Syntaktische Kongruenz

Sei: M Monoid, $L \subset M$.

Definition: Für $a, b \in M$ gilt

$$a \equiv b \ (L) :\Longleftrightarrow \forall\, u, v, \in M: \ (uav \in L \Leftrightarrow ubv \in L).$$

Sprechweise: a syntaktisch kongruent b modulo L.

Es gilt für $a, b, c \in M$

$$\text{Äquivalenz} \begin{cases} a \equiv a \ (L), \\ a \equiv b \ (L) \Leftrightarrow b \equiv a \ (L), \\ a \equiv b \ (L), b \equiv c \ (L) \Rightarrow a \equiv c \ (L) \end{cases}$$

Kongruenz:

$$a \equiv b \ (L) \Leftrightarrow \forall\, u, v \in M: \ uav \equiv ubv \ (L)$$

Syntaktisches Monoid (1)

Definition:

$$[a]_L := \{b \in M \mid a \equiv b(L)\}.$$
$$synt_M(L) := \{[a]_L \mid a \in M\}.$$
$$[a]_L \cdot [b]_L := [a \cdot b]_L.$$

Beweis der Wohldefiniertheit

$$a' \equiv a(L) \longrightarrow [a \cdot b]_L \equiv [a' \cdot b]_L.$$

Beweis: Für alle $u, v \in M$ gilt

$$ua'v \in L \Longleftrightarrow uav \in L.$$

Wähle v speziell: $v = b \cdot v'$.

Es folgt $uabv' \in L \Longleftrightarrow ua'bv' \in L$ für alle $u, v' \in M$, d.h. $[ab]_L = [a'b]_L$.

Analog für $b \equiv b'(L)$: $[ab]_L = [ab']_L$.

Es folgt $[ab]_L = [a'b]_L = [a'b']_L$

Syntaktisches Monoid (2)

Es gilt

$$\begin{array}{rcccl} ([a]\cdot[b])\cdot[c] & = & [ab]\cdot[c] & = & [abc] \\ [a]\cdot([b]\cdot[c]) & = & [a]\cdot[bc] & = & [abc]. \end{array}$$

Also: Das Produkt ist assoziativ.

Es gilt

$$[1]\cdot[a] = [a] = [a]\cdot[1].$$

Also: [1] ist Einheit in $synt_M(L)$.

Also: $synt_M(L)$ ist ein Monoid.

Es heißt das syntaktische Monoid von L bezüglich M.

Ergänzung zu Syntaktisches Monoid

Lemma: $L = \bigcup_{u\in L}[u]_L$

Beweis: $[u] \subset L$ oder $[u] \cap L = \emptyset$

Syntaktisches Monoid (3)

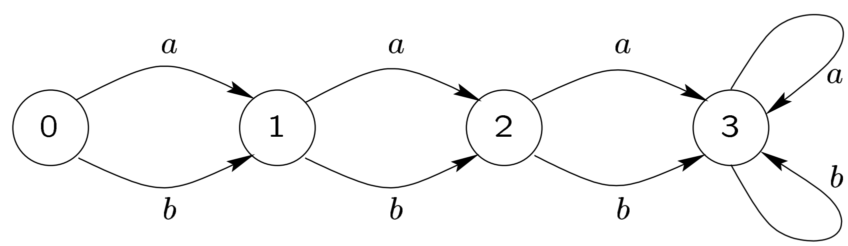
Lemma: L endlich $\Longrightarrow$ $synt(L)$ endlich.

Beweis:

$$m = max\{|w| \mid w \in L\} \Longrightarrow \{w \mid |w| > m\} \subset 0.$$

Beispiel:

$$\begin{aligned} L &= \{aa, ab, ba, bb\} \\ synt(L) &= \{[1], [a], [aa], [aaa] = 0\}. \end{aligned}$$



Ein Beispiel: Das syntaktische Monoid der Dycksprache.

