Philipps-Universität Marburg
Fachbereich 12 Informatik
VL Theoretische Informatik
Prof. Rita Loogen
Wintersemester 2012/13
Zusammengefasst von: Dirk Winkel

Zusammenfassung Skript

Inhaltsverzeichnis

1 Einführung

- Ein *Alphabet* Σ ist eine Menge von *Zeichen*, *Buchstaben* oder *Symbolen*.
- Eine *formale Sprache* L über einem Alphabet Σ ist eine Teilmenge von Σ^*:

$$L \subseteq \Sigma^* \Leftrightarrow L \in \mathfrak{P}(\Sigma^*)$$

- Im allgemeinen sind Sprachen unendliche Mengen, die mit Hilfe von Grammatiken oder Automaten endlich beschrieben werden können.
- Beispiel für eine induktive Definition: $\varepsilon \in \Sigma^*$; $w \in \Sigma^*, a \in \Sigma^* \Rightarrow aw \in \Sigma^*$
- *Konkatenation* bezeichnet die Verkettung von Wörtern
- *Spiegelung* dreht das Wort um: $\overleftarrow{\varepsilon} := \varepsilon$; $\overleftarrow{wa} := a\overleftarrow{w}$
- *Iteration* oder *Potenz* bezeichnet die Vervielfachung eines Wortes: $w^0 := \varepsilon$; $w^{n+1} := w^n w$
- *Mengenoperatoren* $\cup, \cap, \setminus$ sowie *Komplexprodukt* $\cdot$, *Potenz* ^{n} und *Sternoperator* * sind für Sprachen wie für Mengen üblich definiert.
- *Binäre Relationen* ϱ setzten zwei Elemente einer Sprache L ind Relation zu einander: $\varrho \subseteq L \times L$ (vgl. "Funktion").

Teil I
Automatentheorie und Formale Sprachen

2 Grammatiken und die Chomsky-Hierarchie

- *Grammatiken* erzeugen formale Sprachen (z.B. Backus-Naur-Form, hier aber formal wir folgt). Darin kommen vor:
  - *Nonterminalsymbole* N (Großbuchstaben $A, B, ...$), die weiter aufgelöst werden können zu
  - *Terminalsymbole* $\in \Sigma = \{a, b, ...\}$ (Kleinbuchstaben $a, b, ...$ oder Ziffern) (nicht weiter auflösbar)
  - *Startsymbol* S wo die Ableitung von Terminalwörtern (stetiges Ersetzen von Noterminalen zu Terminalen) beginnt
  - *Regeln* oder *Produktionen* P legen fest wie die Ableitungen zu bilden sind. Dabei sind links Nonterminale, rechts Nonterminale oder Terminale, dazwischen ein "$\rightarrow$" (Backus-Naur-Form: "::="). Bei mehreren Regeln zu einer gleichen linken Seite werden diese durch "|" getrennt.
- Eine *Grammatik* G ist also ein Quadrupel: $G = (N, \Sigma, P, S)$
- *Satzformen* $\alpha, \beta, ...$ sind $\in N \times \Sigma$ Konkatenationen von Terminalen und Nonterminalen
- Eine *Ableitungsrelation* $\Rightarrow_G$ bezeichnet eine Ableitung $\alpha \Rightarrow_G \alpha'$ die in G existiert. $\Rightarrow_G^*$ ist dessen reflexive, transitive Hülle, also die Menge aller Konkatenationen von $\Rightarrow_G$.
- Die *von G erzeugte Sprache* ist $L(G) := \{w \in \Sigma^* | S \Rightarrow_G^* w\}$ die Menge der von G erzeugten Wörter.
- für *äquivalente Grammatiken* gilt $L(G_1) = L(G_2)$

***Chomsky-Hierarchie**: Typen der Sprachen:*

Typ0 **(rekursiv aufzählbar)** Keine Einschränkung, also jede Grammatik
$\mathcal{L}_0(\Sigma) = L(\Sigma, TM) = L(\Sigma, DTM)$

