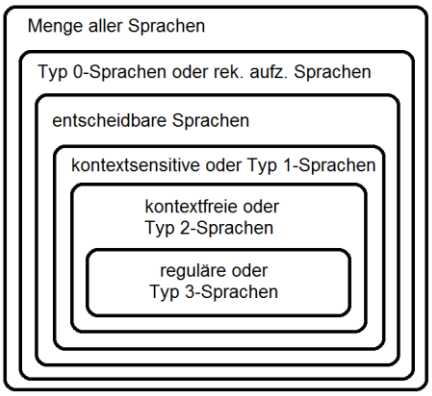
Kapitel 1 – Automatentheorie und Formale Sprachen

1.1.1 Grammatiken

- 4-Tupel G = (V, Σ, P, S)
  Bedingungen: Σ, V endlich und schneiden sich nicht
- kein eindeutiger Ableitungsbaum für ein Wort (in der Regel)
- das Ableiten ist kein eindeutiger, deterministischer Prozess
- reflexiv-transitive Hülle =>*: mehrere Ableitungen
- Linksableitung/Rechtsableitung



1.1.2 Chomsky-Hierarchie

- Typ 0: kleine Einschränkungen für die Grammatik
- Typ 1: (kontextsensitiv) für $w_1 \rightarrow w_2$ ist $|w_1| \leq |w_2|$
- Typ 2: (kontextfrei) für $w_1 \in V$
- Typ 3: (regulär) für $w_2 \in \Sigma \cup \Sigma V$
- Typ 3 ≤ Typ 2 ≤ Typ 1 ≤ Typ 0
- man kann jeder Grammatik mit $\varepsilon \notin L(G)$ eine Grammatik G' ohne ε-Regeln zuordnen
- alle Sprachen von Typ 1,2,3 sind entscheidbar, Typ 0 ist semi-entscheidbar
- Backus-Naur-Form: $A \rightarrow b_1 \mid b_2 \mid \ldots \mid b_n$; $A \rightarrow a[b]c$; $A \rightarrow a\{b\}c$

1.2.1 Endliche Automaten

- deterministischer Automat (DFA): 5-Tupel $M = (Z, \Sigma, \delta, z_0, E)$
- nichtdeterministischer Automat (NFA): 5-Tupel $M = (Z, \Sigma, \delta, S, E)$
- Jede von einem NFA akzeptierbare Sprache ist auch durch einen DFA akzeptierbar. (Scott)
  (Potenzalgorithmus, NFA mit k Zuständen, dann benötigt DFA maximal 2^k Zustände)
- Für Jede reguläre Grammatik G gibt es einen NFA M mit L(G) = T(M).

1.2.3 Reguläre Ausdrücke

- $a \in \Sigma, \varepsilon, (\alpha|\beta), (\alpha)^*$ mit $\alpha, \beta \in \Sigma$ sind reguläre Ausdrücke
- Die Menge der durch reguläre Ausdrücke beschreibbaren Sprachen ist genau die Menge der regulären Sprachen. (Kleene)

1.2.4 Das Pumping Lemma I

- Sei L eine Reguläre Sprache. Dann gibt es eine Zahl n, so dass sich alle Wörter $x \in L$ mit $|x| \geq n$ zerlegen lassen in x = uvw, so dass folgende Eigenschaften erfüllt sind:
  - $|v| \geq 1$
  - $|uv| \leq n$
  - für alle i = 0, 1, 2, … gilt: $uv^iw \in L$
- Angenommen: L wäre regulär
  „Dann gibt es eine Zahl $n \in N$, so dass sich alle Wörter $x \in L$ wie m Pumping-Lemma zerlegen lassen." Betrachten für ein beliebiges $n \in N$, das Wort $x = a^nb^n$ mit $|x|=2n$
  Wegen 2. besteht uv und somit auch v nur aus a's.
  Wegen 3 wäre dann auch $uv^2w = a^{n+|v|}b^n \notin L$
  Wegen 1: $|v| \geq 1 \Rightarrow a^{n+|v|}b^n \notin L$ (Widerspruch)
  Also die Sprache L ist nicht regulär. ▪

1.2.5 Äquivalenzrelationen und Minimalautomaten

- Eine Sprache L ist genau dann regulär, wenn der index von R_L endlich ist. (Myhill, Nerode)
- Minimalisierungsalgorithmus
  - markiere alle Paare mit Endzuständen und nicht Endzuständen
  - markiere alle Paare bei denen es ein Übergang in ein bisher markiertes Paar gibt
  - nicht markierte Paare können jeweils verschmolzen werden