Sei $\Sigma := \Sigma^{+} \cup \Sigma^{-}$

$$\Sigma^{+} := \{\sigma_1, \ldots, \sigma_n\}, \Sigma^{-} := \{\sigma_1^{-}, \ldots, \sigma_n^{-}\}.$$

Syntaktisches Monoid (4)

Definition: $D(\Sigma)$ ist der Durchschnitt aller Teilmengen L von Σ^*, die $(1), (2)$ und (3) erfüllen. $D(\Sigma)$ heißt Dycksprache zu Σ.

$$\begin{array}{lll} (1) & 1 \in D(\Sigma) & \\ (2) & u, v \in D(\Sigma) & \Rightarrow \;\; u \cdot v \in D(\Sigma) \\ (3) & u \in D(\Sigma) & \Rightarrow \;\; \forall_i \sigma_i \cdot u \cdot \sigma_i^- \in D(\Sigma) \end{array}$$

Wegen (1) und (2) gilt: $D(\Sigma)$ ist Monoid.

Man kann zeigen (Übung): (Wir lassen den Index L in $[]_L$, $L = D(Z)$ weg.)

$$[1] = D(\Sigma).$$

$$0 := [\sigma_i] \cdot [\sigma_j^-] \quad \text{für } i \neq j.$$

ist unabhängig von i, j und es gilt

$$\forall_{a \in M} 0 \cdot [a] = [a] \cdot 0 = 0.$$

Zu jedem $u \in (\Sigma^+)^*$ gibt es genau ein $v \in (\Sigma^-)^*$ mit

$$[u] \cdot [v] = [1].$$

Syntaktisches Monoid (5)

Approximation von $L \subset M$

Aufgabe: Sei $L \subset M$ gegeben. Gesucht werden

- ein Monoid S,
- ein Homomorphismus $h : M \to S$,
- ein $T \subset S$

so, daß gilt:

(1) $h^{-1}(T) \supset L$

(2) Es gibt keine Wahl S', T', h', so daß

$$h^{-1}(T) \supsetneq {h'}^{-1}(T') \supset L$$

gilt.

Satz: Wählt man $S := synt_M(L)$ und $T := \{[a] \mid a \in L\}$, dann erhält man eine Lösung der Aufgabe: $L = h^{-1}(T)$ mit $h(a) := [a.]$

Graphen, Kategorien, Funktoren (1)

Definition: $G = (E, K, \varrho)$, $K \neq \emptyset$ heißt **Graph** $\Longleftrightarrow$: $\varrho : E \to 2^k$, $1 \leq card\ \varrho(s) \leq 2$ für $s \in E$. K heißt Knotenmenge, E Kantenmenge des Graphen. Ist $card\ \varrho(s) = 1$, dann heißt s Schleife.

G heißt **orientierter** Graph:

$$\varrho : E \longrightarrow K \times K.$$

Wir definieren zwei Abbildungen $Q : E \to K$, $Z : E \to K$ mit $\varrho(s) = (Q(s), Z(s))$ für $s \in E$. $Q(s)$ heißt Anfangspunkt, $Z(s)$ Endpunkt von s.

Man kann jedem Graphen G einen orientierten Graphen $\tilde{G}$ zuordnen, indem man jeder Kante s zwei Kanten $(P_1 s P_2)$ und $(P_2 s P_1)$ zuordnet, wenn $\varrho(s) = \{P_1, P_2\}$ ist. Wir setzen

$$Q(P_1 s P_2) = P_1, \quad Z(P_1 s P_2) = P_2 \quad \text{und}$$
$$Q(P_2 s P_1) = P_2, \quad Z(P_2 s P_1) = P_1.$$

$\tilde{G}$ heißt der zu G gehörige orientierte Graph.

Graphen, Kategorien, Funktoren (2)

Definition: Ist $G = (E, K, \varrho)$ ein orientierter Graph, dann heißt $w \in E^* - \{1\}$ Weg in G, wenn für $w = (s_1, \ldots, s_n)$

$$Q(s_{i+1}) = Z(s_i) \quad \text{für } i = 1, \ldots, n-1$$

gilt. Wir setzen $Q(w) = Q(s_1)$, $Z(w) = Z(s_n)$. $Q(w)$ heißt der Anfangspunkt, $Z(w)$ der Endpunkt von w.