Typ1 **(kontextsensitiv)** Für jede Produktion $\alpha_1 \rightarrow \alpha_2$ gilt $|\alpha_1| \leq |\alpha_2|$ oder $\alpha_1 = S \wedge \alpha_2 = \varepsilon$ (und, falls $S \rightarrow \varepsilon \in P$, S nie auf rechter Regelseite)
$\mathcal{L}_1(\Sigma) = L(\Sigma, LBA)? \supset ? L(\Sigma, DLBA)$

Typ2 **(kontextfrei)** Die linke Regelseite ist immer genau ein Noterminalsymbol: $A \rightarrow \alpha$
$\mathcal{L}_2(\Sigma) = L(\Sigma, PDA) \supsetneq L(\Sigma, DPDA)$

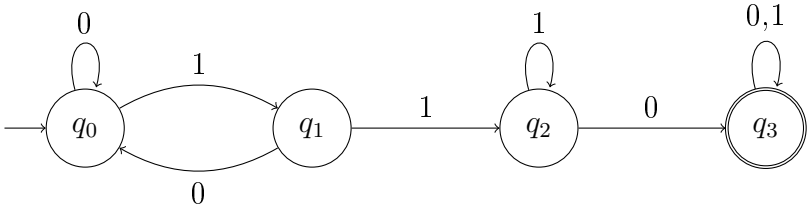
Typ3 **(regulär)** einseitig linear (z.B. *rechtslinear*: $A \rightarrow wB$ oder $A \rightarrow w$)
$\mathcal{L}_3(\Sigma) = L(\Sigma, NFA) = L(\Sigma, DFA) = Reg(\Sigma)$

- Die Menge aller Sprachen, die durch Grammatiken des Typs i erzeugt werden, bezeichnet man als $\mathcal{L}_i$ [Skript: schnörkeligeres L, nicht LaTeXvorhanden :-(]

- $\mathcal{L}_3(\Sigma) \subsetneqq \mathcal{L}_2(\Sigma) \subsetneqq \mathcal{L}_1(\Sigma) \subsetneqq \mathcal{L}_0(\Sigma)$ $(\forall |\Sigma| \geq 2)$
- *ε-freie Grammatiken* haben auf der rechten Regelseite kein ε (oder $S \to \varepsilon$).
  Jede kontextfreie Grammatik kann in eine ε-freie kontextfreie Grammatik übersetzt werden.
- *Abzählbar* ist eine Sprache, falls eine bijektive Abbildung $f : \mathbb{N} \to L$ existiert.
- Das *Wortproblem* fragt, ob ein Wort $w \in L$ ist. Es ist für Typ1-Sprachen entscheidbar.

3 Endliche Automaten

- *Darstellung*: Zustände in Kreisen, Eingangszustand mit einfachem Pfeil, Endzustände in Doppelkreisen, Zustandsübergänge mit Pfeilen (beschriftet)



- Ein ***DFA (deterministischer endlicher Automat)*** ist ein Quintupel $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ mit:
  - Q einer Menge von Zuständen q_i
  - Σ einem Eingabealphabet
  - $\delta(q, a) = q' : Q \times \Sigma \to Q$ eine Transitions- oder Übergangsfunktion, die einen Zustand in einen anderen überführt
  - $q_0 \in Q$ dem Startzustand
  - $F \subset Q$ der Endzustandsmenge
- DFA(Σ) ist die Menge aller DFAs, $\mathcal{L}(\Sigma, \text{DFA})$ ist die Menge aller von DFAs erkennbaren Sprachen über Σ.
- $\overline{\delta}(q, w)$ ist die *Erweiterte Transitionsfunktion*, also eine Verkettung von Transitionen.
- Ein ***NFA (nicht-deterministischer endlicher Automat)*** erlaubt im Gegensatz zum DFA zu einer Transition $\delta(q, a)$ mehrere Zustände, in die die Transition führen kann, also $\delta(q, a) = \{q', q'', ...\}$, sowie ε-Transitionen.
- Eine *ε-Transition* ist eine Transition, die der NFA zufällig ausführt (kann, muss aber nicht ausgeführt werden). Jeder NFA kann in einen ε-freien überführt werden.
- Eine *Konfiguration* von $\mathcal{A}$ ist ein Paar (q, w).
- Eine *Einzelschrittralation* $\vdash_{\mathcal{A}}$ ist eine Transition von einer Konfiguration in eine andere, die reflexive, transitive Hülle $\vdash^*_{\mathcal{A}}$ ist die Verkettung aller möglichen Einzelschrittrelationen.
- *Äquivalent* sind zwei Automaten $\mathcal{A}_1, \mathcal{A}_2$ wenn die von ihnen erkannten Sprachen $L(\mathcal{A}_1), L(\mathcal{A}_2)$ gleich sind.

