# Zusammenfassung

Im Automatenteil der FGI1 Vorlesung haben wir uns mit der Charakterisierung von Sprachfamilien durch immer mächtigere Automatenmodelle und Grammatiken beschäftigt. Folgende Familien haben wir kennengelernt:

1. Die Familie der *regulären Sprachen* $REG$. Diese wird durch deterministische endliche Automaten (DFAs), nichtdeterministische endliche Automaten (NFAs), rechtslineare (oder Typ-3) Grammatiken und reguläre Ausdrücke erfasst, d.h. zu jeder regulären Sprache $L$ gibt es bspw. einen DFA $A$ mit $L(A) = L$. Ebenso gibt es eine Typ-3 Grammatik $G$ mit $L(G) = L$ usw.

2. Die Familie der *kontextfreien Sprachen* $CF$. Diese wird durch nichtdeterministische Kellerautomaten (PDAs) und kontextfreie (oder Typ-2) Grammatiken (CFGs) erfasst. Die PDAs akzeptieren dabei mit leerem Keller. Alternativ ist auch eine Akzeptanz mit Endzustand möglich.

3. Die Familie der *kontextsensitiven Sprachen* $CS$. Diese wurde in der Vorlesung nur kurz erwähnt. Sie wird erfasst von sogenannten monotonen oder kontextsensitiven (oder Typ-1) Grammatiken oder von nichtdeterministischen linear beschränkten Automaten (LBAs).

4. Die Familie der *entscheidbaren Sprachen* $Rec$. Diese wird von Turing-Maschinen erfasst, die bei jeder Eingabe anhalten, d.h. zu einem $L \in Rec$ gibt es eine TM $M$ mit $L(M) = L$ und $M$ hält auf jeder Eingabe an.

5. Die Familie der *aufzählbaren Sprachen* $Re$. Diese wird von nichtdeterministischen Turing-Maschinen (NTMs), deterministischen Turing-Maschinen (DTMs) und Typ-0 Grammatiken erfasst (wobei Typ-0 Grammatiken in der Vorlesung nur kurz erwähnt wurden).

6. Die Familie der *abzählbaren Mengen*. Diese ist dadurch charakterisiert, dass jede abzählbare Menge $M$ endlich ist oder eine Bijektion von den natürlichen Zahlen auf $M$ existiert. Die Familie der abzählbaren Mengen hat hier eigentlich keine große Bedeutung und ist primär als Grenze von oben bzgl. der aufzählbaren Sprachen von Interesse.

Man bezeichnet die regulären, kontextfreien, kontextsensitiven und aufzählbaren Sprachen analog zu den zugehörigen Grammatiken auch als Typ-3, Typ-2, Typ-1 und Typ-0 Sprachen. Zusammen bilden sie die Stufen der sogenannten *Chomsky-Hierarchie*.

Die oben aufgelisteten Sprachfamilien werden von oben nach unten *echt mächtiger*, d.h. die Familie der regulären Sprachen ist eine echte Teilfamilie der Familie der kontextfreien Sprachen, diese wiederum eine echte Teilfamilie der Familie der kontextsensitiven Sprachen usw. Sprachen, die nicht mehr in der Familie der entscheidbaren Sprachen enthalten sind, werden als *unentscheidbar* bezeichnet. Diese spielen eine wichtige Rolle in der Informatik, da ein Nachweis der Unentscheidbarkeit eines Problems, bedeutet, dass dieses nicht von einem Computer gelöst werden kann. (Üblicherweise geht es um Sprachen. Ein Problem kann i.A. aber einfach in eine Sprache überführt werden. Will man z.B. zwei Zahlen addieren, dann kann als zugehörige Sprache bpsw. die Menge der Dreitupel $(x, y, z)$ mit $z = x + y$ gewählt werden.)

