Theoretische Informatik

Zusammenfassung

Aless Lasaruk

29. Dezember 2005

Zusammenfassung

Dieses Dokument enthält Fragen und Antworten zum Fach Theoretische Informatik. Die Fragen sind durch systematisches Durchgehen der Folien der Vorlesung entstanden, decken aber den Stoff nicht vollständig. Man beachte auch, dass die Antworten nicht exakt formuliert sind. Das Ziel der Formulierung ist das Minimieren der schriftlichen Antwort und Maximieren der mündlichen.

Inhaltsverzeichnis

Einleitung

- **Wie sieht die *Chomsky Hierarchie* aus?**
  REG (Regulär) $\subset$ CFL(Kontextfrei) $\subset$ CSL (Kontextsensitiv) $\subset$ RE (Rekursiv aufzählbar).
- **Wie sieht eine übliche Komplexitätshierarchie aus?**
  L (Deterministischer logarithmischer Platz) $\subseteq$ NL (Nicht deterministischer logarithmischer Platz) $\subseteq$ P (Polynomielle deterministische Zeit) $\subseteq$ NP (Nichtdeterministische polynomielle Zeit) $\subseteq$ PSPACE (Polynomieller deterministischer Platz) $\subseteq$ EXPTIME (Exponentielle deterministische Zeit) ...
- **Welche operationellen *Berechnungsmodelle* kennen Sie?**
  Automaten (endlich, keller, Turingmaschine) und Registermaschinen (RAM).
- **Welche Berechnungtypen kennen Sie?**
  Determinismus, Nicht-Determinismus, Alternierung, Parallelität, PRAMS, Schaltkreise.
- **Welche alternativen Berechnungsmodelle kennen Sie?**
  μ-rekursive und partiell rekursive Funktionen, formale Kalküle (Markov Kalkül, λ-Kalkül) und Ersetzungsysteme.
- **Was besagen die Thesen von Church, Turing, Post, Markov, Gödel usw.?**
  Alle obigen Berechnungsmodelle sind universell (weil beweisbar äquivalent und wechselseitig simulierbar) und beschreiben den Algorithmusbegriff oder *Berechenbarkeit*.
- **Was versteht man unter *Komplexität*?**
  Laufzeit und Speicherbedarf (allgemeiner Aufwandsklassen) von Problemen und Algorithmen in einem gegebenen Berechnungsmodell.
- **Wie kann man NP-harte Probleme bewerten?**
  Ineffizient.
- **Wovon ist die Laufzeit bzw. Speicherbedarf von Problemen stark abhängig?**
  Vom Maschinenmodell.
- **Was ist ein Problem aus der Sicht der Informatik?**
  Eine *formale Sprache*. Worte in L sind Beschreibungen für zulässige Probleminstanzen.
- **Nennen Sie einige Problemspezifikationen?**
  Prim in Binär-(Unär)darstellung, Hamilton als Adjazenzliste eines Graphen, $a^n b^n$ Wörter.
- **Welche Konventionen verfolgt man hinsichtlich der Problemformulierung?**
  Umformung auf ein ja/nein Problem, Repräsentation durch ein Alphabet (Zahlen binär, Graphen durch *Adjazenzlisten*), n ist meist die Größe der Probleminstanz formuliert durch die Länge der Eingabe.
- **Welche Techniken verwendet man um Komplexität von Problemen nachzuweisen?**
  Effiziente Simulation unter Zeit bzw. Speicherbeschränkungen, Reduktionen, Diagonalisierung, Abzählargumente (z.B. *Pumping Lemma*), obere/untere Schranken.

- **Was versteht man unter *Klasse*?**
  Menge (von Mengen) oder Sprache (Menge von Worten). Formal ist eine Klasse mehr als eine Menge. Daher ist diese Betrachtung allgemein äußerst problemmatisch! Man kann vermutlich diese eingeschränkte Sicht betreiben, wenn man sich auf ein festes Alphabet beschränkt.
- **Was versteht man unter einer *Komplexitätsklasse*?**
  Eine Menge (eigentlich Klasse) von Problemen lösbar in einem bestimmten Berechungsmodell unter gleichem Aufwand (Funktion). Erfordert einen Vergleichsmaß für Aufwände.
- **Wie viele Komplexitätsklassen gibt es?**
  Soviele, wie Elemente im Kreuzprodukt der Berechnungsmodelle, Vergleichsmaße und Aufwände. Also überabzählbar viele. Man kann zu jeder Sprache, die von einer TM akzeptiert wird mindestens eine Komplexitätsklasse festlegen, indem man die TM mit der Klasse identifiziert.

1 Reguläre Mengen

1.1 Mengentheoretische Beschreibung

- **Nennen Sie einige Charakterisierungen regulärer Mengen!**
  Reguläre Grammatik ($A \to aB, A \to a$), Reguläre Ausdrücke ($a, a^*, a^+, ab, a|b$), endliche Automate, Rechtskongruenzen mit endlichem Index.
- **Nennen Sie einige nichtreguläre Sprachen!**
  Alles, wo man zählen muss, wenn man das Wort akzeptieren will. $\{a^n b^n \mid n \in \mathbb{N}\}$, $\{a^n b^m \mid n, m \in \mathbb{N}, n \neq m\}$, $\{wcw \mid w \in \sigma^*\}$, $\{wcw^R \mid w \in \sigma^*\}$.
- **Wann ist eine Sprache *regulär*?**
  Sie ist leer, aus einem Zeichen oder entsteht durch endliche Vereinigungen, Konkatenationen oder Sternbildungen regulärer Sprachen.
- **Welche einfachen allgemeinen Sprachen sind regulär?**
  Endliche Sprachen.
- **Welche zwei Beschreibungs-Prinzipen gibt es bei der Bildung von Sprachen?**
  Von Innen (Erzeugungspronzip, Grammatik) und von Außen (Akzeptanzprinzip, Automat).
- **Wann ist eine Sprache R^* unendlich?**
  Genau dann, wenn R nichtleer ist und nicht aus dem leeren Wort besteht.

1.2 Formale Beschreibung in Metasprache

- **Wie beschreibt man reguläre Sprachen?**
  Meist durch reguläre Ausdrücke (*REX*), analog zur Definition.
- **Welche Erweiterungen der kennen Sie zu REX?**
  EXREX, indem man Symbole für Schnitt, Komplement, Potenzierung usw. hinzunimmt.
- **Zu welchem Sprachtyp gehört REX?**
  Kontextfrei.
- **Was kann man über die Entscheidbarkeit des Äquivalenzproblems für EXREX sagen?**
  Ist Entscheidbar. Komplexität hängt sehr unterschiedlich von der Beschreibungsform (Art der Terme) ab.
- **Was kann man über ein Kalkül für REX aussagen?**
  Ist korrekt und vollständig.
- **Auf welche Arten kann man Äquivalenz regulärer Ausdrücke beweisen?**
  Syntaktisch und semantisch.

- **Welches Problem existiert bezüglich der Normalform für reguläre Ausdrücke?**
  Es gibt keine eindeutigen Normalform (nur bei DFA).
- **Wann heißt eine Äquivalenzrelation eine *Rechts-Kongruenzrelation*?**
  Wenn aus $a \sim b$ folgt $ax \sim bx$.
- **Wie kann man die Eigenschaft *hat endlichen Index* beschreiben?**
  Es gibt eine endliche Menge von typischen Repräsentanten.
- **Was ist eine *L-induzierte Rechtskongruenz*?**
  $x \sim y$ gdw. für alle $z \in \Sigma^*$ gilt $xz \in L$ gdw. $yz \in L$.
- **Welche Repräsentanten kann man für die Sprache $L = \{a^n b^n \mid n \in \mathbb{N}\}$ angeben?**
  $a^n b^i$ mit $i \leq n$

1.3 Endliche Automaten

- **Welche Anwendungen von endlichen Automaten kennen Sie?**
  Scanner und Compiler (Tokenanalyse), Zustandsmaschinen (Fahrkartenautomat).
- **Wie kann man beim Automaten die akzeptierte Sprache beschreiben?**
  Alles Wörter, sodass nach dem Ende der Eingabe befindet sich der Automat im Endzustand. Falls keine Relation vorhanden, dann befindet sich der Automat in einem Fehlerzustand.
- **Welche Typen von endlichen Automaten kennen Sie?**
  Deterministische, Nichtdeterministische und Nichtdeterminitische mit λ-Übergängen.
- **Wie kann man die Zustandsübergangsfunktion auf Worte erweitern?**
  Man definiert $\delta^*(z, a) = \delta(z, a)$ und $\delta^*(z, aw) = \delta^*(\delta(z, a), w)$.
- **Was ist der Unterschied zwischen äquivalenten DFA und NFA bezüglich ihrer Größe?**
  DFA kann exponentiell viel Zustände beanspruchen.
- **Was ist ein *Mealy-Automat* und was ist ein *Moore-Automat*?**
  Beides Autoamten mit ausgaben. Beim Mealy ist die Ausgabe vom Zustand und Eingabezeichen abhängig und bei Moore nur vom Zustand. Beide Typen sind äquivalent.
- **Welcher Zusammenhang besteht zwischen gerichteten markierten Graphen und Automaten?**
  Man kann sie gegenseitig einander zuordnen.
- **Was sind bei Automaten interessante Fragen?**
  Längste/Kürzeste Wege, Existenz eines Weges.
- **Wie kann man λ-Übergänge als Wegbeschreibung in Automaten bewerten?**
  Beschreibt den Weg unvollständig.
- **Wie kann man Automaten verkleinern?**
  Löschen von überflüssigen Zuständen: Zustände, die vom Startzustand nicht erreichtbar sind bzw. von denen kein Weg in einen Endzustand führt.
- **Wie kann man Partialität von DFA lösen?**
  Man richtet alle unvollständigen Kanten in einen Fangzustand.
- **Was ist ein *AFA*?**
  Alle Berechnungen eines Wortes müssen akzeptierend sein.
- **Welcher Zusammenhang besteht zwischen DFA,NFA,λ-NFA und REX?**
  Die Sprachklassen, die sie beschreiben sind gleichmächtig.
- **Wie erfolgt die Konstruktion von λ-NFA aus regulären Ausdrücken?**
  Seriell-Parallel Graphen. Invariante: Jeder Graph hat einen Anfangs und einen Endzustand (muss überhaupt nicht sein).

