcal{L}(B) \supseteq \mathcal{L}$$

1. Zu zeigen: $\mathcal{L}(B) \subseteq \mathcal{L}$
   (a) Sei $w \in \mathcal{L}(B)$, dann müssen wir $w \in \mathcal{L}$ zeigen.
   (b) Nach Definition 2.2.1 gilt $w \in \mathcal{L}(B) \Rightarrow \delta^*(z_0, w) \in F$.
   (c) Von z_0 aus Umstände des Automaten nutzen, um eindeutige Möglichkeiten zu finden.
   (d) Durch diese Möglichkeiten beschriebene Wörter so bilden, dass $w \in \mathcal{L}$ steht. □

2. Zu zeigen: $\mathcal{L} \subseteq \mathcal{L}(B)$
   (a) Sei $w \in \mathcal{L}$, dann müssen wir $w \in \mathcal{L}(B)$ zeigen.
   (b) Für $w \in \mathcal{L}$ suchen wir einen $\langle$Ausdruck$\rangle$ (beispielsweise eine Potenz des Sprachelements).
   (c) Wir weisen durch Verfolgung von Zuständen nach, dass wir wieder bei einem Endzustand ankommen.
   (d) Dadurch zeigen wir $w = \langle\text{Ausdruck}\rangle \in \mathcal{L}(B)$. $\square$

Hierdurch wurde im Groben das Beweisverfahren geschildert. Um ein Beispiel zu sehen, schau dir im Skript Folien 58f an.

Teil III. Nichtdeterminitischer Endlicher Automat (NFA)

- NFA: ***n**ondeterministic **f**inite **a**utomaton*
- **nichtdeterministisch**: der nächste Zustand ist *mehrdeutig* und *nicht vorhersehbar*.
- **endlich**: endliche Steuerung durch eine endliche *Zustandsmenge*.
- **Automat**: eine selbstausführende Maschine.

III.1 Definition

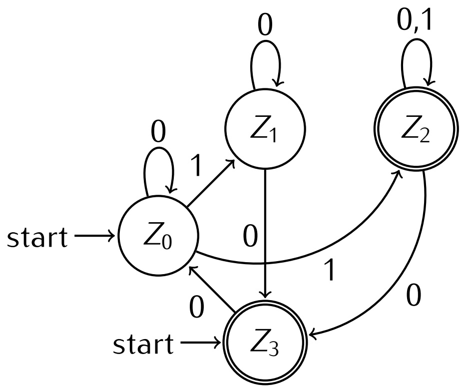
Beschrieben durch das *Quintupel*

$$A := (Q, \Sigma, \delta, S_0, F)$$

- Q ist eine endliche Menge von **Zuständen**
- Σ ist ein endliches **Alphabet** von Eingabesymbolen
- $\delta\colon Q \times \Sigma \to 2^Q$ ist eine **Überführungsfunktion**
- $S_0 \subseteq Q$ ist die Menge der **Startzustände** *(Hier: Menge!)*
- $F \subseteq Q$ ist die Menge der **Endzustände**

III.2 Beispiel

Gegeben sei folgender NFA:



- Er ist **nichtdeterministisch**, da an Z_0, Z_1 und Z_2 Zeichen mehrfach verwendet werden und mehrere Startzustände vorliegen
- Er ist **nicht vollständig**, da beispielsweise bei Z_1 für 1 keine Menge an Folgezuständen definiert ist.
- Menge der Zustände $Q = \{Z_0, Z_1, Z_2, Z_3\}$
- Potenzmenge der Zustände $2^Q = \Big\{\emptyset, \{Z_0\}, \{Z_1\}, \{Z_2\}, \{Z_3\}, \{Z_0, Z_1\}, \{Z_0, Z_2\}, \{Z_0, Z_3\}, \{Z_1, Z_2\}, \{Z_1, Z_3\}, \{Z_2, Z_3\}, \{Z_0, Z_1, Z_2\}, \{Z_0, Z_1, Z_3\}, \{Z_0, Z_2, Z_3\}, \{Z_1, Z_2, Z_3\}, \{Z_0, Z_1, Z_2, Z_3\}\Big\}$
- Alphabet $\Sigma = \{0, 1\}$
- Funktionstabelle zur **Übergangsfunktion** δ:

$Z \in 2^Q$	$\sigma \in \Sigma$	$\delta(Z, \sigma) \in 2^Q$	$Z \in 2^Q$	$\sigma \in \Sigma$	$\delta(Z, \sigma) \in 2^Q$
$\emptyset$	0	$\emptyset$	$\{Z_1, Z_2\}$	0	$\{Z_1, Z_2, Z_3\}$
$\emptyset$	1	$\emptyset$	$\{Z_1, Z_2\}$	1	$\{Z_2\}$
$\{Z_0\}$	0	$\{Z_0\}$	$\{Z_1, Z_3\}$	0	$\{Z_0, Z_1, Z_3\}$
$\{Z_0\}$	1	$\{Z_1, Z_2\}$	$\{Z_1, Z_3\}$	1	$\emptyset$
$\{Z_1\}$	0	$\{Z_1, Z_3\}$	$\{Z_2, Z_3\}$	0	$\{Z_0, Z_2, Z_3\}$
$\{Z_1\}$	1	$\emptyset$	$\{Z_2, Z_3\}$	1	$\{Z_2\}$
$\{Z_2\}$	0	$\{Z_2, Z_3\}$	$\{Z_0, Z_1, Z_2\}$	0	$\{Z_0, Z_1, Z_2, Z_3\}$
$\{Z_2\}$	1	$\{Z_2\}$	$\{Z_0, Z_1, Z_2\}$	1	$\{Z_1, Z_2\}$
$\{Z_3\}$	0	$\{Z_0\}$	$\{Z_0, Z_1, Z_3\}$	0	$\{Z_0, Z_1, Z_3\}$
$\{Z_3\}$	1	$\emptyset$	$\{Z_0, Z_1, Z_3\}$	1	$\{Z_1, Z_2\}$
$\{Z_0, Z_1\}$	0	$\{Z_0, Z_1, Z_3\}$	$\{Z_0, Z_2, Z_3\}$	0	$\{Z_0, Z_2, Z_3\}$
$\{Z_0, Z_1\}$	1	$\{Z_1, Z_2\}$	$\{Z_0, Z_2, Z_3\}$	1	$\{Z_1, Z_2\}$
$\{Z_0, Z_2\}$	0	$\{Z_0, Z_2, Z_3\}$	$\{Z_1, Z_2, Z_3\}$	0	$\{Z_0, Z_1, Z_2, Z_3\}$
$\{Z_0, Z_2\}$	1	$\{Z_1, Z_2\}$	$\{Z_1, Z_2, Z_3\}$	1	$\{Z_2\}$
$\{Z_0, Z_3\}$	0	$\{Z_0\}$	$\{Z_0, Z_1, Z_2, Z_3\}$	0	$\{Z_0, Z_1, Z_2, Z_3\}$
$\{Z_0, Z_3\}$	1	$\{Z_1, Z_2\}$	$\{Z_0, Z_1, Z_2, Z_3\}$	1	$\{Z_1, Z_2\}$

- Menge der Anfangszustände $S_0 = \{Z_0, Z_3\}$
- Menge der Endzustände $F = \{Z_2, Z_3\}$

III.3 Erweiterte Überführungsfunktion

Definiert durch $\delta^*\colon 2^Q \times \Sigma^* \Rightarrow 2^Q$ mit der Zustandsmenge Q, dem Alphabet Σ und der Überführungsfunktion δ, so dass gilt:

$$\delta^*(R, aw) := \delta^*\left(\bigcup_{p \in R} \delta(p, a), w\right)$$

III.4 Akzeptierte Sprache

Die von dem NFA A **akzeptierte Sprache** ist definiert als die Menge:

$$\mathcal{L}(A) := \{w \in \Sigma^* \mid \delta^*(S_0, w) \cap F \neq \emptyset\}$$

Teil IV. Potenzautomat

IV.1 Nichtdeterminismus deterministisch simulieren

Jede von einem **NFA** akzeptierte Menge kann auch von einem **initial zusammenhängenden, vollständigen DFA** akzeptiert werden und ist daher regulär.