Die einzelnen Modelle der obigen Stufen sind zudem äquivalent, d.h. das bspw. DFAs, NFAs, Typ-3 Grammatiken und reguläre Ausdrücke zueinander äquivalent sind. Gegeben ein NFA $A$ kann also ein DFA $B$ und eine Typ-3 Grammatik $G$ angegeben werden mit $L(A) = L(B) = L(G)$. Ferner existiert ein regulärere Ausdruck, der die gleiche Sprache beschreibt.

Wir wollen noch kurz auf einige Besonderheiten in der obigen Klassifizierung eingehen:

- Die Sprachfamilie $CF$ der kontextfreien Sprachen haben wir in der Vorlesung durch PDAs, die mit leerem Keller akzeptieren, oder durch kontextfreie Grammatiken erfasst. Ein PDA akzeptiert dabei mit leerem Keller, wenn er ein vorgelegtes Eingabewort $w$ bis zum Ende lesen kann und dann der Keller komplett geleert ist. Insb. muss auch das zu Anfang im Keller befindliche Kellerbodenzeichen $\perp$ vom Keller entfernt worden sein.

- Es ist auch möglich PDAs zu definieren, die mit Endzustand akzeptieren. Ein Wort wird dann akzeptiert, wenn es bis zum Ende gelesen werden kann und der PDA dann in einem Endzustand ist. Wie der Kellerinhalt zu diesem Zeitpunkt aussieht, ist irrelevant. PDAs, die mit leerem Keller akzeptieren, und PDAs, die mit Endzustand akzeptieren, sind äquivalent, d.h. sie erfassen die gleiche Sprachfamilie. Zu jedem PDA, der mit leerem Keller eine Sprache $L$ akzeptiert, kann also ein PDA konstruiert werden, der mit Endzustand die gleiche Sprache $L$ akzeptiert, und umgekehrt.

- Beide oben aufgeführten PDA-Modelle sind nichtdeterministisch! Es ist auch möglich, einen deterministischen Kellerautomaten einen sogenannten DPDA einzuführen. Die DPDAs, die mit Endzustand akzeptieren, erfassen die Sprachfamilie $detCF$. Diese ist eine echte Oberfamilie der Familie der regulären Sprachen und eine echte Teilfamilie der Familie der kontextfreien Sprachen. Lässt man DPDAs mit leerem Keller akzeptieren, so erhält man eine Sprachfamilie, die "quer" zu $REG$ und $CF$ liegt, d.h. sie enthält einige (aber nicht alle) regulären Sprachen und ebenso einige (aber nicht alle) kontextfreien Sprachen. (Dies aber nur zur Information. Da wir dies in der Vorlesung nur kurz erwähnt haben, ist dies nicht für die Klausur relevant.)

- Die Sprachfamilie $CS$ der kontextsensitiven Sprachen lässt sich durch LBAs und Typ-1 Grammatiken erfassen. Ein LBA ist dabei eine NTM

mit der zustätzlichen Eigenschaft, das bei einer Eingabe der Länge $n$ nur $c \cdot n$ (wobei $c$ eine festgelegte Konstante ist) viele Bandzellen benutzt werden. Eine Typ-1 Grammatik ist eine Grammatik bei der alle Produktionen von der Form $u \to v$ sind, wobei entweder $u = w_1 A w_2$ und $v = w_1 x w_2$ mit $A \in V_N$, $x \in V^+$ und $w_1, w_2 \in V^*$ ist oder $u = S$ und $v = \lambda$ gilt. Zusätzlich kommt $S$ in keiner Regel auf der rechten Seite vor. (Der Begriff monoton kommt daher, dass die rechten Seiten der Regeln stets mindestens so groß wie die linken Seiten sind mit Ausnahme der Regel $S \to \lambda$.) (Auch dies ist nur zur Information und nicht klausurrelevant.)

- Neben den eben erwähnten nichtdeterministischen LBAs lassen sich auch deterministische LBAs (DLBAs) einführen. Es ist aber unklar, ob diese äquivalent zu NLBAs sind.