- **Wie eliminiert man λ-Übergänge?**
  Man berechnet für jeden Zustand z alle Nachfolgerzustände bei λ-Übergängen (Wegproblem). Falls unter Nachfolgern sich ein Endzustand befindet, so wird der z zu einem Endzustand gemacht.
- **Wie kann man die λ-Reduktion einfach beweisen?**
  Per strukturelle Induktion über die Anzahl der λ-Kanten. Man behandelt in einem Schritt λ-Kanten von einem Knoten aus. Nach Induktionsannahme ist dann der entstehende Automat äquivalent zu einem ohne λ-Übergänge. Und die einmalige Reduktion verändert die Sprache nicht.
- **Welches zentrale Unterschied gibt es zwischen λ-NFA's und NFA?**
  Erstere können Endlosschleifen produzieren.
- **Wie kann man einen NFA in einen DFA überführen?**
  Durch Potenzmengenkonstruktion (*Rabin&Scott*). Man betrachtet einen Automaten mit der Potenzmenge der Zustände und Übergängen genau in diese Potenzmengen.
- **Wie beweist man den Satz von Rabin und Scott?**
  Ein Wort ist in der Sprache des NFA genau dann, wenn es eine akzeptierende Folge von Zuständen gibt genau dann, wenn es eine akzeptierende Folge von Teilmengen gibt, in denen jeweils die zur Akzeptanz führenden Zustände liegen genau dann, wenn das Wort im DFA liegt.
- **Was sagt der *Satz von Kleene* aus?**
  Jedes DFA beschreibt eine reguläre Sprache.
- **Wie beweist man den Satz von Kleene?**
  Durch Konstruktion eines regulären Ausdruckes zu jedem Automaten. Man zählt die Zustände auf $\{z_1, ..., z_n\}$ und bildet sich Sprachen $R^k_{i,j}$ mit der Semantik: $R^k_{i,j}$ ist eine Sprache, die den Automaten vom Zustand z_i in den Zustand z_j überführen, ohne dabei Zwischenzustände vom größeren Index als k zu haben. Dann gilt
  $$R^{k+1}_{i,j} = R^k_{i,j} \cup R^k_{i,k+1}(R^k_{k+1,k+1})^* R^k_{k+1,j}.$$
  Die Sprache ist eine Vereinigung aller $R^n_{1,i}$, i ist ein Endzustand.
- **Wie gut ist die Überführung von REX in λ-NFA?**
  Planarer serien-parallelgraph. Lineare Komplexität in der Anzahl der Zeichen.
- **Wie gut ist die Überführung von λ-NFA in NFA?**
  Planarität geht verloren. Kürzere Wege (*Speedup*).
- **Wie gut ist die Überführung von NFA in DFA?**
  Exponentiell in der Anzahl der Zustände aufgrund der Potenzmengenkonstruktion.
- **Kennen Sie Fälle, bei denen es nicht kleiner als exponentiell geht?**
  Ja. Das k-te Symbol ist eine 0.
- **Wie gut ist die Überführung von DFA in REX?**
  4^n Zeichen, aber es gibt viele Wiederholgunen. Es werden nur n^3 Terme gebildet.
- **Wie kann man effizient REX in NFA umwandeln?**
  Wie es sich eigentlich gehört (Kleene). Lookahead und zusätzlich verschmelzen der Zustände.
- **Wie kann man effizient NFA in REX umwandeln?**
  In $O(n^4)$ durch Ersetzen von Kanten durch Reguläre ausdrücke und Kürzen von Knoten durch Verschmelzen der Ausdrücke (Formal: ersetzen aller fehlenden Kanten durch Ausdrücke $p\ \emptyset\ q$, Verschmelzen der Vorgänger und Nachfolger, Kanten sind genau die $R^k_{i,j}$.
- **Wie wirkt $\emptyset$ bei regulären Ausdrücken?**
  Das entsprechende Teilwort ist leer.

1.4 Abschlusseigenschaften

- **Unter welchen Operationen sind reguläre Mengen abgeschlossen?**
  Komplemente, Schnitte, Homomorphismen, Vereinigungen, Spiegelung (gespiegelter regulärer Ausdruck).
- **Wie verhält sich REG im Bezug auf MIN und MAX?**
  Ebenfalls abgeschlossen.
- **Unter welchen Operationen ist REG nicht abgeschlossen?**
  Kopieren ($\{ww \mid w \in L\}$) und Permutation. Beides kann man auf $\{a^n b^n \mid n \geq 0\}$ reduzieren.
- **Wie kann man Kopieren auf $\{a^n b^n \mid n \geq 0\}$ reduzieren?**
  Man kopiert $a^n b$ Ausdrücke. Es entstehte im Wesentlichen $a^n b a^n b \sim a^n b^n$.
- **Was ist der Unterschied zwischen Kopieren und Konkatenation?**
  Bei Konkatenation entstehen *alle* möglichen Kombinationen, z.B. Konkatenation von $\{a^n b \mid n \in \mathbb{N}\}$ mit sich selbst liefert auch $a^n b a^m b$ in der Sprache.

1.5 Pumping-Lemmas

- **Welches Prinzip liegt den Pumping-Lemmas zugrunde?**
  Ein zu langer Weg, länger als die Anzahl der Zustände im Automaten, muss eine Schleife haben.
- **Was sagt das einfache *Puming-Lemma* aus?**
  Wenn eine Sprache regulär ist, dann gibt es so ein n, sodass für jedes Wort $|x| \geq n$ der Sprache eine Zerlegung $x = uvw$ existiert, sodass $|v| \geq 1$, $|uv| \leq n$ und $uv^i w \in L$ für alle $i \in \mathbb{N}$.
- **Wie beweist man das Pumping-Lemma?**
  Bei Wörtern, die Länger als die Anzahl der Automatenzustände ist, muss sich mindestens ein Zustand wiederholen.
- **Welche nichtregulären Sprachen ergeben sich durch Pumping-Lemma?**
  $\{a^n, b^n \mid n \in \mathbb{N}\}$, Palindrome, Primzahlen in Unärdarstellung, Nichtprimzahlen.
- **Ist das einfache Pumping-Lemma für die Regularität einer Sprache hinreichend, notwendig oder beides?**
  Nur notwendig dafür, dass eine sprache Regulär ist (Gegenbeispiel $\{a^i b^j c^k \mid i, j, k \in \mathbb{N}\}$).
- **Wozu wird dann das einfache Pumping-Lemma verwendet?**
  Für den Nachweis, dass eine Sprache nicht regulär ist.
- **Welche sinnvolle Verschärfungen kann man bei Pumping-Lemmas verwenden?**
  Einschränkung der Länge des Pumpingwortes und des nicht gepumpen Wortes durch n, oder für beliebige Positionen in w (Ogden)
- **Kennen Sie eine äquivalente Charakterisierung von Regulär?**
  Lemma von Jaffe.
- **Was sagt das *Lemma von Jaffe* aus?**
  Eine Sprache ist genau dann regulär, wenn ein n existiert, sodass für jedes Wort entweder $|w| < n$ ist oder $w = xyz$ mit $|y| \geq 1$ und für alle $i \in \mathbb{N}$ und Wörter $u \in \Sigma^*$ das Wort $xyzu$ genau dann in L liegt, wenn $xy^i zu$ in L liegt. Also Erweiterung auf einen Schwanz u.
- **Was kann man über den Wert von n im Lemma von Jaffe aussagen?**
  Der Wert ist nicht effektiv berechenbar. Wenn die Sprache über eine TM definiert ist, so ist der Wert nicht erkennbar, weil das Regularitätsproblem für TM nicht entscheidbar ist.