- Endliche Zahl an *in einem Schritt erreichbaren Folgezuständen* bei *jedem* Automaten
- **Endliche Teilmengen** von Q
- Von der *Startzustandmenge* S_0 kann nur eine neue Teilmenge von Q erreicht werden
- Repräsentiert durch die Zustände eines **Potenzautomatens**

IV.2 Definition

Sei $A := (Q, \Sigma, \delta, S_0, F)$ ein NFA, so ist sein **Potenzautomat** definiert wie folgt:

$$B := (2^Q, \Sigma, \delta', S_0, F')$$

- **Menge der Endzustände** $F' := \{M \in 2^Q \mid M \cap F \neq \emptyset\}$
- **Menge der Startzustände** S_0 (abgebildet von A)
- **Überführungsfunktion** δ' ist $\forall M \in 2^Q$ und $\forall x \in \Sigma$ definiert durch:

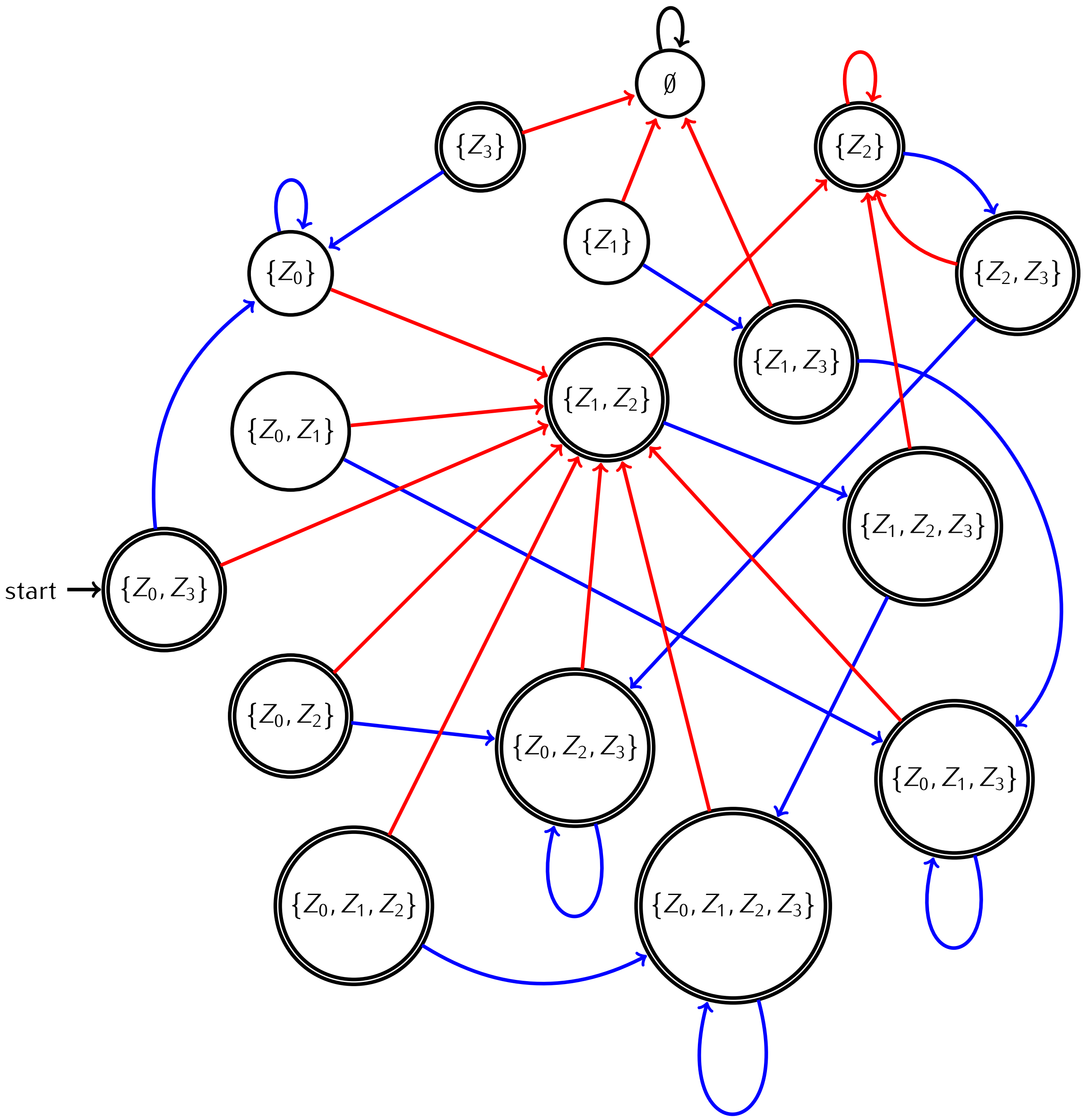
$$\delta'(M, x) := \bigcup_{p \in M} \delta(p, x)$$

