DisMod Begriffe:

**Mengen:**

Teilmenge: Menge, die gleich oder ein Teil einer anderen Menge ist

Echte Teilmenge: Menge, die ein Teil einer anderen Menge ist (nicht gleich!)

Vereinigung: Summe zweier Mengen

Komplement (Differenz): Differenz Zweier Menge (Menge A – Menge B)

Schnittmenge: Menge identischer Elemente aus Menge A und Menge B

Symmetrische Differenz (XOR): Menge unterschiedlicher Elemente aus Menge A und Menge B

Universum P: Menge aller verwendeten Elemente

Potenzmenge: Menge aller möglichen Kombinationen einer Menge (leere Menge nicht vergessen, ist in jeder Potenzmenge vorhanden)

Bsp: Potenzmenge( {0,1} ) = { Ø, {0}, {1}, {0,1} }

Mengenkomplement: Komplement von Menge $\overline{M}$ ist Differenz von Universum und Menge M
($\overline{M} = U \setminus M$)
(Überstriche = Alt + 0 7 7 3, nach Buchstabe)

**Logik:**

Semantische Folgerung |= : Wenn a |= b gilt, dann gilt für jede zu a passende Belegung:
Wenn a = 1, gilt auch b = 1 (jede Belegung die a erfüllt, erfüllt auch b)

Zusatz:
1 |= a → a allgemeingültig
a |= 0 → a unerfüllbar
a |= b ←→ (a → b) allgemeingültig ←→ (a oder ¬b) unerfüllbar
(Beobachtung 3.30. Skript)

Semantische Äquivalenz ≡ :
( a |= b ) und ( b |= a ) ←→ a ≡ b
a ≡ b ←→ (a ⇔ b) allgemeingültig
a allgemeingültig ←→ a ≡ 1

De Morgansche Regel:
¬( a und b ) ≡ ( ¬a oder ¬b )
¬( a oder b ) ≡ ( ¬a und ¬b )
(Word ist eine Hure #scheißsonderzeichen)

Zusatz:
( a → b ) ≡ ( ¬a oder b )
( a ⇔ b) ¬ ( ( a → b ) oder ( b → a ) ) ≡ ¬(a XOR b)
(a → ( a → b ) ) ≡ ( a → b )

Wahrheitstabelle für α = a XOR b

| a | b | (a XOR b) | α | Zeile |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 2 |
| 1 | 0 | 1 | 1 | 3 |
| 1 | 1 | 0 | 0 | 4 |

DNF (disjunktive Normalform [disjuktiv = (hier) Veroderung]:
Welche Kombinationen an Variablen kann ich haben, damit meine Funktion wahr ist?

→ ich benötige eine Kombination in meiner Wahrheitstabelle die eine 1-Zeile hat
→ KNF = Zeile2 oder Zeile3

= (¬a und b) oder (a und ¬b)

KNF (konjunktive Normalfunktion [konjunktiv = (hier) Verundung]:
Welche Kombination von Variablen darf ich nicht haben, damit meine Funktion wahr ist?

→ ich darf keine Kombination aus Variablen haben, die eine 0-Zeile ergeben (nicht Zeile1 und nicht Zeile4)
→ DNF = ¬Zeile1 und ¬Zeile4
= ¬(¬a und ¬b) und ¬(a und b)
(Morgansche Regel anwenden, ¬ in Klammer ziehen
= (a oder b) und (¬a oder ¬b)

Resolution: Nimm Disjunktionsterme und leite von ihnen Regeln ab:

Bsp (Vorlesung): (a → b) und (b → c)

⇔ (¬a oder b) und (¬b oder c)

→ (¬a oder b) und (¬b oder c)
⇔ {¬a, b}, {¬b, c}

→ {¬a, b}, {¬b, c}
{¬a, c}

{¬a, c} ⇔ (¬a oder c) ⇔ a → c

Ergibt sich bei einer Resolution kein Ergebnis, so ist der Term unerfüllbar

Bsp: a und ¬a

{¬a }, {a}
ε
→ unerfüllbar

Zusatz:

(¬a oder b) und (¬b oder a)

{¬a, b}, {¬b, a}
ε

Ist nicht legal, da a = 1 und b = 0 eine erfüllende Belegung für diesen Term wäre

**Graphen:**

Graph: Menge an Knoten V, die über Menge von Kanten E, miteinander verbunden sind G(V, E)

Komplementärgraph: Komplementärgraph (Konfliktgraph) $\overline{G}$, hat alle Knoten V, über die Kanten $\overline{E}$ verbunden (Kanten, die nicht in G vorliegen) [$\overline{E}$ kein offizieller Ausdruck]

Grad: Der Grad eines Knotens beschreibt über wie viele Kanten er mit beliebigen Knoten verbunden ist (zwei Kanten → Grad 2). Grad x eines Graphen sagt aus, dass all seine Knoten über x Kanten miteinander verbunden sind.