Ist G ein nicht orientierter Graph, dann heißt jeder Weg in dem zugehörigen orientierten Graphen $\tilde{G}$ **Weg** in G.

Wir ergänzen die Menge der Wege durch die Menge der „leeren Wege"

$$\{(P, P) \mid P \in K\}.$$

Bezeichnung: $1P := (P, P)$,

$\mathcal{W}(G) = \{w \in E^* - \{1\} \mid w \text{ Weg in } G\} \cup \{1_P \mid p \in K\}$.

Graphen, Kategorien, Funktoren(3)

Der Graph G heißt **zusammenhängend** $:\Longleftrightarrow$ Zu jedem $(P_1, P_2) \in K^2$ gibt es einen Weg in G mit $Q(w) = P_1, Z(w) = P_2$.

Ist G ein orientierter Graph, dann heißt er **schwach** zusammenhängend, wenn es zu $P_1, P_2 \in K$ einen Weg gibt mit

$$P_1 \xrightarrow{w} P_2 \quad \text{oder} \quad P_2 \xrightarrow{w} P_1.$$

Der orientierte Graph heißt **stark** zusammenhängend, falls es zu jedem **Paar** $(P_1, P_2) \in K \times K$ ein $w \in \mathcal{W}(G)$ gibt mit $Q(w) = P_1,)$ $Z(w) = P_2$.

Ein Knoten S von G heißt ein **Stern** von G $:\Longleftrightarrow$ Ist $P \in K$, dann gibt es einen Weg

$$S \xrightarrow{w} P.$$

in G.

Graphen, Kategorien, Funktoren (4)

Lemma: Ein zusammenhängender nicht orientierter Graph hat stets einen Stern; jeder Knoten des Graphen ist auch Stern von G.

Beweis:

Definition: Ein geschlossener, doppelpunktfreier Weg von G heißt **Kreis** von G.

P Doppelpunkt von w: liegt zweimal im Inneren von w.

Graphen, Kategorien, Funktoren (5)

Definition: Ein kreisfreier (orientierter) Graph mit Stern heißt Baum. Sterne eines Baumes heißen Wurzeln des Baumes.

Lemma: Zu jedem Graphen $G = (E, K)$ mit Stern S gibt es einen Baum mit K als Knotenmenge, der Untergraph von G ist und S als Wurzel hat.

Definition: Ein Baum, wie er in dem vorhergehenden Lemma beschrieben wird, heißt **aufspannender** Baum von G.

Lemma: Die aufspannenden Bäume nicht orientierter Graphen lassen sich durch eine Folge von Kantenaustauschschritten ineinander transformieren.

Graphen, Kategorien, Funktoren (6)

Beweis: Übung.

Hinweise:

(1) Definiere Distanz von aufspannenden Bäumen

(2) Zeige, daß sich die Distanz verringern läßt.

Beobachtung: Die Wurzel und die fehlenden Kanten (von G) bestimmen den jeweiligen Baum eindeutig.

Wegekategorie (1)

Die Kategorie der Wege von G

Sind $w_1, w_2 \in \mathcal{W}(G)$ und ist $Q(w_2) = Z(w_1)$

$w_1 = (Q_1, s_1, \ldots, s_n, Z_1),\; w_2 = (Q_2, t_1, \ldots, t_m, Z_2),$

dann ist wegen $Z_1 = Z(w_1) = Q(w_2) = Q_2$ auch

$$w := (Q_1, s_1, \ldots, s_n, t_1, \ldots, t_m, Z_2)$$

ein Weg aus $\mathcal{W}(G)$.

Definition: Das Produkt $w_1 \cdot w_2$ zweier Wege w_1, w_2 ist genau dann definiert, wenn $Z(w_1) = Q(w_2)$ ist und es gilt $w_1 \cdot w_2 := w$.