1.3 Normalformen

- Zu jeder kontextfreien Grammatik G mit $\varepsilon \notin L(G)$ gibt es eine Chomsky-Normalform-Grammatik G‘ mit L(G) = L(G‘)
- Chomsky-Normalform: A → BC, A →a
- Greibach-Normalform: A →aBC…D|bBCE

1.3.2 Das Pumping Lemma II (kontextfrei)

- Sei L eine kontextfreie Sprache. Dann gibt es eine Zahl $n \in N$, so dass sich alle Wörter $z \in L$ mit $|z| \geq n$ zerlegen lassen in z = uvwxy mit folgenden Eigenschaften:
  - $|vx| \geq 1$
  - $|vwx| \leq n$
  - für alle $i \geq 0$ gilt: $uv^iwx^iy \in L$
- Jede kontextfreie Sprache über einem einelementigen Alphabet ist bereits regulär.

1.3.4 Der CYK-Algorithmus

- effizienter als der herkömmliche Algorithmus mit exponentiellen Aufwand
- nach dem Anfangsbuchstaben benannt Cocke, Younger und Kasami
- CYK gibt nur an ob ein Wort in der Sprache ist oder nicht

1.3.5 Kellerautomaten

- 6-Tupel $M = (Z, \Sigma, \Gamma, \delta, z_0, \#)$
- Wort wird erkannt, falls alle Buchstaben eingelesen wurden und der Stack leer ist
- nur ein Startzustand (durch spontane Übergänge sind aber theoretisch mehrere möglich)
- Eine Sprache L ist kontextfrei genau dann, wenn L von einem nichtdeterministischen Kellerautomaten erkannt wird.

1.4 Kontextsensitive und Typ 0-Sprachen

- Für jede Typ 1-Grammatik G mit $\varepsilon \notin L(G)$ gibt es eine Grammatik G‘ in Kuroda-Normalform mit L(G) = L(G‘). (A → a, A → B, A → BC, AB → CD)
- Turingmaschine
  - 7-Tupel $M = (Z, \Sigma, \Gamma, \delta, z_0, \square, E)$
  - den Begriff der Berechenbarkeit, des effektiven Verfahrens exakt zu beschreiben
  - unendliches Band (oder auch Mehrband)
- Die von linear beschränkten, nichtdeterministischen Turingmaschinen (LBAs) akzeptierbaren Sprachen sind genau die kontextsensitiven (Typ 1) Sprachen. (Kuroda)
- Die durch allgemeine TM akzeptierbaren Sprachen sind genau die Typ 0-Sprachen.
- LBA-Problem: Äquivalenz zwischen deterministischen TMs.
- Die Klasse der Kontextsensitiven (also Typ 1-) Sprachen ist unter Komplementbildung abgeschlossen. (Immerman, Szelepcsényi)

1.5 Tabellarischer Überblick

Beschreibungsmittel

Typ 3	reguläre Grammatik DFA/NFA regulärer Ausdruck
Det. kf	LR(k)-Grammatik deterministischer Kellerauttomat (DPDA)
Typ 2	kontextfreie Grammatik Kellerautomat (PDA)
Typ 1	kontextsensitive Grammatik linear beschränkter Automat (LBA)
Typ 0	Typ 0 – Grammatik Turingmaschine (TM)

Determinismus und Nichtdeterminismus

Nichtdet. Automat	**Determ. Automat**	**äquivalent?**
NFA	DFA	ja
PDA	DPDA	nein
LBA	DLBA	?
TM	DTM	ja

Abschlusseigenschaften

	Schnitt	**Vereinigung**	**Komplement**	**Produkt**	**Stern**
Tpy 3	ja	ja	ja	ja	ja
Det. kf.	nein	nein	ja	nein	nein
Typ 2	nein	ja	nein	ja	ja
Typ 1	ja	ja	ja	ja	ja
Typ 0	ja	ja	nein	ja	ja

Entscheidbarkeit

	Wort-problem	**Leerheits-problem**	**Äquivalenz-problem**	**Schnitt-problem**
Typ 3	ja	ja	ja	ja
Det. kf.	ja	ja	ja	nein
Typ 2	ja	ja	nein	nein
Typ 1	ja	nein	nein	nein
Typ 0	nein	nein	nein	nein

Wortproblem

Typ 3 (DFA gegeben)	lineare Komplexität
Det. kf.	lineare Komplexität
Typ 2 (CNF gegeben)	Komplexität: $O(n^3)$
Typ 1	exponentielle Komplexität NP-hart
Typ 0	unlösbar