Ein weiteres wichtiges Thema war die Komplexitätstheorie. Wenn man von einer Sprache bzw. einem Problem bereits weiß, dass es entscheidbar ist, so ist anschließend von Interesse, wie schnell dieses Problem gelöst werden kann. Eine herausragende Rolle spielen dabei die Klassen $P$ und $NP$. Existiert für ein Problem eine Turing-Maschine bzw. allgemein ein Algorithmus, der das Problem in Polynomialzeit löst, so liegt das Problem in $P$. Ein Problem wird dabei in Polynomialzeit gelöst, wenn es ein Polynom $p$ gibt, so dass bei einer Eingabe der Länge $n$ maximal $p(n)$ viele Schritte (von einer DTM) getätigt werden. $NP$ ist eine ähnliche Klasse, allerdings wird hier eine *nichtdeterministische* TM als Modell verwendet. In $NP$ liegen also all jene Probleme, die von einer nichtdeterministischen Turing-Maschine (bzw. einem nichtdeterministischem Algorithmus) in Polynomialzeit gelöst werden können. Da jede DTM als spezielle NTM gesehen werden kann, gilt sofort $P \subseteq NP$. Ob auch die Umkehrung gilt, also ob $P = NP$ gilt, ist unbekannt.

Von Problemen, die in $P$ liegen, sagt man im Allgemeinen, dass sie praktisch lösbar sind. Liegt ein Problem hingegen in $NP \setminus P$, so ist dies nicht der Fall, da eine Lösung dann zu zeitintensiv ist (ein Problem aus $NP$ kann deterministisch in Exponentialzeit gelöst werden). Bisher ist es aber nicht gelungen Techniken zu entwickeln, mit denen gezeigt werden kann, dass ein Problem in $NP$, aber nicht in $P$ liegt. Mit einer solchen Technik könnte man *untere Schranken* für den Zeitbedarf von Problemen etablieren. Da wir dies aber noch nicht können, behelfen wir uns mit einer anderen Technik: Wir weisen ein Problem als $NP$-vollständig nach. Ein $NP$-vollständiges Problem liegt in $NP$ und ferner kann jedes Problem aus $NP$ in Polynomialzeit auf es *reduziert* werden. Eine Reduktion von einer Sprache $L_1$ auf eine Sprache $L_2$ ist dabei eine Funktion $f$, die in Polynomialzeit berechnet werden kann und die $x \in L_1$ genau dann, wenn $f(x) \in L_2$ erfüllt. Von einem $NP$-vollständigen Problem sagt man dann, dass es eines der schwierigsten in $NP$ ist und vermutet ferner, dass es für dieses keinen Algorithmus in $P$ gibt. Der Grund hierfür liegt darin, dass sollte man es doch in $P$ lösen können, daraus $P = NP$ folgen würde. Obgleich wir also noch keine unteren Schranken für die Laufzeit von Algorithmen zeigen können, können wir immerhin zeigen, dass ein Problem recht wahrscheinlich nicht in Polynomialzeit gelöst werden kann.

## Wichtige Begriffe und Verfahren

In der Vorlesung haben wir eine Vielzahl an Begriffen und Verfahren (bzw. Vorgehensweisen) kennengelernt. Es folgt eine (recht umfangreiche) Liste der wichtigsten. Der zwölfte Foliensatz gibt ebenfalls einen schnelle(re)n Überblick über die wichtigen Begriffe und Verfahren und die Probeklausur gibt einen guten Eindruck von zu erwartenden Aufgaben.