Lemma: Das Produkt von Wegen ist, soweit definiert, assoziativ. Für $1_{Q(w)}$ und $1_{Z(w)}$ gilt

$$1_{Q(w)} \cdot w = w = w \cdot 1_{Z(w)}.$$

Beweis: Klar.

Wegekategorie (2)

$(\mathcal{W}(G), \cdot, Q, Z)$ ist eine **Kategorie**.

Definition:

(1) $w \cdot v$, $w \cdot u$ und $r \cdot v$ definiert, dann auch $r \cdot u$ definiert.

(2) Zu jedem $w \in \mathcal{W}$ gibt es eine Linkseins und eine Rechtseins.

Wegekategorie (3)

Definition: Eine Menge $\mathcal{K}$ mit einer partiellen Verknüpfung $\cdot$ heißt Kategorie $:\Longleftrightarrow$ Es gelten

(1) Sind die Produkte $(w_1 \cdot w_2) \cdot w_3$ und $w_1 \cdot (w_2 \cdot w_3)$definiert, dann gilt

$$(w_1 \cdot w_2) \cdot w_3 = w_1 \cdot (w_2 \cdot w_3).$$

(2) Zu jedem w gibt es eine Linkseins $1_{l(w)}$ und $1_{r(w)}$ mit

$$1_{l(w)} \cdot w = w = w \cdot 1_{r(w)}.$$

(3) Ist $w \cdot v$ definiert, dann ist $1_{r(w)} = 1_{l(v)}$.

Bemerkung: Anstelle von $l(w)$ schreibt man üblicherweise $Q(w)$ und anstelle von $r(w)$ $Z(w)$.

Wegekategorie (4)

Der Begriff des Homomorphismus überträgt sich in natürlicher Weise.

(1) $\varphi(w \cdot v) = \varphi(w) \cdot \varphi(w)$

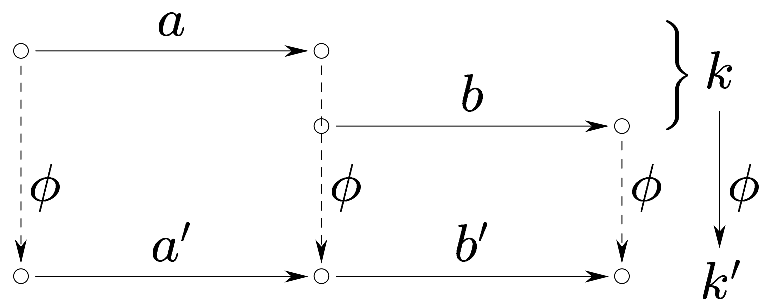
(2) $\varphi(1_{l(w)}) = 1_{l(\varphi(w))}$, $\varphi(1_{r(w)}) = 1_{r\varphi(w)}$.

Im Zusammenhang mit Kategorie spricht man aber nicht von Homomorphismen, sondern von Funktoren.

Grund: Ein für Homomorphismen fundamentaler Satz gilt nicht. $\varphi(\mathcal{K})$ ist nicht notwendig eine Kategorie.

Wegekategorie (5)

Beispiel:



$a \cdot b$ ist **nicht** definiert in $\mathcal{K}$.

Für $a' = \varphi(a)$ und $b' = \varphi(b)$ ist $a' \cdot b'$ definiert.

$\varphi(\mathcal{K})$ enthält $a' \cdot b'$ nicht.

Wegekategorie (6)

Die Definition der „Freiheit“ von Monoiden überträgt sich auf Kategorien. Dem Wortmonoid entspricht (als Verallgemeinerung) die Wegekategorie. Es gilt für jede Kategorie $\mathcal{K}$:

$$\varphi' : E \cup \{1_P \mid P \in K\} \to \mathcal{K}$$

Jede Abbildung φ', die $\varphi'(1_{l(w)}) = 1_{l\varphi'(w)}$ und $\varphi'(1_{r(w)}) = 1_{r(\varphi'(w))}$ erfüllt, läßt sich auf genau eine Weise zu einem Funktor $\varphi : \mathcal{W}(G) \to \mathcal{K}$ fortsetzen.