Kapitel 2 – Berechenbarkeitstheorie

2.1 Intuitiver Berechenbarkeitsbegriff und Churchsche These

- Die durch die formale Definition der Turing-Berechenbarkeit (Äquivalent: WHILE-Berechenbarkeit, GOTO-Berechenbarkeit, μ-Rekursivität) erfasste Klasse von Funktionen stimmt genau mit der Klasse der im intuitiven Sinne berechenbaren Funktionen überein.
- Beispiel:
  - f(n) = 1 falls n ein Anfangsabschnitt der Dezimalbruchentwicklung (DBE) von π ist sonst 0 *(berechenbar)*
  - g(n) = 1 falls n irgendwo in der DBE von π vorkommt sonst 0 *(nicht berechenbar)*
  - h(n) = 1 falls in der DBE von π irgendwo mind. n-mal hintereinander eine 7 vorkommt sonst 0 *(berechenbar)*
  - i(n) = 1 falls das LBA-Problem eine positive Lösung hat sonst 0 *(berechenbar)*

2.2 Turing-Berechenbarkeit

- Zu jeder Mehrband-Turingmaschine M gibt es eine (Einband-) Turingmaschine M‘ mit T(M) = T(M‘) bzw. so, dass M‘ dieselbe Funktion berechnet wie M.

2.3 LOOP-, WHILE- und GOTO-Berechenbarkeit

- Eine Funktion f: $N^k \rightarrow N$ heißt LOOP-berechenbar, falls es ein LOOP-Programm P gibt, das f in dem Sinne berechnet, dass P, gestartet mit $n_1, \ldots, n_k$ in den Variablen $x_1, \ldots, x_k$ (und 0 in den restlichen Variablen) stoppt mit dem Wert $f(n_1, \ldots, n_k)$ in der Variablen x_0.

```
IF x = 0 THEN A END
y := 1;
LOOP x DO y:= 0 END;
LOOP y DO A END
```

- LOOP-berechenbare Funktionen sind totale Funktionen
- Addition, Multiplikation, Div/Mod ist LOOP-berechenbar
- Eine Funktion f: $N^k \rightarrow N$ heißt WHILE-berechenbar, falls es ein WHILE-Programm P gibt, das f in dem Sinne berechnet, dass P, gestartet mit $n_1, \ldots, n_k$ in den Variablen $x_1, \ldots, x_k$ (und 0 in den restlichen Variablen) stoppt mit dem Wert $f(n_1, \ldots, n_k)$ in der Variablen x_0 – sofern $f(n_1, \ldots, n_k)$ definiert ist, ansonsten stoppt P nicht.

```
LOOP x DO P END ⇔
y := x;
WHILE y ≠ 0 DO y := y – 1; P END
```

- Das i-te Band der Turingmaschine entspricht der x_i-ten Variablen
- Turingmaschinen können WHILE-Programme simulieren. Das heißt, jede WHILE-berechenbare Funktion ist auch Turing-berechenbar.
- Jedes WHILE-Programm kann durch ein GOTO-Programm simuliert werden. Das heißt, jede WHILE-berechenbare Funktion ist auch GOTO-berechenbar.

```
WHILE x_i ≠ 0 DO P END
M1:   IF x_i = 0 THEN GOTO M2;
      P;
      GOTO M1;
M2:   ...
```

- Jedes GOTO-Programm kann durch ein WHILE-Programm (mit nur einer WHILE-Schleife) simuliert werden. Also ist jede GOTO-berechenbar Funktion auch WHILE-berechenbar.
- Jede WHILE-berechenbare Funktion kann durch ein WHILE-Programm mit nur einer WHILE-Schleife berechnet werden. (Kleensche Normalform für WHILE-Programme)
- Die LOOP-berechenbaren Funktionen sind eine echte Teilmenge der WHILE-berechenbaren.
- GOTO-Programme können Turingmaschinen simulieren. Also ist jede Turing-berechenbare Funktion auch GOTO-berechenbar.

2.4 Primitiv rekursiv und μ-rekursive Funktionen

- Die Klasse der primitiv rekursiven Funktionen (auf N) ist induktiv wie folgt definiert:
  - Alle konstanten Funkionen, identischen Abbildungen sind primitiv rekursiv.
  - Die Nachfolgerfunktion s(n) = n + 1 auf N ist primitiv rekursiv.