Euler Wege: Kantenfolge in einem Graphen G, die jede Kante genau einmal enthält (Start- und Endknoten müssen hier nicht identisch sein)

Euler Kreise: Kantenfolge in einem Graph G, die jede Kante genau einmal enthält und zusätzlich am Anfangsknoten endet (Start- und Endkonten müssen identisch sein)

Hamilton Weg: Kantenfolge in einem Graphen G, der jeden Knoten genau einmal durchläuft (Start- und Endknoten müssen nicht identisch sein)

Hamilton Kreis: Kantenfolge in einem Graphen G, die jeden Knoten genau einmal durchläuft und am Anfangskonten endet (Start- und Endknoten müssen identisch sein)

Zusammenhangskomponente: Die Zusammenhangskomponente eines Knotens v (in einem ungerichteten Graphen G), besteht aus allen Knoten w, die durch beliebige Wege von v aus erreicht werden können

Starke Zusammenhangskomponente: Ist der Graph G gerichtet, so besteht die starke Zusammenhangskomponente eines Knoten v aus allen Knoten w, die von v aus erreicht werden können und zusätzlich die von w aus v erreichen können (Hinweg von v nach w und Rückweg von w nach v sind möglich)

Zusammenhängend: Ein Graph G heißt zusammenhängend, wenn die Zusammenhangskomponente eines beliebigen Knotens v aus allen Knoten des Graphen G besteht (es existiert ein Knoten, von dem aus alle anderen Knoten erreicht werden können)

Stark zusammenhängend: Ein gerichteter Graph G heißt stark zusammenhängend, wenn die starke Zusammenhangskomponente eines beliebigen Knoten v aus allen Knoten des Graphen G besteht (es existiert ein Knoten, von dem aus alle anderen Knoten erreicht werden können)

Allg. Zusammenhängend = Jeder Knoten v in G kann jeden beliebigen anderen Knoten w erreichen

Knotenfärbung: Jedem Knoten eines Graphen wird eine Farbe zugewiesen. Konflikte treten auf, wenn zwei miteinander verbundene Knoten dieselbe Farbe haben (alle durch Kanten verbundene Knoten müssen unterschiedliche Farben haben)

Zusatz:
Besitzt ein Graph ein Dreieck, so gibt es mindestens drei unterschiedliche Farben bei einer Knotenfärbung

Chromatische Zahl: Kleinste Zahl an notwendigen Farben bei einer Knotenfärbung. Werden weniger Farben verwendet als die chromatische Zahl vorgibt, kann der Graph nicht konfliktfrei gefärbt werden.

Matching: Kantenmenge $E' \subseteq E$, Matching, wenn keine Knoten aus V mit zwei Kanten aus $E'$ verbunden ist (Es gibt keine zwei Kanten in E, die am gleichen Knoten aus V beginnen oder enden)

Perfektes Matching: Ein Matching, bei dem jeder Knoten aus V mit einer Kante aus $E'$ verbunden ist (Erklärungen für Matching/ perfektes Matching etwas verwirrender im Skript, hier mehr umgangssprachlich)

Isomorphie: Zwei Graphen G und $G'$ heißen isomorph, wenn es eine bijektive Abbildung von $f: V \rightarrow V'$ gibt, für die gilt: jeder Knoten i und j aus V, die in G verbunden sind, sind auch als f(i) und f(j) in $G'$ verbunden.
(Intuitiv: Man kann jedem Knoten in G einen Knoten in $G'$ zuordnen, der sich exakt gleich verhält)

Bipartit: Ein ungerichteter Graph heißt bipartit, wenn seine Knotenmenge V in zwei disjunkte Teilmengen V1 und V2 unterteilt werden kann, sodass jede Kante der Kantenmenge E, eines ihrer Enden in V1 und in V2 hat.
Merke: Ein Graph G ist genau dann bipartit, wenn er keinen Kreis ungerader Länge besitzt

Graphen Begriffe:
Wald: ungerichteter Graph ohne Kreise
Baum: ungerichteter, aber zusammenhängender Graph ohne Kreise

Gewurzelter Baum:
Gerichteter Graph mit Wurzel (Knoten mit Ein-Grad <= 1)
Knoten mit Aus-Grad = 0 heißen Blätter
Alle Knoten die keine Wurzel oder Blätter sind, heißen innere Knoten

Zusatz:
Jeder Knoten v, kann von Wurzel des Baumes aus erreicht werden
Knoten, die aus anderen Knoten hervorgehen heißen Kinder. Blätter haben keine Kinder, Wurzeln und innere Knoten schon.