Die Voraussetzung ist immer erfüllt, wenn $\mathcal{K}$ ein Monoid M ist.

Bezeichnung: $\varphi : \mathcal{W}(G) \to M$ heißt Beschriftung von $\mathcal{W}(G)$ mit Elementen aus M.

Endliche Akzeptoren (1)

Definition: $a = (\mathcal{W}(G), S, T, \mu, X)$ heißt (nicht deterministischer) Akzeptor, falls $S, T \subset K(G)$ und $\mu : \mathcal{W}(G) \to X^*$ ein Funktor in das freie Monoid X^* ist. S heißt die Menge der Startknoten und T die Menge der Terminalknoten. $\mathcal{W}(G)(S,T) = \{w \in \mathcal{W}(G) \mid Q(w) \in S$ und $Z(w) \in T\}$, ist die Menge der Wege von S nach T in G.

Die Menge

$$L_a = \{\mu(w) \mid w \in \mathcal{W}(G)(S,T)\}.$$

heißt die durch a akzeptierte Sprache.

Beispiel: Akzeptor für die Dycksprache

$K := synt(L)$,

$E := \{([u], \sigma_i, [u\sigma_i]) \mid u \in \Sigma^*, i = 1, \ldots, n\}_n \cup \{([u], \sigma_i^-, [u\sigma_i^-]) \mid u \in \Sigma^*, i = 1, \ldots, u\}$.

Endliche Akzeptoren (2)

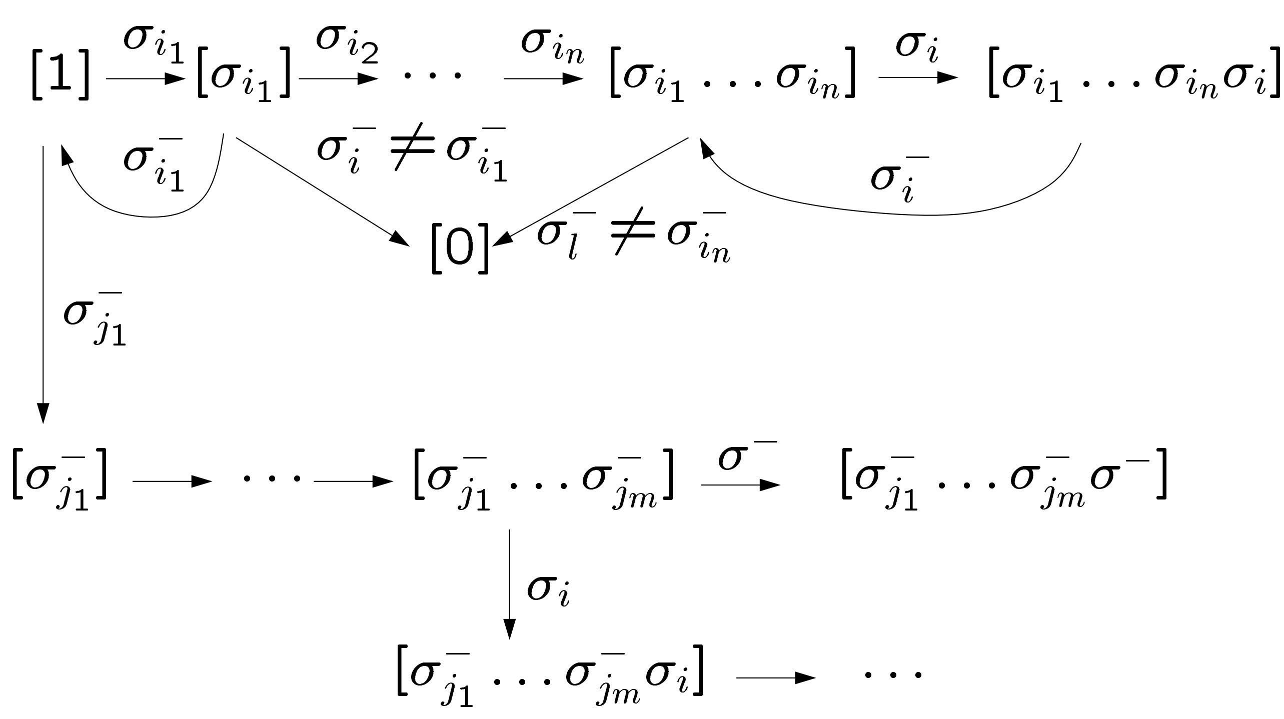
Die Menge der Kanten ist also repräsentiert durch