- Als ε-*Hülle* $\hat{\varepsilon}(T)$ einer Zustandsmenge T wird die Menge aller Zustände bezeichnet, die durch ε-Transitionen aus T erreichbar sind.
- Die *Erweiterte Transitionsfunktion* $\hat{\delta}(q, w)$ eines NFA ist Verkettung aller möglichen Transitionen einschließlich der ε-Hülle.
- Die ***Potenzmengenkonstruktion*** ist ein Verfahren einen NFA in einen DFA zu überführen.
  - *Erreichbar* heißt ein Zustand q falls es ein Wort $w \in \Sigma^*$ gibt mit $\overline{\delta}(q_0, w) = q$
  - Für die Menge aller mit einer Eingabe erreichbaren Zustände im NFA wird ein eigener Zustand definiert (z.B. q_{12} falls durch a sowohl q_1 als auch q_2 erreicht werden kann). Ursprung kann dabei ein Zustand oder eine ebensolche Zustandsmenge (= Zustand in der Potenzmengenkonstruktion) sein.
  - Dabei ist ein *Senkenzustand* $\emptyset$ möglich, aus dem keine Transition heraus führt.

4 Reguläre Ausdrücke

- $[[\cdot]]$ ordnet dem Regulären Ausdruck $\cdot$ als Semantik eine Sprache zu.
- Reguläre Ausdrücke werden ähnlich wie Sprachen angegeben: $a, b, ...$ sind einzelne Elemente ("Terminale"), α, β sind längere Ausdrücke.
- Für Reguläre Ausdrücke $\alpha, \beta \in RA(\Sigma)$ gilt:
  - $[[\Lambda]] := \emptyset$ "leerer Ausdruck" (Achtung: $[[\Lambda^*]] = \{\varepsilon\} \neq [[\Lambda]]$!)
  - $[[a]] := \{a\}$ Die Sprache mit dem einzigen Wort a
  - $[[(\alpha + \beta)]] := [[\alpha]] \cup [[\beta]]$ "(nichtexklusives) oder"
  - $[[(\alpha \cdot \beta)]] := [[\alpha]] \cdot [[\beta]]$ "und"
  - $[[(\alpha^*)]] := [[\alpha]]^*$ beliebige Widerholung, auch 0 mal
- Der *Satz von Kleene* liefert ein Konstruktionsverfahren, um aus einem Regulären Ausdruck einen NFA zu machen. Dabei werden + ("oder"), $\cdot$ ("und") und * ("widerholungen") durch entsprechende Automaten ersetzt.
- *Analyse eines DFA (Methode von Kleene)* (Überführen in einen Regulären Ausdruck): Sei $W_{ij}^k := \{w \in \Sigma | w \text{ überführt } q_i \text{ in } q_j \text{ ohne Benutzung von } q_{k+1}, .., q_n\}$. Dann gilt:

  $$L(\mathcal{A}) = \bigcup_{q_j \in F} W_{1j}^n$$

  sowie

  $$W_{ij}^{k+1} = W_{ij}^k \cup (W_{i,k+1}^k (W_{k+1,k+1}^k)^* W_{k+1,j}^k)$$

  Damit kann rekursiv die Sprache des DFA aus einfachen Mengen dargestellt werden. Diese Darstellung läßt sich schließlich in einen regulären Ausdruck überführen.
- *Analyse eines DFA (Methode mit Äquivalenzsystem)*: Zu erstellen ist ein "Gleichungssystem" (für alle Zustände):
  $x_n \sim \ ax_1 + bx_2 + ...$ wobei a bedeutet "So komme ich in einem Schritt von q_n zu