Binärbaum: Jeder Knoten in gewurzeltem Baum hat Aus-Grad <= 2

Voller Binärbaum: Jeden Knoten in gewurzeltem Baum, der kein Blatt ist, hat Aus-Grad = 2

Vollständiger Binärbaum: Jeder Knoten in gewurzeltem Baum, der kein Blatt ist, hat Aus-Grad = 2 und Alle Blätter haben dieselbe Tiefe (Anmerkung: Wurzel hat höhe 0)

Kontraktion: „Verschmelzung“ von zwei Knoten a, b. Hierbei wird die Kante zwischen den beiden Knoten gelöscht und es entsteht ein neuer Knoten {a, b}, welcher alle restlichen Kanten von a und b besitzt.

Satz von Wagner: Kann aus Graph G durch Kontraktion ein Graph G´ erzeugt werden, der zu den Graphen $K_5$ oder $K_{3,3}$ isomorph ist, so ist G nicht planar.

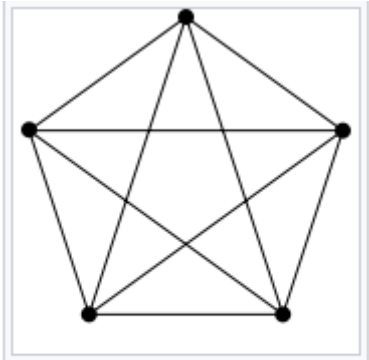
Abb. 1: Der Kuratowski-Graph $K_5$

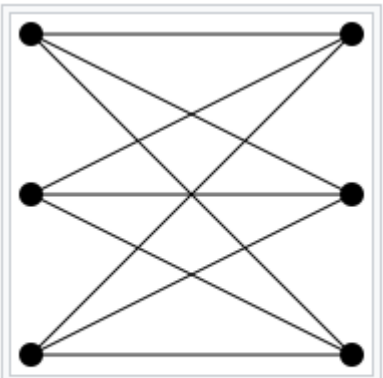
Abb. 2: Der Kuratowski-Graph $K_{3,3}$

**<u>Page Rank</u>**

Peer-Review: Gerichteter Graph, jeder Knoten gibt einen Teil seines Ranks an die Knoten ab, auf die er zeigt. Page Rank (PR) von Knoten j setzt sich zusammen aus:

$$PR_j = \sum_{i \in VOR_{WEB}(j)} \frac{PR_i}{a_i} \qquad \text{mit } PR_j >= 0$$

$PR_i$ ist der PR von Knoten i und $a_i$ ist der Aus-Grad von i (die Zahl der Kanten, die von i ausgehen; 3 ausgehende Kanten → i gibt 1/3 seines PR an jeden Knoten auf den er Zeigt)

Dämpfungsfaktor: Wahrscheinlichkeit von einem Knoten des Graphen in einen beliebigen anderen Knoten zu springen. Wird verwendet um eine Ballung von PR zu vermeiden, wenn einzelne Knoten eines Graphen PR nicht an Gesamtgraphen zurückgeben (PR „fließt“ von einem Teil des Graphen in einen anderen, aber nicht zurück)

PR berechnen: PR wird mithilfe des Dämpfungsfaktors d, den Eigenschaften von Graph G und der Gleichung (siehe unten) berechnet:

$$PR_j = \frac{1-d}{n} + d * \sum_{i \, \epsilon VOR_{WEB}(j)} \frac{PR_i}{a_i}$$

Hierbei ist n Gesamtzahl der Knoten in G

Um den PR für jeden Knoten zu bestimmen, wird die PR Gleichung für jeden Knoten aufgestellt und nach (1-d)/n aufgelöst. Dann kann das Gauß-Verfahren angewendet werden, um die einzelnen PR zu bestimmen.