$$[u] \xrightarrow{\sigma_i} [u\sigma_i],\ \ [u] \xrightarrow{\sigma_i^-} [u\sigma_i^-].$$

$G = (E, K, \varrho)$;

$$\varrho([u], \sigma_i, [u\sigma_i]) = ([u], [u\sigma_i]).$$

Der Graph hat eine sehr einfache Struktur:



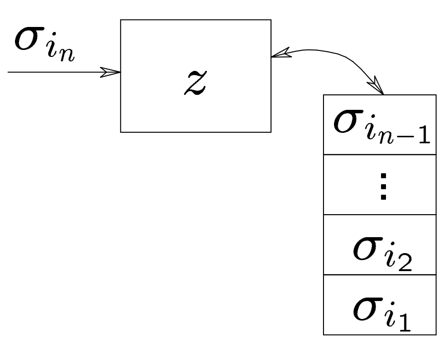
Endliche Akzeptoren (3)

Man sieht, daß der Zweig des Graphen, der mit einem σ_i^- beginnt, keinen (orientierten) Weg zu $T = [1]$ enthält. Damit ist dieser Zweig für den Akzeptor überflüssig.

Erläuterung durch Klammern

$$\begin{tikzcd} {} \arrow[r] & (\,[\,(\,(\,[\arrow[r, "{]}"] \arrow[d, "{)}"] & (\,[\,(\,(\arrow[r] \arrow[d, "{]}"] & {} \\ & 0 & 0 \end{tikzcd}$$

$synt(D(\Sigma))$ repräsentiert die einfachste Form eines Kellerautomaten.



z Zustand 1 weiter offen.

$z = 0$ Ablehnung.

Endliche Akzeptoren (4)

Endlicher Automat

$a = (X, Z, Y, \delta, \lambda, z_0)$, $z_0 \in Z$ (Anfangszustand)

$$\delta \subset X \times Z \times Z, \ \lambda \subset X \times Z \times Y$$

a ist deterministisch, wenn $\delta : X \times Z \to Z$ und $\lambda : X \times Z \to Y$ Abbildungen sind.

Repräsentationen von a durch einen Graphen Knotenmenge Z. Kantenmenge $(x, z_1, z_2) \in \delta$

Graphik s. Tafel!

y bezeichnet die Ausgabe:

$$(x, z, y) \in \lambda.$$

Endliche Akzeptoren (5)

Indem man $Z' = Z \times Y$ setzt und

$$\begin{aligned} \tilde{\delta} = \ & \{((x, y_1), (z_2, y_2)) \mid (x, z_1, z_2) \in \delta \\ & \text{und } (x, z_1, y_2) \in \lambda\} \end{aligned}$$

$$\tilde{\lambda} \subset Z'$$

kann man formal die Ausgabe nur vom Zustand abhängig machen.

Nun definiert man

$$T_y = \{z \mid (z, y) \in \tilde{\lambda}\}.$$

Für jede Ausgabe wählt man eine Menge von Terminalzuständen. Das ist äquivalent zur Originaldefinition.

Somit kann man sich in der Theorie der endlichen Automaten weitgehend auf die Theorie der endlichen Akzeptoren zurückziehen. Das gilt zumindest soweit, als man sich nur für die grundsätzliche Leistungsfähigkeit der Automaten interessiert.