  } Basisfunktionen
  - Jede Funktion, die Komposition/Einsetzung von primitiv rekursiven Funktionen

  - Jede Funktion, die durch primitive Rekursion aus primitiv rekursiven Funktionen entsteht, ist primitiv rekursiv. f(0, …) = g(…); f(n+1, …) = h(f(n,…),…)
- Die Klasse der primitiv rekursiven Funktionen stimmt genau mit der Klasse der LOOP-berechenbaren Funktionen überein.
- Die Klasse der µ-rekursiven Funktionen stimmt genau mit der Klasse der WH Turing-) berechenbaren Funktionen überein.
- Für jede n-stellige µ-rek. Funktionen f gibt es zwei (n + 1)-stellige, primitiv rek. Funktionen p und q, so dass sich f darstellen lässt als $f(x_1,\dots, x_n) = p(x_1,\dots, x_n, \mu q(x_1,\dots, x_n))$ hierbei ist µq die durch Anwendung des µ-Operators auf q entstehende (n-stellige) Funktion. (Kleene)

add(0,x) = x
(identische Abbildung)
add(n + 1, x) = s(add(n,x))
(s ist die Nachfolgerfunktion)

2.5 Die Ackermannfunktion

- ist intuitiv berechenbar (WHILE), nicht LOOP-berechenbar
- Es gibt totale, WHILE-berechenbare Funktionen die nicht LOOP-berechenbar sind

$a(0,y) = y + 1$
$a(x,0) = a(x-1, 1)$ $\quad x > 0$
$a(x,y) = a(x-1, a(x, y-1))$ $\quad x,y > 0$

Lemma A: $y < a(x,y)$
B: $a(x,y) < a(x,y+1)$
C: $a(x,y+1) \leq a(x+1,y)$
D: $a(x,y) < a(x+1,y)$
E: Für jedes LOOP-Programm P gibt e seine Konstante k, so dass für alle n gilt: $f_P(n) < a(k,n)$.

2.6 Halteproblem, Unentscheidbarkeit, Reduzierbarkeit

- Eine Menge $A \leq \Sigma^*$ heißt entscheidbar, falls die charakteristische Funktion von A, nämlich $x_A: \Sigma^* \rightarrow \{0,1\}$, berechenbar ist.
- Eine Menge $A \leq \Sigma^*$ heißt semi-entscheidbar, falls die „halbe“ charakteristische Funktion von A, nämlich $x_A: \Sigma^* \rightarrow \{0,1\}$, berechenbar ist.
- Eine Sprache A ist entscheidbar, genau dann wenn sowohl A als auch A^c semi-entscheidbar sind.
  - A ist rekursiv aufzählbar ⇔ A ist semi-entscheidbar ⇔ A ist vom Typ0 ⇔ A = T(M) für eine Turingmaschine M ⇔ x'_A ist Turing-, WHILE-, GOTO-berechenbar ⇔ A ist Definitionsbereich einer berechenbaren Funktion ⇔ A ist Wertebereich einer berechenbaren Funktion
- Das (spezielle) Halteproblem (auf leerem Band) ist nicht entscheidbar
- Seien $A \leq \Sigma^*$ und $B \leq \Gamma^*$ Sprachen. Dann heißt A auf B reduzierbar – symbolisch mit $A \leq B$ bezeichnet – falls es eine totale und berechenbare Funktino $f: \Sigma^* \rightarrow \Gamma^*$ gibt, so dass für alle $x \in \Sigma^*$ gilt: $x \in A \Leftrightarrow f(x) \in B$.
- Falls $A \leq B$ und B entscheidbar (bzw. semi-ent.) ist, so ist auch A entscheidbar (bzw. semi-ent.)
- Sei R die Klasse aller Turing-berechenbaren Funktionen. Sei S eine beliebige Teilmenge hiervon (mit Ausnahme von $S = \{\}$ und $S = R$). Dann ist die Sprache $C(S) = \{w \mid$ die von M_w berechnete Funktion liegt in S} unentscheidbar.

$x_1 = 1$ $\quad x_2 = 10$ $\quad x_3 = 011$
$y_1 = 101$ $\quad y_2 = 00$ $\quad y_3 = 11$
Lösung: (1, 3, 2, 3)
$x_1 x_3 x_2 x_3 = 101110011 = y_1 y_3 y_2 y_3$

2.7 Das Postsche Korrespondenzproblem

- Das Postsche Korrespondenzproblem PCP ist unentscheidbar.
- Das PCP ist bereits unentscheidbar, wenn man sich auf das Alphabet {0, 1} beschränkt.
- Das Postsche Korrespondenzproblem PCP ist unentscheidbar. (Lemma $H \leq MPCP \leq PCP$)
- Das PCP ist bereits unentscheidbar, wenn man sich auf das Alphabet {0,1} beschränkt.