Page Rank Matrix: Erstelle eine Matrix für den WEB Graphen G (ein gerichteter Graph) mittels der Gleichung:

$$p_{i,j} := \begin{cases} \frac{1-d}{n} + \frac{d}{a_i} & \text{falls } (i,j) \in E, \\ \frac{1-d}{n} & \text{falls } (i,j) \notin E \end{cases}$$

i und j sind die Knoten im Graphen, bzw. die Zeilen (i) und Spalten (j) der Matrix. Der untere Fall mit (i, j) kein Element von E tritt auf, wenn von i aus keine Kante nach j verläuft. $a_i$ ist die Gesamtzahl der Aus-Grad von i. Merke: $d/a_i = d * 1/a_i$

Die Gleichung berechnet die Übergangswahrscheinlichkeit von i nach j (Ergebnis: Übergangsmatrix [alle Spalten einer Zeile addiert ergeben immer 1] ).

Markov-Ketten: W := (G, M) mit Graph G und (Übergangs)Matrix M
Markov-Ketten modellieren die Aufenthaltswahrscheinlichkeiten von Graphen ausgehend von Startzustand π nach k Schritten.
Der Zustand der Kette nach k schritten ist $X_k(\pi)$. Dieser Zustand ist Abhängig von dem Zustand X bei k-1 Schritten ( $X_{k-1}(\pi)$ ).

Wahrscheinlichkeiten $\pi^k$ sich nach k Schritten in den jeweiligen Koten des Graphen G zu befinden ist gegeben durch:

$\pi^k = \pi^{(k-1)} * P = \pi^{(k-2)} * P^2 = \ldots = \pi^{(0)} * P^k$

→ $P^k$ multipliziert mit Anfangszustand $\pi^{(0)}$ ergibt Wahrscheinlichkeiten sich nach k Schritten in jeweiligen Knoten des Graphen zu befinden

Zusatz:
Sind in einem Graphen alle Übergangswahrscheinlichkeiten bekannt, so können diese einfach in eine Matrix M geschrieben werden (Zeile i und Spalte j stellen wieder die Kante {i, j} dar).

Irreduzibel: Ein Graph ist irreduzibel, wenn er stark zusammenhängend ist (siehe oben)

Periode: Ein Zustand (Knoten?) i hat die Periode p, wenn die Länge aller Wege von i nach i (von i über beliebige Wege zu Knoten x und zurück) durch p teilbar ist. Die Periode p ist hierbei Größtmöglich (Größter gemeinsamer Teiler aller Weglängen)
Bsp: Menge aller Weglängen = {2, 4, 6, 8, 12} → Periode 2
Menge aller Weglängen = {2, 3, 4} → Periode 1

Aperiodisch: Ein Graph G heißt aperiodisch, wenn kein Zustand (Knoten?) eine Periode p > 1 besitzt

Ergodisch: Ein Graph G heißt ergodisch, wenn er Irreduzibel und aperiodisch ist

Grenzverteilung: Die Grenzverteilung einer Markov-Kette existiert, wenn nach undendlich vielen Schritten eine Verteilung von Zustandswahrscheinlichkeiten existiert, die unabhängig vom Anfangszustand ist (→ $\lim_{k \to \text{unendlich}} P^k$ muss existieren und alle Zeilen müssen identisch sein)

$$\lim_{k \to \infty} \pi \cdot P^k = \mathcal{G}(\mathcal{M})$$

Stationäre Verteilung:
Eine Verteilung π (Anfangsposition: Knoten v in Graph G) ist stationär, wenn für die Verteilung π in einer Markov-Kette M(G, P) gilt:
π = π * P
(→ Die Verteilung π ändert sich nicht, nachdem…?)

Ist eine Markov-Kette ergodisch → es gibt genau eine stationäre Verteilung σ, die mit der Grenzverteilung übereinstimmt
→ G(M) = σ (G(M) ist Grenzverteilung der Markov-Kette)

Gamblers Ruin Problem:
Wahrscheinlichkeit $s_K$ dass Spieler die Bank „sprengt“ (Eigenes Geld + Kapital der Bank erspielt), wenn Chance pro Runde zu gewinnen p ≠ 1/2

$$s_K = \frac{\left(\frac{q}{p}\right)^K - 1}{\left(\frac{q}{p}\right)^M - 1}$$

p = Wahrscheinlichkeit pro Spielrunde zu gewinnen, q = Wahrscheinlichkeit zu verlieren (q = 1 – p), K ist das Startkapital des Spielers, M ist das Startkapital K + Kapital der Bank N (M = K + N) [Evt. auch einfach die Zielsumme, die beim Glücksspielt erreicht werden soll]