q_1" und b bedeutet "So komme ich in einem Schritt von q_n zu q_2" etc.. Hinter dem Endzustand steht noch "$+\Lambda^*$".
Dieses Gleichungssystem ist dann zu lösen, wobei "+" ein "oder" bedeutet und "$\cdot$" ein "und"! Dabei gilt die *Resolutionsregel*:

$$x \sim \alpha x + \beta \wedge \varepsilon \notin [[\alpha]] \Rightarrow x \sim \alpha^* \beta$$

Die Lösung nach x_1 ist der reguläre Ausdruck, der die Sprache des DFA beschreibt.

5 Eigenschaften regulärer Sprachen

- Die *Nerode-Relation* $\varrho_L \subseteq \Sigma^* \times \Sigma^*$ einer Sprache L ist die Menge aller zweier Worte u, v für die $uw \in L \Leftrightarrow vw \in L$ ($\rightarrow$ Zyklus!)
- Eine *Äquivalenzklasse* $[u]$ eines Wortes u ist die Menge aller Wörter, die mit u in einer Äquivalenzrelation stehen: $[u] := \{v \in \Sigma^* | (u, v) \in \varrho_L\}$
- *Satz von Myhill, Nerode*: Eine Sprache L ist $\in L(\Sigma, DFA)$ wenn sie endliche viele Äquivalenzklassen hat: $L \in \mathcal{L}_3 \Leftrightarrow Index(\varrho_L) < \infty$
  Endliche Sprachen sind damit immer $\in \mathcal{L}_3$.
- Ein *Äquivalenzklassenautomat* $= (Q, \Sigma, \delta, q_0, R)$ ist definiert mit:
  - $Q = \{[x_1], .., [x_k]\}$
  - $q_0 := [\varepsilon]$
  - $\delta([u], a) := [ua]$
  - $F := \{[x_i] | [x_i] \subseteq L\}$
- Der *Minimalautomat* einer Sprache $L \in \mathcal{L}_3$ entspricht bis auf Isomorphie dem Äquivalenzklassenautomaten. Er kann bestimmt werden durch:
  - Elimination aller nicht erreichbaran Zustände
  - Zusammenfassen aller äquivalenten Zustände
- Ein *reduzierter Automat* ist ein Automat, dessen nicht erreichbaren Zustände gestrichen wurden.
- Zwei Zustände heißen *k-äquivalent* ($\overset{k}{\sim}$) wenn $\forall |w| \leq k$ gilt:
  $\bar{\delta}(q_1, w) \in F \Leftrightarrow \bar{\delta}(q_2, w) \in F$
  Also: Nach k Schritten ist man, egal mit welcher Eingabe, immer in einem Endzustand oder immer nicht darin.
- Verfahren mit *Überdeckungsmatrix* (zur Bestimmung eines Minimalautomaten):
  Sei $r^k(q_1, q_2) = 0$ falls $q_1 \overset{k}{\sim} q_2$, sonst 1
  Dann können ab $k = 0$ obere Dreiecksmatrizen $R^k := r^k(q_i, q_j)$ angegeben werden, die ab einem k immer gleich bleiben. Wo sich in der Dreiecksmatrix dann noch 0en befinden können äquivalente Zustände abgelesen werden, die zusammengefasst werden können zu einem Zustand.