IV.3 Beispiel

Wir bilden den Potenzautomaten zu unserem NFA aus dem obigen Beispiel:

- Wir erwarten (aufgrund der Potenzmenge) $2^{|Q|} = 2^4 = 16$ Zustände.
- Wir haben nur noch **einen Startzustand** *(DFAs akzeptieren nur einen).*
- Der gewonnene Automat ist **vollständig**.

(Blaue Übergänge stehen für 0, rote für 1 und schwarze für 0,1)



IV.4 Schlussfolgerungen und Anwendungen

NFAs akzeptieren dieselbe Sprachfamilie $\mathcal{REG}$ wie DFAs.

- Sind L_1 und L_2 reguläre Sprachen, dann ist auch $L_1 \cup L_2$ regulär.
- formal: $A = (Q_1 \uplus Q_2 \uplus \{q_A\}, \Sigma_1 \cup \Sigma_2, \delta_A, q_A, F_1 \cup F_2)$. [1]
- weitere **Abschlusseigenschaften**:
  - Vereinigung
  - Produkt
  - Durchnittsbildung
  - Komplement
  - Homomorphismen

Teil V. Minimale Automaten

V.1 Feststellungen

- Zu jeder Sprache gibt es **immer** vDFAs mit kleinstmöglicher Zustandsanzahl.
- Diese Automaten sind bis auf die Zustandsbezeichnungen **isomorph**.
- **minimaler vDFA der Sprache $\mathcal{L}$**: vDFA mit kleiner anzahl an Zuständen, der $\mathcal{L}$akzeptiert.
- effektive Konstruktion mithilfe von **Äquivalenzklassen** der sog. **Nerode-Äquivalenz**.

V.2 Nerode-Äquivalenz

Sei A ein DFA, dann ist die automatenspezifische Äquivalenzrelation $R_A \subseteq \Sigma^* \times \Sigma^*$ erklärt durch:

$$u \; R_A \; v \text{ genau dann, wenn } \delta^*(z_0, u) = \delta^*(z_0, v)$$

Sei $L \subseteq \Sigma^*$ eine reguläre Menge, dann ist die **Nerode-Äquivalenz** oder **syntaktische Rechtskongruenz** $R_L \subseteq \Sigma^* \times \Sigma^*$

$$u \; R_L \; v \text{ genau dann, wenn } \forall w \in \Sigma^* \colon (uw \in \mathcal{L} \Leftrightarrow vw \in \mathcal{L}).$$

[1] Das Zeichen „$\uplus$“ definiert die Disjunktion von zwei Mengen ohne deren Schnittmenge *(Kontravalenz)*: $A \uplus B = A \cup B \wedge A \cap B = \emptyset$

V.3 Äquivalente Aussagen

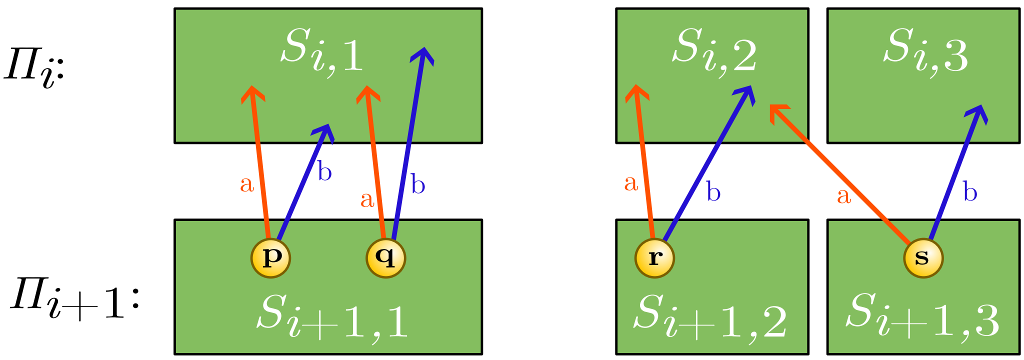
Die folgenden Aussagen sind ¨aquivalent:

1. $\mathcal{L} \in \Sigma^*$ ist regulär.
2. $\mathcal{L}$ ist Vereinigung von Äquivalenzklassen einer rechtsinvarianten Äquivalenzrelation mit endlichem Index.
3. Die Nerode-Äquivalenz R_L hat endlichen Index.