Ist p = 1/2, dann gilt: $s_k$ = K/M

**Automaten**

DFA (deterministische, endliche Automaten [determined finite automatons])
Nomenklatur:
DFA, A = (Σ, Q, δ, $q_0$, F)
Σ = Menge der Buchstaben, die in den Automaten eingelesen werden können (Eingabealphabet)
Bsp: {a, b, c}, {0, 1},… etc.
Q = Menge aller vorhandenen Zustände (Knoten) in dem Automaten vorhanden sind (Zustandsmenge)
δ = Übergansfunktion/ Überführungsfunktion mit Q x Σ
$q_0$ = Element von Q. Zustand in dem sich der Automat befindet, bevor Wörter eingelesen werden (Startzustand)
F = Teilmenge der Zustände Q, gleichzeitig Menge aller akzeptierenden Zustände (**F**inal states; Endzustände, akzeptierende Zustände)

Akzeptierte Sprache L(A)
L = Sprache
A = Automat
L(x) akzeptierte Sprache von Automat A
Automaten A und B sind äquivalent, wenn beide Automaten dieselbe Sprache L akzeptieren
→ L(A) = L(B)

Erweiterte Überführungsfunktion δ mit Dach (δ^)
δ^(q, w) (das Dach muss über das δ, #fuckwordevenmore) mit q Element von Zustandsmenge Q und w Element von Σ* (Menge aller möglichen Wörter) [Anmerkung: ε ist das „leere Wort“, hier wird kein Buchstabe eingelesen]
Für die erweiterte Übergangsfunktion gilt:

$$\text{F.a. } q \in Q \text{ ist } \widehat{\delta}(q, \varepsilon) := q.$$

$$\text{F.a. } q \in Q,\ w \in \Sigma^* \text{ und } a \in \Sigma \text{ gilt für } q' := \widehat{\delta}(q, w):$$

$$\widehat{\delta}(q, wa) := \delta(q', a)$$

Äquivalenzrelationen: gerrichtete Graphen lassen sich als (2-stellige) Relation R über der Knotenmenge V ausdrücken. Möglich, da die Menge an Kanten E eine Teilmenge aller möglichen Kanten $V^2$ ist.

Äquivalenzrelationen (Eigentschaften):

Reflexiv: Für jeden Knoten v gilt:

$$(v, v) \in E$$

(jeder Knoten v hat eine Kante, die auf ihn selbst zeigt)

Symmetrisch: Für alle Knoten v, w gilt:

$$\text{Wenn } (v, w) \in E, \text{ dann auch } (w, v) \in E.$$

(wenn eine Kante von v nach w existiert, existiert auch eine Kante von w nach v)

Transitiv: Für alle Knoten v, w, u gilt:

$$\text{Ist } (v,w) \in E \text{ und } (w,u) \in E, \text{ so auch } (v,u) \in E$$

(wenn es eine Kante von v nach w und eine Kante von w nach u gibt, dann gibt es auch eine Kante von v nach u)

Verschmelzungsrelation:
Verhalten sich zwei Zustände v, w des Automaten A exakt gleich (dieselben Übergänge führen in dieselben Äquivalenzklassen und die Zustände sind entweder alle verwerfend oder akzeptierend), so heißen sie äquivalent $v \equiv_A w$

$$\text{f.a. Worte } w \in \Sigma^* : \widehat{\delta}(p,w) \in F \iff \widehat{\delta}(q,w) \in F$$

Äquivalenzklassen:
Wenn der Automat A ein DFA ist, in dem Verschmelzungsrelationen vorkommen, dann gilt:
Eine Verschmelzungsrelation $\equiv_A$ ist eine Äquivalenzrelation. Zustände in Äquivalenzrelationen können in Äquivalenzklasse Zusammengefasst werden. Es wird ein „Vertreter" aus der Klasse gewählt, der diese „repräsentiert"
Bsp: $1 \equiv_A 3 \equiv_A 4 \equiv_A 7$ = Äquivalenzklasse {1, 3, 4, 7} = $[3]_E$ (E = Äquivalenzrelation)

Der Entstehende Automat A´ heißt Äquivalenzklassenautomat und die Anzahl seiner Zustände ist gleich der Anzahl seiner Äquivalenzklassen