1.6 Charakterisierungen für REG

- **Was ist ein zweiwege Automat?**
  Ein Automat, der die Eingabe mehrfach lesen kann.
- **Welcher Zusammenhang besteht zwischen Zweiwege- und normalen Automaten?**
  Es gibt zu jedem Zweiwege-Automaten einen äquivalenzen Automaten.
- **Was besagt der *Satz von Myhill-Nerode*?**
  Eine Sprache ist genau dann regulär, wenn der Index der zugehörigen Rechtskongruenzrelation endlich ist.
- **Wie zeigt man, dass die Rechtskongruenzrelation regulärer Sprachen vom endlichen Index ist?**
  Man definiert für einen Automaten Kongruenzklassen, die dadurch gegeben sind, dass zwei Wörter äquivalent sind gdw. sie vom Startzustand in dem selben Automatenzustand landen. Die Anzahl der Klassen ist endlich, weil endlich viele Zustände erreicht werden können. Die Vereinigung der Klassen mit einem Endzustand repräsentiert offenbar die Sprache. Die Rechtskongruenzrelation der Sprache ist eine Vergröberung der definierten Relation, denn für $x \sim_L y$ gilt $xl \in L$ gdw. $yl \in L$, denn man kann x gegen y austauschen.
- **Wie zeigt man, dass Sprachen mit Rechtskongruenzrelation mit endlichem Index regulär sind?**
  Man konstruiert einen Automaten, der Äquivalenzklassen als Zustände hat und genau dann eine Verbindung von $z_1 \rightarrow_a z_2$ hat, wenn man durch das Symbol a von der Äquivalenzklasse von z_1 zu der von z_2 kommt.
- **Welche Eigenschaft hat der Automat aus dem Beweis des Satzes von Myhill-Nerode?**
  Der ist minimal. Wäre das nicht der Fall, so könnte man den Automaten eine Rechtskongruenzrelation mit weniger Äquivalenzklassen überführen.
- **Wie kann man einen minimalen Automaten konstruieren?**
  Man beginnt mit allen Zuständen als Äquivalenzklassen und verschmilzt nacheinander die mit $\delta(z,x) \in E$ gdw. $\delta(z',x) \in E$. Das macht man durch geeignete Markiereungen (alle Endzustand/Nichtendzustand-Paare markieren, Ein paar wird markiert, wenn für jedes Symbol vom Alphabet die Nachfolgerzustände markiert sind).
- **Welche Komplexität hat Konstruktion eines minimalen Automaten?**
  Kubisch.
- **Gibt es eindeutige Normalformen für reguläre Ausdrücke?**
  Nein.
- **Wie sind REX, EXREX und DFA in der Praxis zu bewerten?**
  Spezifikationen mit REX und EXREX aber die Implementierung mit DFA. Komplexität bei DFA kann sehr hoch sein.
- **Wie kann man reguläre Sprachen durch Homomorphismen beschreiben?**
  Für jede reguläre Sprache gibt es Homomorphismen, sodass $R = f^{-1} \circ g \circ h^{-1}(\$)$. (\$ ist ein Endzustnadszeichen)
- **Was ist die Idee bei der Darstellung mit Homomorphismen?**
  Präpariere den Automaten so, dass keine Befehle in den Startzustand und keine aus dem Endzustand gehen. Man kodiert die Befehle durch (q, a, q').
- **Welcher Zusammenhang besteht zwischen Grammatiken und Automaten?**
  Jedes NFA kann durch eine rechtslineare Grammatik dargestellt werden ($A \rightarrow aB$ und $A \rightarrow a$).
- **Wie konstruiert man zu einem NFA eine Grammatik?**
  Zustände sind Nichtterminale Zeichen (Endzustände zusätzlich noch jedweils eine Terminalproduktion). Übergänge sind Produktionen.

1.7 Entscheidbarkeitsprobleme

- **Welche *Entscheidbarkeitsprobleme* kennen Sie?**
  Wortproblem, Leerheitsproblem, Universalitätsproblem, Endlichkeitsproblem, Äquivalenzproblem, Inklusionsproblem.
- **Was kann man über die obigen Probleme für reguläre Sprachen aussagen?**
  Sie sind alle entscheidbar.
- **Wie hoch ist die Komplexität für das Wortproblem bei DFA, NFA/REX bzw. EXREX?**
  Linear, Quadratisch in Zuständen/Symbolen (Breitensuche) bzw. unklar definiert (Umformen in REX).
- **Wie hoch ist die Komplexität für das Leerheitsproblem bei DFA, NFA/REX bzw. EXREX?**
  Graphenerreichbarkeit bei DFA/NFA und nicht mehr elementar für EXREX (EXPSPACE-hart).
- **Wie hoch ist die Komplexität für das Endlichkeitsproblem bei DFA, NFA/REX?**
  Naiv exponentiell in der Anzahl der Zustände (Durchprobieren aller Wörter mit Länge $n \leq |w| \leq 2n$ oder per Graphenalgorithmen.
- **Wie hoch ist die Komplexität für das Universalitätsproblem bei DFA, NFA/REX bzw. EXREX?**
  Linear über Komplement, PSPACE-Vollständig, EXREX nicht mehr elementar.
- **Wie kann man die Komplexität von Inklusionsproblemen und Äquivalenzproblemen einordnen?**
  Mindestens so schwer, wie Universalitätsproblem. Äquivalenz bildet einen Speziallfall der Universalität mit fixer Sprache Σ^*.

1.8 Anwendungen von regulären Sprachen

- **Wie kann man reguläre Sprachen anwenden?**
  Alle Probleme mit einem endlichen Wertebereich, Scanner/Compiler, State-Charts (Zustandsdiagramme/Protokolle).
- **Wie kann man mit Hilfe von Automaten Pattern-Matching implementieren?**
  Automat, der eine Zustandkette hat aus dem Pattern und stets in den Anfangszustand fällt beim Fehler.
- **Wie kann man die Korrektheit von Protokollen beschreiben?**
  Schleifen, Dead-Ends usw.

2 Kontextfreie Sprachen

- **Welche unterschiedlichen Charakterisierungen gibt es für kontextfreie Sprachen?**
  Kontextfreie Grammatiken, eBNF, NPDA (Nicht deterministische Kellerautomaten), Normalformen, Homomorphie über $Dyck_2$.

2.1 Anwendungen

- **Was kann man mit kontextfreien Sprachen beschreiben?**
  Syntax von Programmen, REX, natürliche Sprachen.
- **Welche Produktionen hat eine kontextfreie Grammatik?**
  $T \to \alpha$, wobei T ein Terminalzeichen ist und α Komposition von Terminal und nichtterminal Zeichen.

- **Welche Beispiele von kontextfreien Sprachen kennen Sie?**
  Jede reguläre Sprache ist kontextfrei, Echte: Klammerungen (Dyck), $\{a^n b^n \mid n \geq \mathbb{N}\}$, $\{a^n b^m \mid n \neq m \geq \mathbb{N}\}$
- **Welche nicht kontextfreien Sprachen kennen Sie?**
  $\{a^n b^n c^n \mid n \in \mathbb{N}\}$, Primzahlen in Unärdarstellung.
- **Was kann man über die eBNF, kontextfreie Grammatiken und Syntaxdiagramme aussagen?**
  Diese sind äquivalente Beschreibungen für kontextfreie Sprachen.
- **Wie kann man die Grammatiken vereinfachen?**
  Nutzlose Zeichen entfernen, λ-Elimination, Elimination von Kettenproduktionen, Chomsky-Normalform und Greibach-Normalform.
- **Wie entfernt man nutzlose Zeichen?**
  Man sucht alle nichterreichbaren Variablen (NL-Vollständig). Man entfernt alle Produktionen, die keine terminalen Wörter ableiten können.
- **Was kann man durch Entfernen nutzloser Zeichen erreichen?**
  Leerheitsproblem ist in Polynomialzeit entscheidbar.
- **Wie entfernt man λ-Produktionen?**
  Man erkennt alle Produktionen, die λ ableiten können (rückwärts markieren), entfernt diese und fügt für jede Regel in der eine markierte Variable vorkam eine Regel ohne dieser Variable.
- **Was kann man durch Entfernen von λ-Produktionen erreichen?**
  Das Wortproblem für kontextfreie Grammatiken braucht höchstens linearen Speicherplatz auf einer nichtdeterministischen Turingmaschine.
- **Was muß man bei λ-Reduktion beachten?**
  Falls λ in der Sprache, so muss eine Produktion der Form $S \to \lambda$ eingeführt werden. Dann darf man aber nicht auf S zurückschliessen.
- **Was ist eine Kettenproduktion?**
  Eine Produktion der Form $A \to B$.
- **Wie kann man Kettenproduktionen eliminieren?**
  Einsetzen der rechten Seiten in die linken.
- **Was kann man durch Entfernen von Kettenproduktionen erreichen?**
  Das Wortproblem braucht nichtdeterministisch linear Zeit. Geht deterministisch in $O(n^3)$ (CYK-Algorithmus).
- **Was ist *CNF (Chomski-Normalform)*?**
  Man hat nur Produktionen der Form $A \to BC$ oder $A \to a$.
- **Was erreicht man durch die CNF?**
  Ableitungsbäume sind Binärbäume.
- **Was ist *GNF (Greibach-Normalform)*?**
  Produktionen haben die Form $A \to aBC$. (Nicht $A \to aB$, sonst hat man reguläre Sprachen)
- **Was folgert man aus GNF?**
  Bei jedem Ableitungsschritt wird ein Terminalzeichen erzeugt. Die Länge des Wortes ist Anzahl der Ableitungsschritte.
- **Was ist eine linear kontextfreie Grammatik?**
  Eine Variable auf der rechten Seite $A \to Ba$.
- **Welche Beispiel-Sprachen sind linear kontextfrei und welche echt kontextfrei?**
  $\{a^n b^n \mid n \geq \mathbb{N}\}$ linear kontextfrei. $\{a^n b^n a^m b^m \mid n, m \geq \mathbb{N}\}$ sind in CFL aber nicht linear kontextfrei.