- Das ***Pumping-Lemma*** weist nach, dass eine Sprache nicht durch einen endlichen Automaten erkannt werden kann ($L \notin \mathcal{L}_3$)
  Es besagt, dass sich, sofern $L \in \mathcal{L}_3$, für ein existierendes n jedes Wort z mit $|z| \geq n$ in $z = uvw$ zerlegen läßt, so dass
  - $|v| \geq 1$
  - $|uv| \leq n$
  - $uv^i w \in L \forall i \in \mathbb{N}$ (Wort läßt sich auf- und abpumpen)

  denn in einem Endlichen Automaten treten ab einer gewissen Wortlänge Zyklen auf (der Zustand wurde schon mal besucht).
  Ist das Pumping-Lemma erfüllt muss die Sprache aber nicht zwingend Typ-3 sein! (Nur Ausschluss-Kriterium)
- *Leerheitsproblem*: Ist $L = \emptyset$? ($\rightarrow$ Wortproblem bis $|w| \leq n$)
- *Endlichkeitsproblem*: Ist $|L| < \infty$? ($\rightarrow$ Wortproblem $n \leq |w| < 2n$)
- *Äquivalenzproblem*: Ist $L_1 = L_2$?

6 Kontextfreie Sprachen

- *Kontextfrei* sind Sprachen kontextfreier Grammatiken, also der Form $A \rightarrow \alpha$
- Ein *Ableitungs-* oder *Syntaxbaum* ist eine Baumdarstellng der Grammatik mit der Wurzel S. Nonterminale sind dann Knoten, Terminale sind Blätter. Dabei sind unterschiedliche Ableitungen in der Regel möglich.
- *Eindeutig* ist die Grammatik, wenn genau ein Ableitungsbaum existiert.
- *Kettenregeln* sind Produktionen der Form $A \rightarrow B$, wo auf der rechten Seite nur ein Nonterminal steht. Diese können eliminiert werden, indem sie entsprechend mit der Produktion für das rechte Nonterminal (hier B) ersetzt werden (ggf. rekursiv).
- Die ***Chomsky-Normalform*** hat nur Produktionen der Art
  - $A \rightarrow BC$ (zwei Nonterminale)
  - $A \rightarrow a$ (ein Terminal)
  - $S \rightarrow \varepsilon$ und S tritt auf keiner rechten Regelseite auf.

  und stellt damit einen Binärbaum dar. Jede Typ-2-Grammatik hat eine äquivalente Grammatik dieser Form. Diese kann erzeugt werden indem man:
  1. Kettenregeln entfernt
  2. Für jedes rechte Terminalsymbol a (auch Symbole wie z.B. "+"!), das nicht alleine steht, ein Nonterminal $N_a \rightarrow a$ einführt
  3. Für jede rechte Seite, die mehr als zwei Nonterminale hat, müssen (ggf. rekursiv) weitere Regeln C_i eingeführt werden (z.B. erstes stehen lassen, den Rest C_1 nennen usw.).
- Der ***CYK-Algorithmus*** löst das Wortproblem für eine Typ-2-Sprache. Dazu:

1. Grammatik in CNF überführen
2. Terminale horizontal auflisten
3. darunter die entsprechenden Nonterminale
4. Mit dem CYK-Algorithmus immer zwei Elemente ermitteln (hier nicht erklärt, ggf. z.B. bei YouTube nachsehen!) prüfen ob sie so auf einer rechten Regelseite der CNF auftauchen. Dabei ergibt sich schließlich eine umgedrehte Pyramide, "Knoten" sind Mengen erzeugender Produktionen (ggf. leere Menge). Ist die Spitze ein S gilt $w \in L$, ansonsten $w \notin L$.

- Das ***Pumping-Lemma für $\mathcal{L}_2$-Sprachen*** ist eine Erweiterung des "kleinen" Pumping-Lemmas: Für jedes Wort $z \in L \in \mathcal{L}_2 \exists n$ so dass alle z mit der Mindestlänge n sich in $z = uvwxy$ zerlegen lassen mit:
  - $vx \neq \varepsilon$
  - $|vwx| \leq n$
  - $uv^i wx^i y \in L$ (Wort läßt sich auf- und abpumpen)
- *$\mathcal{L}_2$-Eigenschaften*: Abeschlossen unter Vereinigung, Komplexprodukt, Sternoperator, jedoch nicht unter Durchschnitt und Komplement
- ***Kellerautomaten (PDA)*** erkennen Typ-2-Sprachen. Dabei wird zum einen ein Eingabewort zeichenweise gelesen. Zum andern steht dem Automaten ein Stack (= Keller) zur Speicherung von Werten zur Verfügung. Der Kellerautomat ist ein Tupel