Zusatz:
Zwei unterschiedliche Äquivalenzklassen $[v]_E$ , $[w]_E$ stimmen entweder überein ($[v]_E = [w]_E$) oder sind disjunkt ($[v]_E \cap [w]_E = \emptyset$)

Index(A): Der Index des Automaten A, auch ist die Menge an unterschiedlichen Äquivalenzklassen in dem gegebenen Automaten A

Zeugen:
Um alle äquivalenten Zustände in einem Automaten zu bestimmen, werden alle inäquivalenten Zustände über Zeugen identifiziert.
Bsp δ(1, a) ≡ akzeptierender Zustand δ(2, a) ≡ verwerfender Zustand

➔ a ist ein Zeuge für die inäquivalenz von 1 und 2

Minimierung von DFA (Skript Beispiel 7.30)

Nerode-Relation:
Rückt Bewertung Äquivalenzen ab von Struktur des Automaten A und hin zu Struktur von Sprache L ($\equiv_A$ beschreibt Äquivalenz von Zuständen in Automaten, $\equiv_L$ beschreibt Äquivalenz von Worten)
Genau wie Verschmelzungsrelationen, sind auch Nerode-Relationen Äquivalenzrelationen
→ sie können zu Äquivalenzklassen zusammengefasst werden (für wörter v,w gilt: $v \equiv_L w$)
→ $[v]_L = \{v, w\}$

Nerode Automat:
Aus den Äquivalenzklassen der Nerode-Relation kann ein sog. Nerode-Automat $N_L$ erzeugt werden. Der Nerode-Automat akzeptiert die Sprache L
$N_L = (\Sigma, Q, \delta, q_0, F)$

Anmerkung:

$$u \equiv_L w \implies ua \equiv_L wa$$

(sind die Worte u und v Äquivalent, dann gilt mit Wort a: ua $\equiv_L$ wa)

Satz von Myhill-Nerode I:
- Index(L(A)) stimmt mit der minimalen Zustandszahl eines DFA für die Sprache L(A) überein
- Der Äquivalenzklassenautomat A´ ist minimal

Satz von Myhill-Nerode II:
Wenn L eine Teilmenge aller möglichen Worte des Alphabets $\Sigma^*$ ist, dann ist L genau dann regulär, wenn der Index von L endliche ist

$$L \text{ ist regulär} \iff \text{Index}(L) \text{ ist endlich}$$

Reguläre Sprache:
Eine Sprache L heißt regulär, wenn ein DFA A existiert, der diese Sprache akzeptiert
(L = L(A))

Eine Sprache L ist nicht regulär, wenn der minimale Automat $N_L$ unendlich viele zustände hat.

NFA (nicht deterministische, endliche Automaten [nondeterministic finate automatons]):
Nomenklatur:
NFA, $A = (\Sigma, Q, \delta, q_0, F)$
Grundsätzlich analog zu DFA, aber:
$\delta$ = Delta ist hier die Übergangsfunktion, die jedem Zustand q und jedem Symbol a (Element von $\Sigma$) eine Menge von Möglichen Nachfolgezuständen zuordnet. Da Automat nicht deterministisch, können bei gleichere Eingabe unterschiedliche Ergebnisse herauskommen
→ man Arbeitet mit Mengen von möglichen Folgezuständen
Anmerkung: Ist ein Ergebnis von $\delta$ die leere Menge (das Übergangssymbol existiert für den gegebenen Zustand nicht), dann „stürzt“ der Automat ab → das Wort wird verworfen

Erweiterte Übergangsfunktion von NFAs ist hieran angepasst, sowie die Definition von akzeptierten Sprachen

$$L(A) = \{ w \in \Sigma^* : \widehat{\delta}(q_0, w) \cap F \neq \emptyset \}$$

→ Eine Sprache ist akzeptiert, wenn die Menge von möglichen Nachfolgezuständen (irgend)einen akzeptierten Zustand enthält
(wenn es eine Möglichkeit gibt, dass das Wort w akzeptiert wird, gilt es als akzeptiert)

Zusatz:
Für jeden NFA A, existiert auch ein DFA A´, der dieselbe Sprache akzeptiert wie der NFA (L(A) = L(A´))
[der DFA besitzt als Zustände, alle möglichen Mengen an Zuständen, die in dem entsprechenden NFA auftreten können]

**Reguläre Ausdrücke**