2.2 Pumping Lemma für CFL

- **Was sagt das *Pumping-Lemma für CFL* aus?**
  Eine Sprache ist kontextfrei, falls es so ein n gibt, sodass für jedes Wort $|a| \geq n$ eine Zerlegung $uvwxy$ gibt mit $|vx| \geq 1$, $|vwx| \leq n$ und für alle $i \in \mathbb{N}$ gilt $uv^iwx^iy \in L$.
- **Wie beweist man das Pumping-Lemma für CFL?**
  Wähle eine Grammatik in CNF. In jedem Ableitungsbaum, der zu groß ist, gibt es mehrfache Verwendung einer Produktion. Man wähle das erste und das letze Vorkommen. Das liefert das Pumpen, denn links und rechts beim ersten Vorkommen der Produktion liegen u und y. Unterhalb der letzten Anwendung der Produktion liegt w und links und rechts liegen i-fache v und x.
- **Was kann man mit Pumping-Lemma zeigen?**
  Das eine Sprache nicht kontextfrei ist.
- **Kennen Sie eine nicht kontextfreie Sprache?**
  $\{a^nb^nc^n \mid n \in \mathbb{N}\}$. Sei ein Wort $a^nb^nc^n$ gegeben. Wegen der Bedingung $|vwx| \leq n$ kann nicht vx aus allen drei Zeichen bestehen. Wegen $|vx| \geq 1$ muss aber $uwy \in L$ gelten. Aber uwy kann aus obigen Überlegungen keine Form $a^mb^mc^m$ haben.
- **Ist das Pumping Lemma für die Zugehörigkeit zu CFL hinreichend?**
  Nein. $b^* \cup \{a^kb^{n^2} \mid k, n \geq 1\}$

2.3 Komplexitäten von Problemen

- **Welche Komplexität hat das Wortproblem für CFL?**
  $O(n^3)$ mit dem CYK-Algorithmus.
- **Welche Probleme bei CFL sind P-Vollständig?**
  Leerheit, Endlichkeit, Leeres Wort.
- **Wie viel Speicherplatz braucht das Wortproblem für CFL?**
  $O(\log^2 n)$.
- **Welche Probleme sind für CFL unentscheidbar?**
  Universalität, Inklusion, Äquivalenzproblem, Regularitätsproblem.

2.4 Kellerautomaten

- **Was ist ein *Kellerautomat (PDA)*?**
  Ein Automat, der bei jedem Zugriff auf einen Keller schreiben kann oder von einem Keller lesen kann.
- **Ist ein PDA deterministisch oder nicht?**
  Nichtdeterministisch.
- **Was ist ein akzeptierender Zustand bei einem PDA?**
  Leerer Keller, Endzustand, beides. Alles äquivalent.
- **Was kann man über das Verhältnis zwischen PDA und CFG aussagen?**
  Beschreiben das selbe.

2.5 Abschlusseigenschaften von CFL

- **Unter welchen Operationen ist CFL abgeschlossen?**
  Vereinigung, Konkatenation, Sternbildung.
- **Unter welchen Operationen ist CFL nicht abgeschlossen?**
  Komplement, Schnitt.
- **Kennen sie eine typische CFL?**
  Klammerungen mit zwei Klammerarten (Homomorphismen: $h(h^{-1}(Dyck_2) \cap R)$.

- **Was ist die *Greibach-Sprache*?**
  Nichtdeterministische Variante von doppelten Klammerungen.
- **Was kann man über die Greibach Sprache aussagen?**
  Jede CFL läßt sich als Urbild eines Homomorphismus bzgl. der Greibachsprache darstellen.

2.6 Sprachklassen zwischen CFL und REG

- **Kennen Sie Sprachklassen zwischen CFL und Regulär?**
  Deterministisch kontextfreie Sprachen.
- **Wodurch werden die DCFL erzeugt?**
  Durch deterministische Kellerautomaten.
- **Kennen Sie Beispiele für deterministisch kontextfreie Sprachen?**
  Korrekte Klammerungen mit einer Klammer ($LR(k)$ Sprachen), Palindrome.

2.7 Andere Ersetzungssysteme

- **Kennen Sie andere Ersetzungssysteme?**
  Typ-0 Sprachen (Kontextsensitiv), Lindenmayer Systeme und Patternsprachen.
- **Was sind Lindenmayer-Systeme?**
  Totalparallele Ersetzungen.
- **Wo sind die Lindenmayer Systeme einzuordnen?**
  Zwischen CFL und CSL.

3 Turing Maschinen

- **Was motiviert *Turing-Maschinen*?**
  Das einfachste Berechnungsmodell, Befehle haben lokale Wirkung, keine Speicherverwaltung.

3.1 Definition des Maschinenmodells

- **Was ist bei der Definition einer DTM zu beachten?**
  Deterministisch, Endliche Kontrolle, Ausgabeband wird nur beschrieben, Wirkung nur lokal um die Köpfe.
- **Was ist die Konfiguration einer mehrband-TM?**
  Ein Wort $b_0|e_0, b_1|e_1, ..., b_n|e_n, y$ mit der Position der Köpfe (jeweils auf dem ersten Zeichen von e_k) auf den Bändern und dem Ausgabewort.
- **Welche Arten der Berechnung kennen Sie bezüglich des Ergebnisses?**
  Akzeptierend, haltend, berechnend.
- **Welche Eigenschaft hat die berechnete Funktion für eine TM bezüglich ihres Definitionsbereiches?**
  Berechnete Funktion ist partiell.
- **Was ist *Gödelisierung einer DTM*?**
  Eindeutige binäre Kodierung einer TM. Wichtig ist, dass man die Befehle kodiert.
- **Welche Sprachklassen ergeben sich im Zusammenhang mit TM?**
  RE (Sprachen aller DTM), *REK* (Sprachen haltender DTM), *PR* (Es gibt eine DTM, die die Funktion berechnet), *TR*, *REG* (Reguläre Sprachen akzeptiert von Automaten), *CFL* (Kellerautomaten), *P* (Effizient lösbare Probleme), *NP* (nicht effizient lösbare Probleme).
- **Was ist NP-hart?**
  NP \ P.

- **Was besagt die These von Church und Turing?**
  DTM erfaßt den Berechenbarkeitsbegriff.
- **Wie kann man die Anzahl der Turing-Maschinen einsehen?**
  Turing-Maschinen sind aufzählbar und somit ist deren Anzahl der Kardinalität von $\mathbb{N}$ gleich. Viel weniger als die Anzahl von Abbildungen von $\mathbb{N} \to \mathbb{N}$.
- **Welche spezielle interessante Probleme gibt es?**
  $\{a^n b^n \mid n \geq 0\}$, $\{a^n b^m \mid n \neq m\}$, $\{a^n b^n c^n \mid n \geq 0\}$, unäre Primzahlen, binäre Primzahlen, Primzahlendarstellung, SAT, Hamilton.
- **Was ist die *Universelle Sprache*?**
  Wörter: Gödelisierung einer DTM gefolgt von einem akzeptierten Wort.
- **Was ist die *Diagonalsprache*?**
  Wörter: Gödelisierung einer DTM gefolgt von einem nicht akzeptierten Wort.
- **Was ist die *Spezielle Diagonalsprache*?**
  Wörter: Gödelisireung einer DTM, die sich selbst nicht akzeptiert.
- **Wie schwer sind uniäre/binäre Darstellungen von Primzahlen?**
  Binärdarstellung schwer aber in P, Unärdarstellung wegen zu großer Länge in $O(n)$.
- **Wie schwer sind die Universelle/Diagonalsprache?**
  Sehr schwierig. Diagonalsprachen sind nicht rekursiv aufzählbar.
- **Welche Veränderungen liefert Nichtdeterminismus bezüglich der Konstruktion der Maschine?**
  Die Übergangsfunktion ist mehrwertig.
- **Wie sehen die Berechnungen der Wörter bei einer NTM?**
  Berechnungs-Bäume (Zustandsübergangsdiagramme) und Berechnungsgraphen (Konfigurationsgraphen).
- **Wie bemißt man Komplexität bei NTM?**
  Laufzeit ist die Länge der Pfade (auch unendlich lange Pfade möglich) im Berechnungsgraphen und Speicher ist die Größe der Konfigurationen.
- **Welche Technik verwenden nichtdeterministischen Turingmaschinen beim Erkennen der Wörter?**
  Rate (richtig) eine Befehlsfolge und verifiziere ob diese akzeptierend ist.

3.2 Komplexität

- **Wie sind genau die Komplexitätsklassen definiert?**
  Sprachen, die durch Turing-Maschinen in bestimmter Zeit bzw. unter Verwendung bestimmten Platzes erkannt werden.
- **Welche Maße verwendet man zur Bewertung der Zeitkomplexität?**
  Minimale Berechnungslänge ($T(w)$), maximale Berechnungslänge $T(w)$ für $|w| = n$ ($T(n)$).
- **Welche Maße verwendet man zur Bewetung der Speicherkomplexität?**
  Größe zwischenzeitlich benötigte Speicherbedarf $S(B(w))$, Für ein Wort starke/schwache Berechnungsbeschränkung (für alle Worte oder für ein Wort) $S(w)$.
- **Welche Grundlegenden Komplexitätsklassen gibt es?**
  XSPACE($f(n)$) für Speicherbeschränkte und XTIME($f(n)$) für zeitbeschränkte XTM.
- **Welche Abkürzungen verwendet man zusätzlich?**
  $P = DTIME(n^{O(1)})$, $NP = NTIME(n^{O(1)})$, $PSPACE = DSPACE(n^{O(1)})$.
- **Wenn man $n^{O(1)}$ mit Poly abkürzt, wie kann man die Klassen beschreiben?**
  L = DSPACE($\log n$) $\subseteq$ NL = NSPACE($\log n$) $\subseteq$ P = DTIME(poly) $\subseteq$ NP = NTIME(poly) $\subseteq$ PSPACE = DSPACE(poly) $\subseteq$ EXPTIME = DTIME(2^{poly}).