- Mathematische Grundbegriffe wie Mengen, Funktionen, Relationen usw.
- Grundbegriffe der formalen Sprachen wie Alphabet, Konkatenation, das leere Wort $\lambda$ (oder $\epsilon$), (formale) Sprache. Die Notationen $|w|$ für die Länge eines Wortes $w$ und $|w|_a$ für die Anzahl des Auftretens des Buchstaben $a$ in dem Wort $w$. Ferner die Notation $w^{rev}$ für das "Umdrehen" des Wortes, d.h. es ist bspw. $011^{rev} = 110$.
- Deterministischer, endlicher Automat (DFA) inkl. der einzelnen "Bestandteile", hier also Zustandsmenge, Eingabealphabet, Überführungsfunktion, Startzustand und Menge der Endzustände. Sowie der aufbauenden Begriffe erweiterte Überführungsfunktion, Konfiguration, Konfigurationsübergang, Rechnung, Erfolgsrechnung, akzeptierte Sprache, vollständig und initial zusammenhängend.
- Wichtige Verfahren: Gegeben ein DFA, die von ihm akzeptierte Sprache bestimmen und gegeben eine Sprache, einen DFA konstruieren, der sie akzeptiert. In beiden Fällen ist anschließend ein Beweis nötig, dass tatsächlich $L(A) = L$ gilt. Dabei ist i.A. $L(A) \subseteq L$ und $L \subseteq L(A)$ zu zeigen, was sauber zu trennen ist. (Zur Konstruktion eines DFA haben wir zwei Techniken kennengelernt.)
- Nichtdeterministischer, endlicher Automat (NFA) mit teilweiser Abwandlung obiger Begriffe. Besonders wichtig ist hier die Änderung der Akzeptanzbedingung im Vergleich zum DFA.
- Wichtiges Verfahren: Potenzautomatenkonstruktion (hier insb. auch beachten wie der Start- und wie die Endzustände bestimmt werden und dass der Potenzautomat durch die Einführung einer Senke (der Zustand $\emptyset$) stets vollständig ist.)
- Weitere Formalismen: NFAs mit $\lambda$-Kanten und reguläre Ausdrücke.
- DFAs, NFAs, NFAs mit $\lambda$-Kanten und reguläre Ausdrücke sind äquivalent und erfassen genau die Familie der regulären Sprachen.
- Abschlusseigenschaften. Eine Abschlusseigenschaft einer Sprachfamilie $\mathcal{C}$ ist eine Operation $\circ$, so dass für je zwei $L_1, L_2 \in \mathcal{C}$ wieder $L_1 \circ L_2 \in \mathcal{C}$ gilt (und ähnlich für einstellige Operationen wie z.B. die Komplementbildung).
- Die regulären Sprachen sind gegenüber $\cup, \cdot, +, *, \cap$ sowie Komplementbildung abgeschlossen (und sogar noch ggü. weiteren Operationen).