  $$\mathcal{A} = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$$

  wobei
  - q, Σ, q_0, F definiert sind wie bei den anderen Automaten
  - Γ das Kelleralphabet (kann auch Σ sein oder enthalten)
  - Z_0 das Kellerstartsymbol
  - $\delta : Q \times (\Sigma \cup \varepsilon) \times \Gamma \to \mathfrak{P}_f(Q \times \Gamma^*)$ die Transitionsfunktion darstellt.
    Beispiel: $\delta : q_0 a Z_0 \to q_1 b Z_0$ beschreibt "Ist-Zustand q_0, a gelesen, Z_0 vom Keller geholt $\to$ in Zustand q_1 gehen und bZ_0 auf Keller schreiben"

  Kellerautomaten können entweder als Liste von δ-Transitionen angegeben werden oder grafisch wie gewohnt dargestellt werden, wobei die Transition im obigen Beispiel mit $a, Z_0 | bZ_0$ beschriftet würde.
- Die *Konfigurationsfolge* eine PDA wird angegeben durch "(Zustand, Eingaberest, Keller)".
- *Endkonfigurationen von Kellerautomaten* Es gibt drei mögliche Endkonfigurationen, die alle $\mathcal{L}_2$ erkennen können:
  - $L(\mathcal{A}, F)$: (q, ε, γ) mit $q \in F$ (Endzustand)
  - $L(\mathcal{A}, \varepsilon)$: $(q, \varepsilon, \varepsilon$ mit $q \in Q$ (leerer Keller)
  - $L(\mathcal{A}, \varepsilon, F)$: $(q, \varepsilon, \varepsilon$ mit $q \in F$ (leerer Keller und Endzustand)

- Sprachen von *deterministischen Kellerautomaten (DPDA)* sind eine echte Teilmenge von $L(PDA)$ (Im Gegensatz zum DFA!). Sie terminieren per Endzustand ($q \in F$). Für sie gilt

$$|\delta(q, a, Z)| + |\delta(q, \varepsilon, Z)| \leq 1$$

  Transitionen sind also eindeutig.
  Es werden von ihnen die *deterministischen kontextfreien Sprachen* erkannt.

7 Turingmaschinen

- Eine ***Turingmaschine*** ist ein 7-Tupel

$$\mathcal{A} = (Q, \Sigma, \Gamma, \delta, q_0, \bar{b}, F)$$

  wobei das Arbeitsalphabet $\Gamma \supset \Sigma$ eine Übermenge des Eingabealphabetes ist und $\bar{b}$ das Leerzeichen/Blank darstellt.
  Die Turingmaschine arbeitet auf einem unendlichen Lese- und Schreibband, welches zugleich der Eingabe dient.

- Eine *Einzelschrittrelation* ist dann: $\vdash_{\mathcal{A}} \subseteq \Gamma^* Q \Gamma \Gamma^* \times \Gamma^* Q \Gamma \Gamma^*$

- Eine Transition $q_0 a A R q_1$ bedeutet: Anfangszustand q_0, gelesenes Zeichen a, geschriebenes Zeichen A, Positionsänderung des Lese/Schreibkopfes nach R (R: rechts, L: links, N: keine Bewegung), Nachfolgezustand q_1. Eine Auflistung der Transitionen heißt *Turingtafel*, die auch Kommentare enthalten kann.

- Eine *Konfigurationsfolge* zeichnet den Zustandsverlauf der Turingmaschine auf. Es wird dabei der Inhalt des Lese/Schreibbandes angegeben, der Zustand steht links neben der aktuellen Potition des Lese-/Schreibkopfes. Beispiel (vgl. oben): $\alpha q_0 a \beta$.