- **Welcher Zusammenhang besteht zwischen DSPACE(poly) und NSPACE(poly)?**
  Die sind gleich.
- **Welcher Zusammenhang besteht zwischen DTIME(2^{poly}) und NTIME(2^{poly})?**
  Inklusion.
- **Wie ist in die Hierarchie die Menge der regulären Sprachen einzuordnen?**
  REG = DSPACE(1) = NSPACE(1).
- **Welche echten Inklusionen sind in der Hierarchie beweisbar?**
  NSPACE(1) $\subset$ DSPACE(log), XSPACE(log) $\subset$ XSPACE(poly).
- **Gibt es eine Komplexitätsklasse, die die alle Umschliesst?**
  RE, REK.
- **Gibt es eine Komplexitätsklasse weiter als REK?**
  Nicht mit Hilfe der Turingmaschinen beschreibbar.
- **Wieso darf man mit "O" sorglos umgehen?**
  Aufgrund der Speedup-Theoreme (Vergrösserung des Alphabets).
- **Was versteht man unter *Komplementbildung* bei einer Komplexitätsklasse?**
  Eine Komplementklasse enthält genau die Komplementsprachen der ursprünglichen Klasse. Man schreibt Co- davor.
- **Welche Komplexitätsklassen sind trivial unter Komplement abgeschlossen?**
  Deterministischen Speicher und Zeitklassen.
- **Warum gilt obige Behauptung?**
  Man kann bei deterministischen Modellen akzeptierende mit nichtakzeptierenden Zuständen vertauschen.
- **Was muss man aber beachten beim dem Vertauschen von akzeptierenden und nichtakzeptierenden Zuständen?**
  Es gibt möglicherweise unendliche Schleifen. Dafür muss man eine Uhr bzw. einen Speicherzähler einbauen.
- **Wieso funktioniert eine Uhr nicht bei nichtdeterministischen Modellen?**
  Ein nichtdeterministischer Automat kann eine Schleife wieder verlassen.

3.3 Konstruktion und Entwurf von TM

- **Was ist eine *many one Reduktion*?**
  Eine Sprache L' heißt auf L many one reduzierbar, wenn es eine totale berechenbare Funktion f gibt mit $w \in L'$ gdw. $f(w) \in L$. Man schreibt $L' \leq L$.
- **Wie beweist man Inklusionen von zwei Komplexitätsklassen?**
  Man Simuliert die TM auf dem selben Speicherplatz in polynomieller Abhängigkeit der Zeit.
- **Welche Beweisanteile müssen stets vorliegen?**
  Konstruktionsvorschrift, Korrektheitsbeweis, Komplexitätsanalyse.
- **Welche grundsätzlichen Tricks gibt es dabei?**
  Große Alphabete, mehrere Bänder, Längere Programme (endliche Kontrolle).
- **Welche Graphenverfahren werden zur Komplexitätsklasseninklusion verwendet?**
  Breitensuche auf dem Berechnungsgraphe, Divide&Conquer (Savitch NSPACE(S) $\subseteq$ DSPACE(S^2)), Induktives Zählen (Immerman NSPACE = co-NSPACE).
- **Mit welcher Komplexität kann man eine k-Band DTM auf einer 1-Band-DTM simulieren?**
  $T(n)^2$. Man braucht $T(n)$ Zeit um einen Befehl zu simulieren und $T(n)$ Zeit um durch den Berechnungsbaum durchzukommen.

- **Wie viel Zeit braucht man um eine k-Band DTM auf einer 2-Band-DTM zu simulieren?**
  Man Teilt das Band nach dem man die Bänder verschmolzen hat in exponentiell wachsende Blöcke. Bei Bedarf kopiert man die Blöcke in den nächst grösseren Bereich. Das kostet $c2^i$ Zeit bei $T(n)/2^i$ Bewegungen. Insgesamt ergibt sich durch Summieren der $\log(T(n))$ Blöcke die Komplexität.
- **Was bedeutet *Keller*, *Zähler* bzw. *Schlange*?**
  Kellerverhalten eines Turingbandes, Stack mit einelementigem Alphabet, Zwei Köpfe mit +1 Bewegungen.
- **Wie simuliert man zwei Keller mit einer Turing-Maschine?**
  Vom Kopf links und rechts die Keller. Gleiche Zeit, gleicher Speicherplatz.
- **Wie simuliert man eine Schlange?**
  Zwischen den Schlangenköpfen hinundherlaufen. Zeit $T(n)S(n)$, gleicher Speicherplatz.
- **Wieso ist die Simulation von einem Zähler mit einem Keller exponentiell?**
  Man kodiert unär beim Zähler. Daher muss der Zähler umkodiert werden.
- **Wie kann man 2 Zählermaschinen einordnen?**
  Sie sind universell, wie TM.
- **Was sagen die Speedup-Theoreme aus?**
  SPACE(S(n)) = SPACE(O(S(n))) und TIME(T(n)) = TIME(O(T(n))) für T(n) » n.
- **Was wird für SPACE und TIME benötigt?**
  Für SPACE eine c-Kompression. Für TIME eine $16c$ Kompression (in 8 zuständen werden die benachbarten Felder gelesen und 16c Schritte ausgeführt).
- **Warum funktionieren die Speedup-Theoreme?**
  Zustände sind kostenlos.
- **Welcher Zusammenhang besteht zwischen NTIME(O(n)), NTIME(n) und NTIME2(n)?**
  Alle gleich.

3.4 Registermaschinen

- **Was ist eine *RAM*?**
  Random Access Maschine. Von Neumann Modell. Realistischeres Berechnungsmodell.
- **Was ist der Unterschied zwischen uniformen und logarithmischen Kosten?**
  Bei uniformer Betrachtung haben Objekte einheitliche Kosten. Bei logarithmischer Betrachtung wird die Anzahl der Bits gezählt.
- **Wie wird bei RAM die Zeitkomplexität gemessen?**
  Anzahl der Schritte bzw. gewichtet mit den nichtuniformen Kosten.
- **Wie wird bei RAM die Speicherkomplexität gemessen?**
  Anzahl der belegten Zellen bzw. Anzahl der Bits.
- **Welche RAM-Modelle kennen Sie?**
  Berechnungsarten kombinieren, parallel (PRAM), echte RAM (RRAM), nichtdeterministisch.
- **Wie verhalten sich RAMS mit Rechenoperationen bezüglich der Kosten?**
  Zu jeder Maschine mit logarithmischen Kostenmaß kann man effektiv eine mit uniformen angeben. URAM-log-TIME/SPACE(T(n)) $\subseteq$ RAM-UTIME/SPACE(T(n)).
- **Welche Komplexität hat die Umformung von einer uniformen in eine logarithmische AddRAM?**
  $T(n)^2$ bei beschränkter Zahlengröße der Eingabe. Durch addieren (rekursives verdoppeln) kann die Maschine nur $c2^{T(n)}$ lange Wörter konstruieren. Dadurch ergibt sich die Abschätzung $T(n)\log(2^{T(n)})$.

- **Mit welcher logarithmischer Komplexität kann man MultRAM in AddRAM simulieren?**
  $O(T(n)^3)$. Multiplikation durch rekursives Verdoppeln in einem Unterprogramm. Zuerst wird die Binärdarstellung von eines der Faktoren berechnet. Die Multiplikation kostet bei geschickter Darstellung $O(T(n)^2)$.
- **Wit welcher Komplexität kann man MultRAM mit uniformen Kosten auf logarithischen Kosten simulieren?**
  Genau $\Theta(2^{T(n)})$, also exponentieller Aufwand. Multiplikatives rekursives Verdoppeln liefert 2^{2^n}, was bereits 2^n logairthmische Kosten hat.
- **Was darf man auf einer MultRAM mit uniformen kosten nicht machen?**
  Multiplikativ verdoppeln.
- **Wie verhalten sich k-Band DTM und ARAMS?**
  Man kann eine k-Band DTM in $O(T(n))$ bei uniformen und $O(T(n)\log S(n))$ bei nicht uniformen kosten simulieren. Man speichert den Zustand der Maschine verzahnt in den Speicher. Jeder Schritt kostet $\log(S(n))$.
- **Mit welcher Komplexität kann man eine ARAM mit einer DTM simulieren?**
  $O(T(n)^2)$.
- **Welche RAM Modelle sind polynomiell äquivalent?**
  U-ARAM, L-ARAM, M-RAM, Mehrband DTM.
- **Welche Modelle sind nicht polynomiell äquivalent?**
  X-SRAM und X-ARAM, U-ARAM und U-MRAM.