- Pumping Lemma für reguläre Sprachen (Aussage? Sinn? Wie wendet man es an?).
- Nichtdeterministischer Kellerautomat (PDA) wieder inkl. der nötigen Begriffe. Hier kommt insb. die Handhabung des Kellers hinzu. Die mit leerem Keller akzeptierte Sprache wird mit $L_\lambda(A)$ oder $N(A)$, die mit Endzustand akzeptierte Sprache mit $L_Z(A)$ oder $L(A)$ bezeichnet (meist ist dies aber im Kontext klar).
- Wichtige Verfahren: Wie oben auch hier gegeben eine Sprache einen PDA konstruieren können (hier haben wir wieder zwei Konstruktionsmethoden kennengelernt) und andersherum gegeben einen PDA, die Sprache bestimmen können (seltener). Hinzu kommt auch hier der Beweis von $L_\lambda(A) = L$ etc.
- Grammatiken, insb. kontextfreie Grammatiken und (rechts-)lineare Grammatiken sowie die zugehörigen Begriffe wie Nonterminale, Terminale, Produktionen, Startsymbol, Ableitung, generierte Sprache.
- Die rechtslinearen Grammatiken generieren genau die regulären Sprachen (und sind damit wieder äquivalent zu DFAs, NFAs usw.). Die kontextfreien Grammatiken generieren genau die Sprachfamilie der kontextfreien Sprachen. Sie sind äquivalent zu PDAs.
- Konstruktionsmethoden zu einer Sprache eine Grammatik zu konstruieren, die die Sprache generiert und andersherum zu einer Grammatik die Sprache zu bestimmen, die sie generiert. Bei beiden ist wieder ein Beweis von $L(G) = L$ nötig, bei dem wieder zwei Richtungen zu zeigen sind.
- Abschlusseigenshaften. Die kontextfreien Sprachen sind ggü. $\cup, \cdot$ und $*$ abgeschlossen, nicht aber ggü. $\cap$, Komplementbildung und Mengendifferenz.
- Wichtiges Verfahren: Herstellung der (erweiterten) Chomsky-Normalform.
- Pumping Lemma kontextfreier Sprachen (Aussage? Sinn? Wie wendet man es an? Unterschied zum Pumping Lemma für reguläre Sprachen?).
- Deterministische Turing-Maschine inkl. der zugehörigen Begriffe (LSK, Bandalphabet, akzeptierte Sprache usw.).
- Turing-Maschinen können Sprachen akzeptieren und Funktionen berechnen.
- Wichtige Verfahren: Zu einer Sprache eine TM konstruieren, die sie akzeptiert, bzw. zu einer Funktion eine TM konstruieren, die sie berechnet. (Die Rückrichtung geht wie bei DFAs und PDAs auch, ist allerdings eher unüblich, da das eher daran erinnert unkommentierten Sourcecode zu verstehen.) Auch hier muss argumentiert werden, warum die TM das gewünschte tut. Oft gibt man zudem nicht mehr das Zustandsübergangsdiagramm einer TM an, sondern beschreibt nur noch, was die TM tut.

- Alternative TM-Modelle wie bspw. die TM mit beidseitig unendlichem Band und die $k$-Band off-line TM. Konstruiert man eine TM, sollte man immer angeben, welches Modell man benutzt!
- Nichtdeterministische Turing-Maschinen (NTM) wieder inkl. der zugehörigen Begriffe.
- "Programmieren" einer NTM unter Ausnutzung des Nichtdeterminismus. In den Folien findet man z.B. eine NTM für das Cliquenproblem (Foliensatz 8).
- Äquivalenz von NTM und DTM.
- Aufzählbar bzw. rekursiv aufzählbar und die Sprachfamilie $RE$.
- Entscheidbar und die Sprachfamilie $REC$.
- Zusammenhang zwischen entscheidbar und aufzählbar. ($L$ ist entscheidbar genau dann, wenn $L$ und $\overline{L}$ aufzählbar sind.)
- Abschlusseigenschaften der aufzählbaren und der entscheidbaren Sprachen. (Sind sie z.B. ggü. $\cap$ und $\cup$ abgeschlossen? Warum (nicht)?)
- Unentscheidbare Sprachen (insb. das Halteproblem) und wie man neue Probleme als unentscheidbar nachweist (Stichwort: Reduktion).
- Nicht mehr aufzählbare Sprachen (die Sprache $L_d$).
- Zeit- und Platzkomplexität. $t(n)$-zeit- und $s(n)$-platzbeschränkt, worst-case-Komplexität. Zeit- und Platzschranken für NTMs.
- Die Komplexitätsklassen $P, NP, PSpace$ und $NPSpace$.
- $O$-Notation.
- Probleme in $P$, Probleme in $NP$ und wie man dies zeigt.
- (Polynomialzeit-)Reduktion.
- $L_1 \leq_p L_2$ als Notation für die Reduktion in Polynomialzeit.
- $NP$-Vollständigkeit.
- Transitivität von $\leq_p$.
- Folgerung dass $P = NP$ gilt, sollte ein $NP$-vollständiges Problem in $P$ liegen.
- Verfahren, um ein neues Problem als NP-vollständig nachzuweisen.
- Die Klasse der $NP$-vollständigen Probleme wird mit $NPC$, die der $NP$-schwierigen Probleme mit $NPH$ bezeichnet.