V.4 Konstruktion

Sei A ein beliebiger DFA. Von Q werden wiederholt **Partitionen** (feinere, disjunkte Zerlegungen) erzeugt:

1. die **erste Partition** von Q ist $\Pi_1 := \{S_{1,1}, S_{1,2}\}$, wobei $S_{1,1} := F$ *(Menge der Endzustände)* und $S_{1,1} := Q \setminus F$ *(Menge aller anderen Zustände)* sei.
2. Sei $\Pi_i = \{S_{i,1}, S_{i,2}, \ldots, S_{i,k_i}\}, k_i \geq 0$. Bilde Partition Π_{i+1} gemäß folgender Bedingung: Zwei Zustände $p, q \in Q$ gehören genau dann zum selben Block, wenn für jedes $a \in \Sigma$ gilt: $\delta(q, a)$ und $\delta(p, a)$ liegen im gleichen Block in Π_i.
3. Falls für ein $i \in \mathbb{N}$ irgendwann $\Pi_{i+1} = \Pi_i$ gilt, dann ist „Pi_i die **Zustandsmenge des minimalen Automaten**, andernfalls fahre man mit Schritt 2 fort.



Alle Folgezustände von p und q liegen im selben Block S_{i+1}!

Nur $\delta(s,a)$ liegt im selben Block der Folgezustände von r, darum liegen r und s **nicht** im selben Block S_{i+1}!

Kapitel 3: Konfiguration eines NFA, ϵ-NFA, Pumping-Lemma

Gehalten am 10. April, Folien 96 bis 120

Teil I. Konfiguration eines NFA

I.1 Definition

Zur Beschreibung der **aktuellen Situation**, in der sich ein NFA befindet, reicht der eingenommene Zustand alleine nicht aus, wichtig ist die noch zu verarbeitende Eingabe!

Für einen NFA A heißt $k \in Q \times \Sigma^*$ **Konfiguration** genau dann, wenn

- $k = (p, w)$ mit $w \in \Sigma^*$ und $p \in Q$, das heißt:
  A befindet sich *im Zustand* p und die *noch zu verarbeitende Eingabe* ist w.
- $w = \epsilon$: Die Eingabe wurde vollständig gelesen.

I.2 Folgekonfiguration und die reflexive, transitive Hülle $\vdash$

(q, w) ist genau dann eine **Folgekonfiguration** von (p, aw), wenn $q \in \delta^*(p, a)$. Wir notiert dies durch:

$$(p, aw) \vdash (q, w)$$

$\vdash^*$ bezeichnet die **reflexive, transitive Hülle** von $\vdash$, das heißt es existieren sämtliche Folgekonfigurationen, die durch Verknüpfung anderer Folgekonfigurationen gebildet werden können (oder die Folgekonfiguration ist die aktuelle Konfiguration). Formal ausgedrückt für Konfigurationen k_1, k_2, k_3:

$$k_1 \vdash^* k_2 \iff (\exists k_2 \colon k_1 \vdash^* k_2 \land k_2 \vdash k_3) \lor k_1 = k_3$$

Teil II. ϵ-NFA

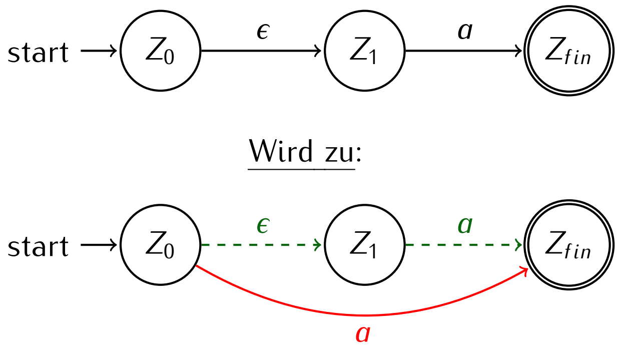
II.1 Definition

Ein **endlicher Automat mit ϵ-Übergängen** (ϵ-**FA**), ist ein NFA $A := (Q, \Sigma, \delta, S_0, F)$, mit:

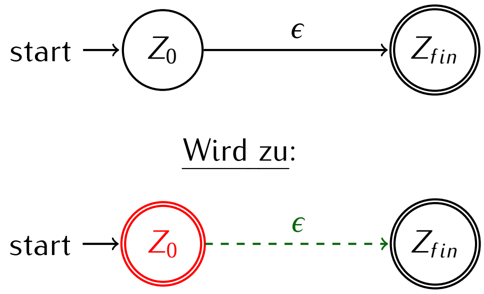
$$\delta \subseteq Q \times \big(\Sigma \cup \{\epsilon\}\big) \times Q.$$

II.2 ϵ-FA äquivalent zum NFA

Zusammenfassen von Kanten:



Schaffung neuer Endzustände:



Teil III. Pumping Lemma (uvw-Theorem)

III.1 Grundidee

Ein Automat A mit einer endlichen Anzahl n an Zuständen kann ein Wort w mit einer Wortlänge $|w| = n - 1$ lesen, so dass kein Zustand mehr als einmal erreicht wird.

Wenn nun aber die Wortlänge $|w| \geq n$ ist, muss mindest eine Zustand mehr als einmal erreicht werden, so dass eine **Schleife** entstanden ist. Sobald aber eine Schleife im Automaten existiert, kan nauch ein unendlich langes Wort akzeptiert werden, da die Schleife imemr wieder durchlaufen werden kann.

III.2 Bedingungen

Für jedes Wort z einer Sprache $\mathcal{L}$ zu einem Automaten mit n Zuständen und $|z| \geq n$

- existiert eine **Zerlegung** in u, v und w: $z = u \cdot v \cdot w$
- gibt ist eine **Schleife** nach max. n Zeichen: $|uv| \leq n$
- ist die **Schleifeninschrift** nicht leer: $|v| \geq 1$
- liegt die *notwendige Bedingung* für eine **reguläre Sprache** vor: $\forall i \in \mathbb{N}_0 . u \cdot v^i \cdot w \in \mathcal{L}$

III.3 Kochrezept: $\mathcal{L} \notin \mathcal{REG}$ mit Pumping Lemma beweisen

Zu zeigen: $\mathcal{L} \notin \mathcal{REG}$

1. Man wähle ein *akzeptiertes Wort* $z \in \mathcal{L}$.
2. Es ist vorauszusetzen, dass die Wortlänge $|z| \geq n$ ist, mit $n \in \mathbb{N}$.
3. Man zerlege z in $u \cdot v \cdot w$, wobei $|v| \geq 1$ sein muss.
4. Aus $|z| \geq n$ muss folgen, dass es eine Schleife gibt, also $|uv| \leq n$.
5. Wäre $\mathcal{L}$ regulär, so wäre nun allerdings auch $uv^0w \in \mathcal{L}$ (da die Schleife bei einer regulären Sprache weggelassen werden könnte), allerdings führt dies zu einem Widerpruch! $\lightning$
6. Damit kann $\mathcal{L}$ nicht regulär sein, es folgt also $\mathcal{L} \notin \mathcal{REG}$. $\square$

Kapitel 4: Reguläre Ausdrücke, GFA

Gehalten am 16. April, Folien 121 bis 139

Teil I. Operatoren regulärer Ausdrücke

I.1 Definitionen regulärer Ausdrücke

Die **regulären Ausdrücke** über einem endlichen Alphabet Σ sind definiert durch:

1. $\emptyset$ als regulärer Ausdruck für die **leere Menge** $M_\emptyset := \emptyset$
2. Jedes $a \in \Sigma$ für die Menge $M_a := \{a\}$
3. Sind A und B reguläre Ausdrücke, welche die Mengen M_A und M_B beschreiben, dann sind **induktiv** folgende regulären Ausdrücke definiert:
   - $(A + B)$ für $M_A \cup M_B$
   - $(A \cdot B)$ für $M_A \cdot M_B$
   - A^* für M_A^*
   - A^+ für M_A^+

Nur die mit den obigen Definitionen erzeugten Ausdrücke sind regulären Ausdrücke!