4 Komplexitätshierarchien

4.1 Simulationen

- **Was ergibt sich aus der Tatsache, dass jede DTM auch eine NTM ist?**
  $DTIME(f) \subseteq NTIME(f)$ und $DSPACE(f) \subseteq NSPACE(f)$.
- **Wie kann man den Speicherplatz beschreiben, den eine NTM für k Schritte braucht?**
  Der ist nicht mehr als k Zellen.
- **Was ergibt sich aus obiger Bemerkung?**
  $XTIME(f) \subseteq XSPACE(f)$
- **Was kostet ein gleichzeitiger Wechsel von N zu D und Space zu Time?**
  Für $f \geq \log$ gilt $NSPACE(f) \subseteq DTIME(2^{O(f)})$.
- **Wieso gilt obige Aussage?**
  Man simuliert eine durch $f(n)$ speicherbeschränkte NTM durch eine $O(n2^{O(f(n))})$ zeitbeschränkte DTM und betrachte den Berechnungsgraphen. Die Anzahl der Konfigurationen ist kleiner $c^{f(n)}$. Breitensuche liefert die Komplexität.
- **Welcher Bestandteil liefert den exponentiellen Anteil?**
  Alphabet der TM hoch die Anzahl der Schritte.
- **Was muss man beachten bei der Implementierung?**
  Bei Lazy-Abarbeitung kann man nicht erkennen, ob eine Schleife kommt. Daher muss man sich auf die Schranke stützen und rechtzeitig abbrechen.
- **Wie implementiert man das Abbrechen?**
  Man baut eine Uhr und bricht zu lange Berechnungspfade (also Breitenschlangenphasen) ab.
- **Wie viel Speicher kostet der Einbau der Uhr?**
  Keinen Zusätzlichen. Man zählt die Anzahl der Schritte in c-adischer Darstellung. Wegen $S(n) \geq \log n$ gilt $n \leq 2^{S(n)}$ und somit $\log_2(n) \leq S(n)$. Also knapp bemessen.

- **Welche Form hat dann der Berechnungsgraph?**
  Der ist wegen der Uhr azyklisch.
- **Funktioniert das auch mit Berechnungsbaum statt Berechnungsgraphen?**
  Nein der Baum ist zu tief. Enthält doppelt exponentielle Anzahl von Knoten/Blättern.
- **Welche Konsequenzen kann man durch den ND/ST Wechsel ziehen?**
  $NL \subseteq P$, $PSPACE \subseteq EXPTIME$.
- **Mit welcher Komplexität kann man NTIME auf DSPACE simulieren?**
  Mit gleicher. $NTIME(f) \subseteq DSPACE(f)$
- **Wieso gilt obige Aussage?**
  Man simuliert eine NTM auf einer zeitbeschränkten DTM und zählt im Berechnungsbaum mit Tiefensuche. Die Konfigurationen sind kleiner $f(n)$. Die werden aufgezählt und simuliert.
- **Funktionieren hier auch Berechnungsgraph oder Breitensuche?**
  Nein die Queue z.B. für BFS braucht zu viel Platz.
- **Was liefert obige Aussage?**
  $NP \subseteq PSPACE$
- **Was bedeutet eine Funktion ist *stark bandkonstruierbar*?**
  Es gibt eine DTM, sodass bei jeder Eingabe mit Länge n TM genau $f(n)$ Speicher/Zeit benötigt.
- **Was bedeutet eine Funktion ist *schwach bandkonstruierbar*?**
  Es gibt eine DTM, sodass auf Wörtern der Sprache die TM weniger als $f(n)$ Speicher/Zeit braucht. Es gibt aber ein Wort der Länge n, bei dem $f(n)$ Speicherplatz benötigt wird.
- **Welche Aussage kann man über Bandkonstruierbarkeit bei ausreichend großen Funktionen treffen?**
  Ab einer geeigneten Größe (linear) spielt die keine Rolle. (Man erzeugt alle Wörter der Länge n lexikographisch und simuliert diese)
- **Warum benötigt man den Unterschied zwischen stark und schwach bandkonstruiert überhaupt?**
  Es gibt eine DTM z.B., die nie mehr als $\log(\log(n))$ Zeit braucht.
- **Welche Maschine braucht nie mehr als $\log(\log(n))$ Speicher?**
  Für die Sprache der benachbarten durch Trennsymbol getrennten natürlchen Binärzahlen, die aufsteigend nacheinander sortiert sind. Das prüfen, ob benachbarte Zahlen nachfolger voneinander sind, braucht $\log n$ Speicher. Wenn man das für alle k macht bekommt man in etwa $\log(k!) = \log(\log(n))$.
- **Kennen Sie Funktionen, die stark zeit/speicherbeschränkt sind?**
  $\log n$, $\log^2 n$, Polynome, 2^n.
- **Wie kann man aus alten neue zeit/speicherbeschränkte Funktionen erzeugen?**
  Addition, Multiplikation, Potenzierung.
- **Was sagt der Satz von *Savitch* aus?**
  Für $f(n) \geq \log n$ gilt $NSPACE(f) \subseteq DSPACE(f^2)$ für f stark bandkonstruierbar. Eigentlich $DSPACE(S(n)\log(T(n)))$ (stark/schwach).
- **Wie beweist man den Satz von Savitch?**
  Man simuliert eine NTM auf DTM und prüft mit Divide&Conquere ob es einen Weg von einer Konfiguration von einer anderen in 2^k schritten gibt. Der Speicheraufwand ist dabei $S(n)$ für lokale Parameter und $\log(T(n))$ die Rekursionstiefe.
- **Können Sie die rekursive Prozedur aufschreiben?**
  Falls $k = 0$ Vergleich der Konfigurationen oder der Nachfolgerkonfiguration. Sonst prüfe für alle Konfigurationen ob $Test(K, K'')$ und $Test(K'', K')$ klappt und gibt 1 zurück. Sonst 0.

- **Wie simuliert man die NTM auf DTM nach Savitch?**
  Test anwenden auf Start und alle akzeptierenden Konfigurationen. (Falls schwach speicherbeschränkt terminiert der Test nicht).
- **Welche Konsequenz liefert der Satz von Savitch?**
  $DSPACE(poly) = NSPACE(poly)$. Also $NSPACE = PSPACE$.
- **Was kann man über analoge Aussagen zum Satz von Savitch für Funktionen unter $\log n$ aussagen?**
  Analoge aussagen unbekannt. Auf jedenn Fall ist Savitvh falsch für $\{a^i b^j \mid i \neq j\}$ ist schwach auf $\log \log n$ erkennbar. Stark benötigt mindestens $\log n$.
- **Kann man den Satz von Savitch verbessern?**
  Offen.
- **Was kann man sagen über große Klassen?**
  Für stärkeren Wachstum als bei Polynomen gilt $NSPACE(f(n)) = DSPACE(f(n))$.

4.2 Hierarchiesätze

- **Was beschreiben die Hierarchiesätze?**
  Echte Inklusionen zwischen Komplexitätsklassen?
- **Welche zentralen Sätze kennen Sie?**
  Für $f_1(n) << f_2(n)$ gilt $SPACE(f_1(n)) \subset SPACE(f_2(n))$ (Man braucht nur starken Speicher), $TIME(f_1(n)) \subset TIME(f_2 \log(f_2(n)))$ (Wegen dem Zähler auf einem Band) und $TIME_k(f_1(n)) \subset TIME_{k+1}(f_2(n))$.
- **Was kann man für Klassen mit $S_1(n) << S_2(n)$ aussagen?**
  Es gibt stets eine Sprache in $SPACE(S_2)$, die nicht in $SPACE(S_1)$ liegt. Man konstruiert eine S_2 Bandbeschränkte DTM mit $L = L(TM)$. Jede Turingmaschine aus $SPACE(S_1)$ macht einen Fehler.
- **Welche Sprache verwendet man dabei?**
  $L \subseteq \{w \mid w = 1^* < M >, w \notin L(M)\}$ mit $w \in L$ gdw. $w \in L(M_2)$.
- **Wie konstruiert man die Maschine?**
  M_2 reserviert S_2 Speicher, simuliert M und verwirft gdw. M akzeptiert oder wenn der Speicherplatz nicht ausreicht.
- **Warum kann M_2 die Operationen durchführen?**
  Die Simulation braucht $\log(\Gamma)S_1$ Speicherplatz. Daher kann S_2 den Speicherplatz bieten.
- **Wieso versagt jede TM auf der konstruierten Sprache mindestens einmal?**
  Es gibt eine genügend lange Kodierung der Maschine M_1 aufgrund des Präfix. Dann gilt $w \in L(M_1)$ gdw. $w \notin L$ gdw. $w \notin L(M_1)$.
- **Wann sind die Komplexitätsklassen unvergleichbar?**
  Wenn die Funktionen bzgl. der O-Notation es nicht sind.
- **Gibt es eine Funktion mit $DTIME(f) = REK$?**
  Nein. Zu jeder Funktion f gibt es eine noch rekursive Sprache, die nicht in $DTIME(f)$ liegt.
- **Gibt es eine größte Zeitbeschränkungsfunktion?**
  Analog. Nein.
- **Wie kann man das beweisen?**
  Konstruiere eine Sprache $\{w =< M > | \ M$ akzeptiert w nicht in $f\}$. Angenommen es gäbe eine TM M mit $L = L(M)$. Dann liefert $w =< M >$ einen Widerspruch.
- **Was besagt der Satz über die *allgemeine Zeithierarchie*?**
  Für $\liminf f_1(n)\frac{\log f_1}{f_2} = 0$ gibt es eine Sprache, die in $DTIME(f_2)$ ist aber nicht in $DTIME(f_1)$.