- $L(\Sigma, TM) = L(\Sigma, DTM) = \mathcal{L}_0$

- *Linear beschränkte Automaten (LBA)* sind Turingmaschinen mit einem endlichen Lese-/Schribband, das durch ¢und \$ beschränkt wird. Es gilt $L(LBA) = \mathcal{L}_1$

Teil II
Berechenbarkeitstheorie und Komplexitätstheorie

8 Formalisierung des Berechenbarkeitsbegriffes

- Eine Funktion $f = f_{\mathcal{A}}$ heißt *Turing-Berechenbar* wenn eine Turingmaschine existiert die das Ergebnis von f berechnet und auf das Band schreibt. Dabei muss gelten:

$$f_{\mathcal{A}}(w) = v \Leftrightarrow q_0 w \bar{b} \vdash_{\mathcal{A}}^* \alpha q v \bar{b} \beta$$

  Das Ergebnis ist also von der Position des Lese/Schribkopfes bis zu einem $\bar{b}$ auf dem Band. Die DTM muss hier anhalten.

- Die *semi-charakteristische Funktion* einer Sprache L ist definiert als
  $\chi'_L(w) = \varepsilon$ falls $w \in L$, *n.d.* sonst.
  $L \in \mathcal{L}_0 \Leftrightarrow \chi'_L$ ist Turing-berechenbar $\Leftrightarrow L$ ist semi-entscheidbar (Denn wenn $w \notin L$ terminiert $\mathcal{A}$ nicht)

- Die *Church'sche These* nimmt an, dass eine Funktion genau dann berechenbar ist, wenn sie Turing-Berechenbar ist. Bewiesen sind bislang die Äquivalenzen von Turing-, While-, Goto-Berechenbarkeit, μ-Rekursivität und λ-Definierbarkeit (und weitere).

- ***LOOP-Berechenbare Funktionen*** sind eine echte Teilmenge der Turing-berechenbaren Funktionen.

- Ein *LOOP-Programm* ist definiert durch

$$\mathcal{P}_{LOOP} = \{\texttt{in}(X_1, ..X_n); \texttt{istvar}(X_n + 1, .., X_m); \alpha, \texttt{out}X_1 | m \geq 1\}$$

  wobei X_i Variablen (Zahlen) sind und $\alpha \in \mathcal{A}$ eine oder mehrere (auch verschachtelte) der folgenden Anweisungen:

  - Wertzuweisung $\mathcal{W} : X_i := X_j + 1$ oder $X_i := 0$
  - "Verkettung" $\alpha; \beta$
  - $\texttt{loop}V(\alpha)$; eine V-malige widerholung der Anweisung(en) α

- Die *Semantik von LOOP* besteht aus

  - der *Eingabeabbildung* $\texttt{in}_m^{(n)} : (a_1, .., a_n) \to (a_i, .., a_n, 0, .., 0)$
  - der *Ausgabeabbildung* $\texttt{out}_1^{(m)} : (a_1, .., a_m) \to a_1$
  - sowie der *Zustandstransformation* $[[\alpha]]^{(m)} : \mathbb{N}^m \to \mathbb{N}^m$, induktiv definiert durch
    * $[[X_i := X_j + 1]]^{(m)}(a_1, .., a_m) := (a_1, .., a_{i-1}, a_j + 1, a_{i+1}, .., a_m)$
      $[[X_i := 0]]^{(m)}(a_1, .., a_m) := (a_1, .., a_{i-1}, 0, a_{i+1}, .., a_m)$
    * $[[\alpha; \beta]]^{(m)} := [[\beta]]^{(m)} \circ [[\alpha]]^{(m)}$
    * $[[\texttt{loop}X_i(\alpha)]]^{(m)}(a_1, .., a_m) := ([[\alpha]]^{(m)})^{a_i}(a_1, .., a_m)$

  Damit kann die Semantik (Funktion) des LOOP-Programmes bestimmt werden.