I.2 Wo ist das ϵ geblieben?

ϵ wird in regulären Ausdrücken durch $\emptyset^* = \{\epsilon\}$ beschrieben.

Dies begründet sich durch die Defintion des freien Monoids.

I.3 Klammerregeln

Regeln zur Klammerersparnis:

- unäre Operatoren vor binären ($*$ vor $\cdot$ und $+$)
- Punkt vor Strich ($\cdot$ vor $+$)

I.4 Reguläre Ausdrücke in der Praxis

Reguläre Ausdrücke werden häufig zur **Stringverarbeitung** in allen möglichen Programmiersprachen verwendet:

- *PHP* (`preg_match_all`, `preg_replace` etc.)
- *Java* (`String.split`, `String.replaceAll`, `String.matches` etc.)
- *JavaScript*, *Python*, *Perl* und weitere

Andere Beispiele sind die *Unix-Tools* `grep` und `egrep`, welche Wörter in Dateinamen mithilfe von regulären Ausdrücken suchen.

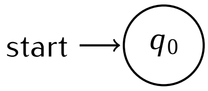
I.5 Tabelle der regulären Ausdrücke

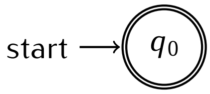
Folgende Tabelle enthält einige reguläre Ausdrücke zur Stringverarbeitung:

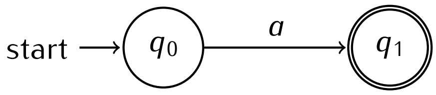
$\mathcal{E}$xpr	**Zweck**	**Anwendung**	
`\`	Maskiert das darauffolgende Zeichen	`\*`	Der Stern wird wie ein normaler Stern behandelt.
`^`	Anfang eines Worts	`^FGI`	Jeder Ausdruck muss mit „FGI" anfangen.
`$`	Ende eines Worts	`FGI$`	Jeder Ausdruck muss mit „FGI" enden.
`*`	Zeichen beliebig oft	`FGI*`	Findet FG, FGI, FGIIIIIII, ...
`+`	Zeichen mindestens einmal	`FGI+`	Findet FGI, FGIIIIIII, ...
`?`	Zeichen höchstens einmal	`FG?I`	Findet FI und FGI.
`.`	Beliebiges Zeichen	`F.I`	Findet FGI, FUI, FAI, FII, F4I, ...
`\d`	Beliebige Zahl	`FGI-\d`	Findet FGI-1, FGI-2, ...
`\w`	Alphanumerischer Ausdruck	`H\wllo`	Findet H4llo, Hallo, H3llo, H_llo ...
`\s`	Whitespace	`Wo\sist`	Findet Wo ist (mit Leerzeichen, Zeilenumbruch, Tabulator etc. dazwischen)
`\r`	Wagenrücklauf		
`\n`	Zeilenvorschub		
`\t`	Tabulator		
`\f`	Seitenvorschub		
(...)	Gruppierung	`(ab.)`	Merkt sich den Wert innerhalb der Klammern intern. Findet dabei aba, abc, ab9, ab_ etc.
[...]	Sammlung	`H[ae]llo`	Findet Hallo und Hello
`-`	„bis"-Zeichen in Sammlungen	`[a-z]`	Findet alle kleinen Buchstaben.

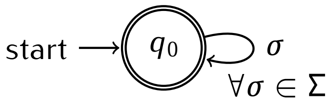
Teil II. Reguläre Ausdrücke bilden

II.1 Einfache Mengen sind regulär

- Die leere Menge ist regulär. 

- Die Menge $\emptyset^* = \{\epsilon\}$ ist regulär. 

- Die Menge $\{a\}$ ist regulär. 

- Σ^* ist regulär. 

- $\forall M$ ist M^+ nur abkürzend für $M \cdot M^*$.
- $\forall M$ ist $M \cdot \emptyset = \emptyset \cdot M = \emptyset$.

Teil III. GFA

Kapitel 5: Kontextfreie Grammatiken

Gehalten am 16. April, Folien 1 bis 41, Skript „Kontextfreie Grammatiken“

Sachregister