- **Warum taucht hier ein log-Faktor auf?**
  Simulation auf einer 2-Band DTM. Man kann die Anzahl der Bänder nicht in Abhängigkeit von der zu simulierenden Maschine wählen.
- **Wie beweist man obigen Satz?**
  Simulation auf einer 2-Band DTM. Ansonsten genau so, wie die allgemeine Zeithierarchie.
- **Was folgert man aus dem allgemeinen Hierarhiesatz?**
  Es gibt beliebig viele Klassen zwischen polynomiellen Klassen oder logarithmischen Potenzen.
- **Was kann man über die Zeithierarchien bei fester Bandzahl aussagen?**
  Es gibt stets Sprachen in $DTIME_{k+1}(f_2)$, die nicht in der Klasse $DTIME_k(f_1)$, falls $f_1 << f_2$.
- **Wie beweist man den obigen Satz?**
  Die $k+1$ Maschine kopiert M auf ein neues Band und simuliert M. Dabei kostet jeder Schritt die Länge der Gödelisierung von M wegen der Suche.
- **Auf welche Bedingungen lassen sich die Hierarchiesätze reduzieren?**
  Komplementbildung und Simulation.
- **Was bedeutet *universell*?**
  $Y(f_2)$ ist universell für $X(f_1)$, wenn eine Y-Maschine existiert, die jede X maschine simulieren kann. $X(f_1) \subseteq Y(f_1)$.
- **Was bedeutet *kontrollierbar*?**
  Y ist bzgl. X f_2-Kontrollierbar, wenn die universelle Maschine Y bei der Simulation die Schranke f_2 garantieren kann.
- **Was besagt der *allgemeine Hierarchiesatz*?**
  $Y(f_2)$ Universell für $X(f_1)$, Y ist f_2-Kontrollierbar und $Y(f_2)$ abgeschlossen unter Komplement liefert $Y(f_2) \subset X(f_1)$.
- **Wie beweist man den allgemeinen Hierarchiesatz?**
  Man simuliert Maschinen der Untergeordneten Klasse auf Eingaben der Form $1^* < M >$ auf Kosten f_2. Sowohl Komplement als auch die Sprache selbst liegen in $Y(f_2)$. Auf genügend lange Kodierungen kann die universelle Maschine die Simulation abschliessen, sodass für keine Maschine in X das Komplement darstellbar ist.
- **Welche Konsequenz liefert der allgemeine Hierearchiesatz?**
  Echte Hierarchie bei NSPACE mit Immerman.
- **Wie kann man den Hierarchiesatz verfeinern?**
  Für t_1, t_2, f Platzkonstruierbar $DTIME(t_1) \subseteq DTIME(t_2) \rightarrow DTIME(t_1 \circ f) \subseteq DTIME(t_2 \circ f)$ für $t_1 \geq (1+\epsilon)n$. Analog Speicher ($\geq \log n$).
- **Wie beweist man den Translationssatz?**
  Man wähle eine Sprache in $DTIME(t_1 \circ f)$ und bilde für die $t_1 \circ f$ Zeitbeschränkte Maschine M_1 die Sprache

  $$L_2 = \{X\#^r \mid M_1 \text{ erkennt } X \text{ in } t_1(|X|+r)\}.$$

  Wähle eine Maschine M_2 für L_2, welche t_1 zeitbeschränkt ist. Dann ist $L_2 \in DTIME(t_2)$ nach Voraussetzung. Man betrachte die t_2 beschränkte Maschine M_3, die L_2 erkennt. Bilde nun $t_2 \circ f$-Zeitbeschränkte Maschine M_4, die M_3 auf Eingabe $X\#^{f(|X|)-|X|}$ simuliert. Es gilt für die Sprache $L_4 = L(M_4)$. $X \in L_4$ gdw. $X\#^l \in L_3$ gdw. $X\#^l \in L_2$ gdw. $X \in L_1$.
- **Was muss man bei Speicherhierarchien beachten?**
  Man kann die Eingabe nicht mit Symbolen verlängern. Statt dessen wird ein Zähler benutzt.
- **Was folgert man aus dem Translationssatz?**
  Es gibt $NSPACE$ Hierarchien (mit Savitch). DTIME(n) $\subset$ DTIME($n \log n$).

- **Was sagt *Borodins Gap Theorem*?**
  Es existiert für eine totale berechenbare Funktion $g \geq n$ eine total berechenbare Funktion $s : \mathbb{N} \to \mathbb{N}$ mit DITME(s) = DTIME($g \circ s$).
- **Wie beweist man das Theorem?**
  Da in einer Aufzählung von TM die Menge der $\{t_k(n) \mid 1 \leq k \leq n\}$ (Aufwände von den ersten n TM's auf Eingaben der Länge n) endlich ist, so gibt es eine Lücke. Die Lücke kann man durch systematisches Zählen berechnen (nicht konstruierbar).
- **Für welche speziellen Klassen gilt DSPACE = NSPACE?**
  Regulär, Lücke unter $\log \log n$, AuxPDA (Charakterisiertung von P).
- **Für welche Klassen gilt TIME = SPACE?**
  Haltende TM, Primitiv rekursive Funktionen.
- **Was kann man über fas überall speicherbeschränkte TM's sagen?**
  Die sind speicherbeschränkt, weil man die endlich vielen anderen Eingaben merken kann.
- **Kann man den Speicherbedarf effektiv per TM bestimmen?**
  Ja. Mit einer Uhr.

5 Speicherkomplexitäten L und NL

5.1 L und NL Probleme

- **Warum sind L und NL Probleme interessant?**
  Mit Hilfe der Translationstechnik kann man die Hierarchien auf andere Probleme übertragen.
- **Welche Annahmen trifft man bei L und NL?**
  Eingaben sind geeignet kodiert.
- **Welche Sprachen liegen in L?**
  $\{a^n b^n \mid n \in \mathbb{N}\}$, Klammerungen, Palindrome, symmetrische Graphen.
- **Welche universelle Eigenschaft besitzen $\log n$ beschränkte TM?**
  Man kann korrekte oder falsche Berechnungen jeder TM kontrollieren. Man kann die Sprache der akzeptierenden Berechnungen erkennen.
- **Was wird kontrolliert?**
  Zustandsübergänge und Inhalt der Arbeitsbänder.
- **Warum wird nur $\log n$ Platz gebraucht?**
  Weil sich vom Zustand in den nächsten nur lokal was ändert. D.h. man braucht nur kontrollieren, ob alles außerhalb gleich ist, wozu man nur das Zeichen speichert, welches kontrolliert wird.
- **Ist das Leerheitsproblem für L entscheidbar?**
  Nein. Wenn es entscheidbar wäre, dann könnte man für eine TM prüfen, ob die Menge der akzeptierenden Konfigurationen leer ist und somit für beliebige TM das Leerheitsproblem entscheiden.
- **Welche Abschlusseigenschaften haben L und NL?**
  Vereinigung, Komplement, Schnitt, Konkatenation. Stern für L unklar. Homomorphie negativ für beide.
- **Warum sind N und NL nicht unter Homomorphismen abgeschlossen?**
  Weil der Abschluss ganz RE liefert.
- **Was sagt der Satz von *Immerman* aus?**
  Für stark bandkonstruierbare Funktion $f \geq \log n$ gilt $NSPACE(f) = CoNSPACE(f)$.

- **Wie beweist man den Satz von Immerman?**
  Die Idee liegt im induktiven Zählen aller erreichbaren Konfigurationen. Es gibt eine nichtdeterministische Prozedur $REACH(t, r, C_1, ..., C_n)$, die bei der Eingabe der Anzahl von in höchstens t erreichbaren Konfigurationen r auf dem Speicherplatz $O(f)$ entscheidet, ob eine der Konfigurationen $C_1, ..., C_n$ in t Schritten erreichbar ist. Es gibt eine Prozedur $COUNT(c, t)$, die bei der Eingabe c der Anzahl der in t erreichbaren Konfigurationen die Anzahl der Erreichbaren Konfigurationen in $t + 1$ berechnet. Schließlich gibt es ein Entscheidungsverfahren für das Komplement.
- **Wie zählt man die Konfigurationen auf?**
  Jede Konfiguration kann mit Hilfe von $S(|x|)$ Zeichen beschrieben werden. Daher kann man sie lexikographisch aufzählen.
- **Wie ist die Testprozedur $REACH$ definiert?**
  Sei r die (Schätzung für die) Anzahl der erreichbaren Konfigurationen. Für jede Konfigurationen D (in einer Lexikographischen Aufzählung) rät man nichtdeterministisch eine Konfiguration, die in t Schritten erreichbar ist und prüft, falls D erreicht wurde ob gilt $C_i = D$. Ist das der Fall, so bricht man mit einer Antwort "ja" ab. Ist das nicht der Fall so erhöht man einen Zähler. Hat man am Ende alle r Konfigurationen abgezählt, so ist die Antwort "nein", sonst "?".
- **Wie ist die Testprozedur $COUNT$ definiert?**
  Man prüft für jede Konfiguration mit Hilfe von $REACH$, ob eine der Vorgängerkonfigurationen in $t-1$ Schritten erreichbar ist. Ist dies der Fall wird ein Zähler hochgezählt. Liefert $REACH$ ein Misßerfolg "?", so bricht man ab mit "?". Am Ende der Prozedur gibt man den Zähler zurück.
- **Wie sieht das Entscheidungsverfahren für das Komplement aus?**
  Man berechne mit $COUNT$ die Anzahl der erreichbaren Konfigurationen für das Wort x durch Zählen von 0 bis zur Zeitschranke t. Prüfe für alle akzeptierenden Konfigurationen mit $REACH$, ob sie bei x erreicht werden können. Falls ja und gib $x \notin L$ aus.
- **Was macht man, wenn die Schranke t am Anfang nicht vorhanden ist?**
  Man beginnt mit einem kleinen Wert von t und verdoppelt den solange, bis mit grösserer Schranke keine neuen Konfigurationen erreichbar sind. Das wird durch einen zusätzlichen Aufruf von $REACH$ realisiert.