2.8 Unentscheidbare Grammatik-Probleme

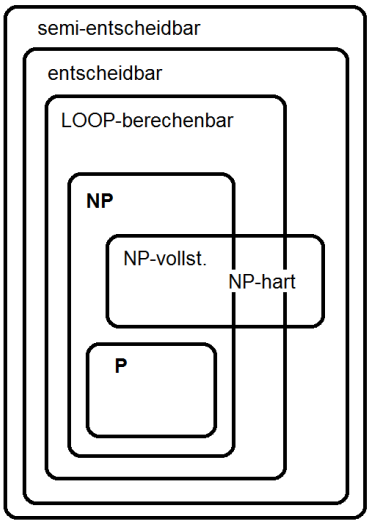
- Gegeben zwei kontextfreie Grammatiken G_1, G_2, so sind folgende Fragestellungen unentscheidbar: Ist $L(G_1) \cap L(G_2) = \{\}$? Ist $|L(G_1) \cap L(G_2)| = \infty$? Ist $L(G_1) \cap L(G_2)$ kontextfrei? Ist $L(G_1)$ ist Teilmenge von $L(G_2)$? Ist $L(G_1) = L(G_2)$?

- Wenn zwei deterministisch kontextfreie Sprachen L_1, L_2 gegeben sind, so sind folgende Fragestellungen unentscheidbar: Ist $L_1 \cap L_2 = \{\}$? Ist $|L_1 \cap L_2| = \infty$? Ist $L_1 \cap L_2$ kontextfrei? Ist L_1 ist Teilmenge von L_2?
- Gegeben eine kontextfreie Grammtik G, dann sind folgende Fragestellungen unentscheidbar: Ist G mehrdeutig? Ist $L(G)^c$ kontextfrei? Ist L(G) regulär? Ist L(G) deterministisch kontextfrei?
- Gegeben eine kon. Sprache L_1 und eine reg. Sprache L_2, so ist es unent. festzustellen, ob $L_1 = L_2$ gilt.
- Das Leerheitsproblem und das Endlichkeitsproblem für Typ 1-Sprachen sind nicht ent.

Kapitel 3 – Komplexitätstheorie

3.1 Komplexitätklassen und P-NP-Problem

- Sei f: N → N eine Funktion. Die Klasse TIME(f(n)) besteht aus allen Sprachen A, für die es eine deterministische Mehrband-Turingmaschine M gibt mit A = T(M) und $time_M(x) \leq f(|x|)$. Hierbei bedeutet $time_M$: Σ* → N die Anzahl der Rechenschritte von M bei Eingabe x.
- Eine f(n) rechenzeit-beschränkte Mehrband-TM kann durch eine $O(f^2(n))$ rechenzeit-beschränkte Einband-TM simuliert werden.
- Die Komplexitätklasse P ist wie folgt definiert:
  P = { A | es gibt eine Turingmaschine M und ein Polynom p mit T(M) = A und $time_M(x) \leq p(|x|)$}
- Die Klasse P, ebenso wie $TIME(2^n)$ oder gar $TIME(2^{\cdots^2})$ sind LOOP-berechenbar! (funktioniert allg: solange die Komplexität des WHILE-Programms LOOP-berechenbar ist)
- Für nichtdeterministische Turingmaschinen M sei
  $ntime_M(x)$ = {min [Länge einer akzeptierenden Rechnung von M auf x] falls x ϵ T(M), sonst 0
  Sei f: N→N eine Funktion. Die Klasse NTIME(f(n)) besteht aus allen Sprachen A, für die es eine nichtdet. Mehrband-TM M gibt mit A = T(M) und $ntime_M(x) \leq f(|x|)$
- Es ist klar dass P Teilmenge von NP, allerdings ist bis heute ungeklärt ob P ≠ NP.



3.2 NP-Vollständigkeit

- Seien A Teilmenge von Σ* und B von Γ* Sprachen. Dann heißt A auf B polynomial reduzierbar – symbolisch mit $A \leq_p B$ bezeichnet – falls es eine totale und mit polynomialer Komplexität berechenbare Funktion f: Σ* → Γ* gibt, so dass für alle x ϵ Σ* gilt: x ϵ A ⇔ f(x) ϵ B
- Falls $A \leq_p B$ und B ϵ P, so ist auch A ϵ P. (ebenso mit NP) (umgangssprachlich: A leichter als B)
- Eine Sprache A heißt NP-hart, falls für alle Sprachen L ϵ NP gilt: $L \leq_p A$. Eine Sprache A heißt NP-vollständig, falls A NP hart ist und A ϵ NP gilt.
- Sei A NP-vollständig. Dann gilt: A ϵ P ⇔ P = NP (Millennium-Problem, man glaubt P ≠ NP)
- Das Erfüllbarkeitsproblem der Aussagenlogik (SAT) ist NP-vollständig.