- *primitiv-Rekursive Grundfunktionen*:

  - *Nachfolgerfunktion* $S : a \to a + 1$
  - *Nullkonstante* $0 : () \to 0$
  - *Projektion* $pr_i^{(n)} : (a_1, .., a_n) \to a_i$

- *Primitiv rekursive Funktionen* werden durch

  - *Komposition* $(f \circ f' = f(f'))$
    und
  - *primitiver Rekursion*:
    $h(a_1, .., a_n, 0) := f(a_1, .., a_n)$
    $h(a_1, .., a_n, a + 1) := g(a_1, .., a_n, a, h(a_a, .., a_n, a))$

erzeugt. f ist genau dann primitiv rekursiv, wenn f LOOP-berechenbar ist.

- ***WHILE-Programme*** unterscheiden sich von LOOP-Programmen durch
  - zwei weitere *Wertzuweisungen*:
    $\mathcal{W} := \{X_i := X_j + 1, X_i := X_j - 1, X_i := X_j, X_i := 0\}$ sowie
  - die Schleife $\texttt{while}X_i \neq 0\texttt{do}\alpha\texttt{od} \in \mathcal{A}$

  für die Semantik gilt gleiches wie für das LOOP-Programm sowie
  $[[\texttt{while}X_i \neq 0\texttt{do}\alpha\texttt{od}]]^{(m)}(a_1, .., a_m) := ([[\alpha]]^{(m)})^k(a_1, .., a_m)$
  für k minimal, so dass der Term $= 0$ ist (ggf. nicht definiert)

- Eine ***Minimalisierung*** einer Funktion $f : \mathbb{N}^{n+1} \to \mathbb{N}$ ist definiert als
  $$\mu(f)(a_1, .., a_n) := \begin{cases} b & \text{falls } f(a_1, .., a_n, b) = 0 \text{ mit } b \text{ minimal} \\ n.d. & \text{sonst} \end{cases}$$

- *μ-rekursive Funktionen* sind durch primitive Grundfunktionen in endlichen Schritten durch Komposition, primitive Rekursion oder Minimalisierung erzeugbar. Sie sind mit höchstens einer While-Schleife berechenbar.

9 Entscheidbarkeit, Aufzählbarkeit, Berechenbarkeit

- Die *charakteristische Funktion* ist gegeben durch
  $$\chi_L(w) = \begin{cases} \varepsilon & \text{falls } w \in L \\ a_1 & \text{sonst} \end{cases}$$
  Wenn diese Turing-Berechenbar ist heißt L *entscheidbar*
  $\Leftrightarrow L$ und $\Sigma^* \setminus L$ rek. aufzählbar $\Leftrightarrow \Sigma^* \setminus L$ entscheidbar

- L *rekursiv aufzählbar* $\Leftrightarrow L = \emptyset \vee \exists f : Bild(f) = \{f(w) | w \in \Sigma^*\} = L$ wobei f total ("beidseitig eindeutig") und berechenbar sein muss
  $\Leftrightarrow$ *Semi-entscheidbar* $\Leftrightarrow \chi'_L$ berechenbar $\Leftrightarrow L \in \mathcal{L}_0$

- $\mathcal{L}_1(\Sigma) \subsetneqq \{L \subseteq \Sigma^* | L \text{ entscheidbar}\} \subsetneqq \mathcal{L}_0(\Sigma)$

- L_1 ist auf L_2 *reduzierbar* $\Leftrightarrow L_1 \leq L_2$ genau dann, wenn
  eine totale und berechenbare Funktion f existiert mit $w \in L_1 \Leftrightarrow f(w) \in L_2 \forall w \in \Sigma^*$
  Ist dann L_2 (semi-)entscheidbar, so ist auch L_1 (semi-)entscheidbar.

- Der *Satz von Rice* besagt, dass eine Eigenschaft entweder für alle rekursiv aufzählbaren Sprachen entscheidbar, oder für alle nicht-entscheidbar ist (= nichttriviale Eigenschaft P).

- Die *Universelle Sprache*, das *Halteproblem* und das *Postsche Korrespondenzproblem* sind nicht entscheidbare Probleme.