5.2 NL-Vollständige Probleme

- **Was bedeutet $\leq_{log}$?**
  Es gibt eine log-Bandberechenbare Reduktion $x \in L$ gdw. $f(x) \in L'$.
- **Welche Klassen sind unter $\leq_{\log}$ abgeschlossen?**
  L, NL, P, NP, PSPACE, REK, RE.
- **Welche Klassen sind nicht $\leq_{\log}$ abgeschlossen?**
  XSPACE(n^i), XTIME(n^i)
- **Wie kann man obige Aussage begründen?**
  Aus Translation und Reduktion ergibt sich $DSPACE(n^i) \subseteq DSPACE(n)$ wegen $\log n^i \in O(\log n)$.
- **Welche Sprache kann man als schwierigste in L bezeichnen?**
  Erreichbarkeit im gerichteten Baum.
- **Welche vollständigen Sprachen kennen Sie in NL?**
  GAP (Grapherreichbarkteit), kürzeste Wege, Zyklus Bestimmung.
- **Wie zeigt man, dass GAP in NL liegt?**
  Rate eine Kantenfolge und prüfe, ob die die knoten Verbindet. Der Speicherplatz reicht zum Kodieren der Knotennummern v_i, v_{i+1}.
- **Wie kann man das Wortproblem für NL auf GAP reduzieren?**
  Man konstruiere den Berechnungsgraphen einer log Bandbeschränkten NTM von L mit einer eindeutigen Halteknfiguration. Dann entspricht GAP der Erreichbarkeit.

6 P: Zeitkomplexität

6.1 Effizient lösbare Probleme

- **Nennen Sie einige effiziente Probleme?**
  DFS, BFS, Sortieren, Primzahlen.

6.2 Eigenschaften der Klasse P

- **Welche Eigenschaften hat P?**
  Eichte Hierarchie bei $DTIME(n^i)$, $NL \subseteq P$, $CFL \subseteq P$, Abgeschlossen unter log-Band Reduktionen.
- **Unter welchen Eigenschaften is P abgeschlossen?**
  Schnitt, Vereinigung, Differenz, Konkatenation.
- **Unter welchen Eigenschaften is P nicht abgeschlossen?**
  Unter löschendem Homomorphismus.

6.3 Charakterisierungen von P

- **Wie kann man P charakterisieren?**
  Durch log-Band AuxPDA. Logarithmisch beschränkte Arbeitsbänder und freier Keller.
- **Was gilt für (nicht-) deterministische AuxPDA?**
  DTIME($c^{S(n)}$) = DAuxPDA(S(n)) = NAuxPDA(S(n)).
- **Was kann man daraus schliessen?**
  P = AuxPDA($\log n$)
- **Wo gilt der Zusammenhang ND = D sonst?**
  Reguläre Sprachen, AuxPDA für P, TM's hoher Komplexität, Haltende TM's.
- **Was ist eine *bewegungsuniforme DTM*?**
  Die Bewegungen des Kopfes sind nur von der Länge des Wortes n abhängig.
- **Wie zeigt man den Simulationssatz?**
  Man betrachtet eine Funktion $Test(q, Z, t) = true$ gdw. die Parameter q, Z, t stimmen. Simulation mit Baumartiger rekursion, weil der Test in zwei Teile zerfällt: Zum letzten Zeitpunkt ein gab es einen Zustandsübergang nach q und zur letzten Besuchszeit Zeichen Z geschrieben. Berechnung, wann die Maschine zum letzten Mal einen Platz besucht hat kann in $\log n$-Speicher berechnet werden. Analog reicht logarithmischer Speicher für das Abspeichern der Testdaten.
- **Wie simuliert man AuxPDA auf einer DTM?**
  Man konstruiert den Erreichbarkeitsgraphen durch Einfügen neuer Kanten. Knoten sind Konfigurationen nur mit einem Kellersymbol.

6.4 Spezialisierungen

- **Welche typen von Kellerautomaten liegen in P?**
  PDA, DPDA, 2-Keller PDA.

6.5 P-Vollständige Probleme

- **Kennen Sie P-vollständige Probleme?**
  AGAP (Markierung der Quelle, wenn alle Senken markiert sind), CVP (Erfüllbarkeit von booleschen Termen), CFG-$\emptyset$ (Leerheitsproblem bei CFG/Kellerautomaten).
- **Wie kann man die Vollständigkeit von CVP zeigen?**
  Chomsky-Normalform. AB ist ein "und" und die Produktionen werden über ein "oder" verbunden.

6.6 Gegenüberstellung

- **Welche typischen Probleme hat L?**
  Eindeutige Pfade in einem gerichteten Graph.
- **Welche typischen Probleme hat NL?**
  Pfade in einem gerichteten Graph. GAP. Alle inneren Knoten sind $\vee$-Knoten.
- **Welche typischen Probleme hat $coNL$?**
  Alle inneren Knoten sind $\wedge$-Knoten.
- **Warum gilt $NL = coNL$?**
  Immerman.
- **Welche typischen Probleme hat P?**
  AGAP, CVP.

7 Tabellen

7.1 Hierarchien

Aussage	Vor.	Beweis	Stichwörter
DTIME(f) $\subseteq$ NTIME(f) DSPACE(f) $\subseteq$ NSPACE(f)	-	Jede DTM ist NTM	Simulation
XTIME(f) $\subseteq$ XSPACE(f)	-	TM braucht für k Schritte max k Platz	Simulation
NSPACE(f) $\subseteq$ DTIME($2^{O(f)}$)	$f \geq \log n$	Simulation von NTM durch DTM mit Zeitbeschränkung $O(nc^f)$. Breitensuche auf dem Berechnungsgraph.	Simulation, Breitensuche
NL $\subseteq$ P, PSPACE $\subseteq$ EXPTIME	-	Korrolar von oben	-
NTIME(f) $\subseteq$ DSPACE(f)	-	Tiefensuche auf einem Berechnungsbaum. Durchprobieren aller Befehlskombinationen $\leq f$	Simulation, Tiefensuche
NP $\subseteq$ PSPACE	-	Korrolar von oben	-
Savitch: NSPACE(s) $\subseteq$ DSPACE($s \log t$)	$s \geq \log n$	Divide&Conquer auf dem Berechnungsgraphen, Test-Funktion(K,K',t). Wort wird akzeptiert, falls Test(S,E,t) gelingt.	Simulation, Berechnungsgraph, Divide&Counquer
NPSPACE = PSPACE	-	Korrolar von oben	-
Platzhierarchie: SPACE(s_1) $\subset$ SPACE(s_2)	$s_1 << s_2$, $s_2 \geq \log n$, s_2 bandkonstruierbar	Konstruktion der Sprache durch alle Wörter $w = 1^* <$ $M >$ mit $w \notin L(M)$, sodass auf s_2 Speicherplatz M verwirft w.	Simulation, Diagonalisierung
Zeithierarchie: DTIME(f) $\subset$ DTIME(g)	$f \log f << g$	Diagonalsprache wird auf einer 2-Band DTM simuliert in der Zeit $T_2(n)$. Für genügend große n kann M_2 die Berechnung durchführen.	Simulation, Diagonalsprache, 2-Band, Uhr
DTIME(n^i) $\subset$ DTIME(n^{i+1})	-	Korrolar von oben	-
DTIME(n^i) $\subset$ P	-	Korrolar von oben	-
Transloationstechnik: $P_f(L) \in DTIME(O(g))$ gdw. $L \in DTIME(O(g \circ f))$.	f Bandkonstruierbar, $f \geq n$, $f \geq n$	Simulation auf von Wörtern von L.	Simulation, Padding, Uhr.
NSPACE(n^p) $\subset$ NSPACE(n^{p+1})	-	Korrolar von oben	-
DTIME(n) $\subset$ DTIME($n \log n$)	-	Korrolar von oben	-
Immerman: NSPACE(f) = coNSPACE(f)	$f \geq \log n$, f schwach konstruierbar	Induktives Zählen	Simulation, Induktives Zählen

7.2 Komplexitätsklassen

Klasse	Definition	Gleichwertig	Enthalten	Typisches Problem
REG	DSPACE(1)	NSPACE(1), DFA, NFA, REX		Statecharts, Protokolle
L	DSPACE(log)		NL	Eindeutige erreichbarkeit in Graphen
NL	NSPACE(log)	coNSPACE(log)	P	GAP
P	DTIME(poly)	coP	NP	Boolesche QFF
NP	NTIME(poly)		PSPACE	SAT
PSPACE	DSPACE(poly)	NSPACE	EXPTIME	QF
EXPTIME	DTIME(2^{poly})		RE	QF

Index