Endliche Akzeptoren (6)

Definition: $REG(X^*) = \{L \subset X^* \mid$ es gibt ein $a = (G, S, T, X, \mu)\}$, so daß

$$L = \mu(\mathcal{W}(G)(S, T)).$$

$REG(X^*)$ heißt die Menge der **regulären Mengen**.

Wir haben gesehen, daß

$$L = h^{-1}(T), \quad \begin{array}{l} h : X^* \to synt(L) \\ T = \{[a]_L \mid a \in L\}. \end{array}$$

Ist L endlich, so ist auch $synt(L)$ endlich.

Frage: Was kann man über L aussagen, wenn $synt(L)$ endlich ist?

Frage: Kann man die endlichen Akzeptoren aus einfacheren Akzeptoren aufbauen?

Wie sehen die regulären Sprachen aus? Kann man sie aus wenigen einfachen reg. Sprachen konstruieren?

Endliche Akzeptoren (7)

Beobachtung: $\{x\} \in REG(X^*)$ für $x \in X$, $\emptyset \in REG(X^*)$.

Satz:

$$L_1, L_2 \in REG(X^*) \Rightarrow \begin{array}{ll} (1) & L_1 \cup L_2 \in REG(X^*) \\ (2) & L_1 \cdot L_2 \in REG(X^*) \\ (3) & L_1^* \in REG(X^*) \end{array}$$

Beweis: Tafel

Satz: Ist $h : X^* \longrightarrow \tilde{X}^*$ ein Homomorphismus, dann gilt

$$L \in REG(X^*) \Rightarrow h(L) \in REG(\tilde{X}^*).$$

Endliche Akzeptoren (8)

$sp : X^* \longrightarrow X^*$ definiert durch

$$\begin{array}{lcl} sp(1) & = & 1 \\ sp(x) & = & x \quad \text{für } x \in X \\ sp(u \cdot v) & = & sp(v) \cdot sp(u). \end{array}$$

Sonderfall eines Antihomomorphismus.

Satz: $L \in REG(X^*) \Rightarrow sp(L) \in REG(X^*)$.

Satz: $L \in REG(X^*) \Rightarrow \overline{L} := X^* - L \in REG(X^*)$.

Der Beweis ergibt sich aus den folgenden Lemmata.

Endliche Akzeptoren (9)

Lemma: Zu jedem Akzeptor a gibt es einen Akzeptor a', der (1), (2) und (3) erfüllt.

(1) $L_a = L_{a'}$ und $\sharp S' = \sharp T' = 1$.

(2) $\mu'(s) \in X \cup \{1\}$ für jede Kante s von a'.

(3) Ist $\mu'(s_1) = \mu'(s_2) = 1$ für zwei Kanten s_1, s_2 von a', dann gilt $s_1 = s_2$ und $Q(s_1) \in S', Z(s_1) \in T'$.

Beweis: Der Beweis erfolgt in drei Schritten.

Schritt 1: Ist s eine Kante von a mit $m := \mu(s) > 1$, dann ersetze s durch einen Weg

$$(Q(s), s_1, s_2, \ldots, s_m, Z(s))$$

mit neuen Kanten $s_1, \ldots, s_m$ und definiere

$$\mu(s_i) = x_i \quad \text{für} \quad \mu(s) = x_1, \ldots, x_m,\ x_i \in X.$$

Endliche Akzeptoren (10)

Wiederhole diesen Schritt so lange, bis alle solchen Kanten eliminiert sind.

Resultat: Akzeptor a_1 mit $L_{a_1} = L_a$ und $\mu_1(s) \in X \cup \{1\}$.

Schritt 2: 1–elementige Start- und Terminalmengen.

(a) Falls S_1 mehrere Startknoten enthält, ($\sharp S_1 > 1,$) dann füge einen neuen Knoten P_0 ein und zu jedem $P \in S_1$ eine Kante s_P mit $Q(s_P) = P_0$ und $Z(s_P) = P$. Setze $\mu(s_P) = 1$.

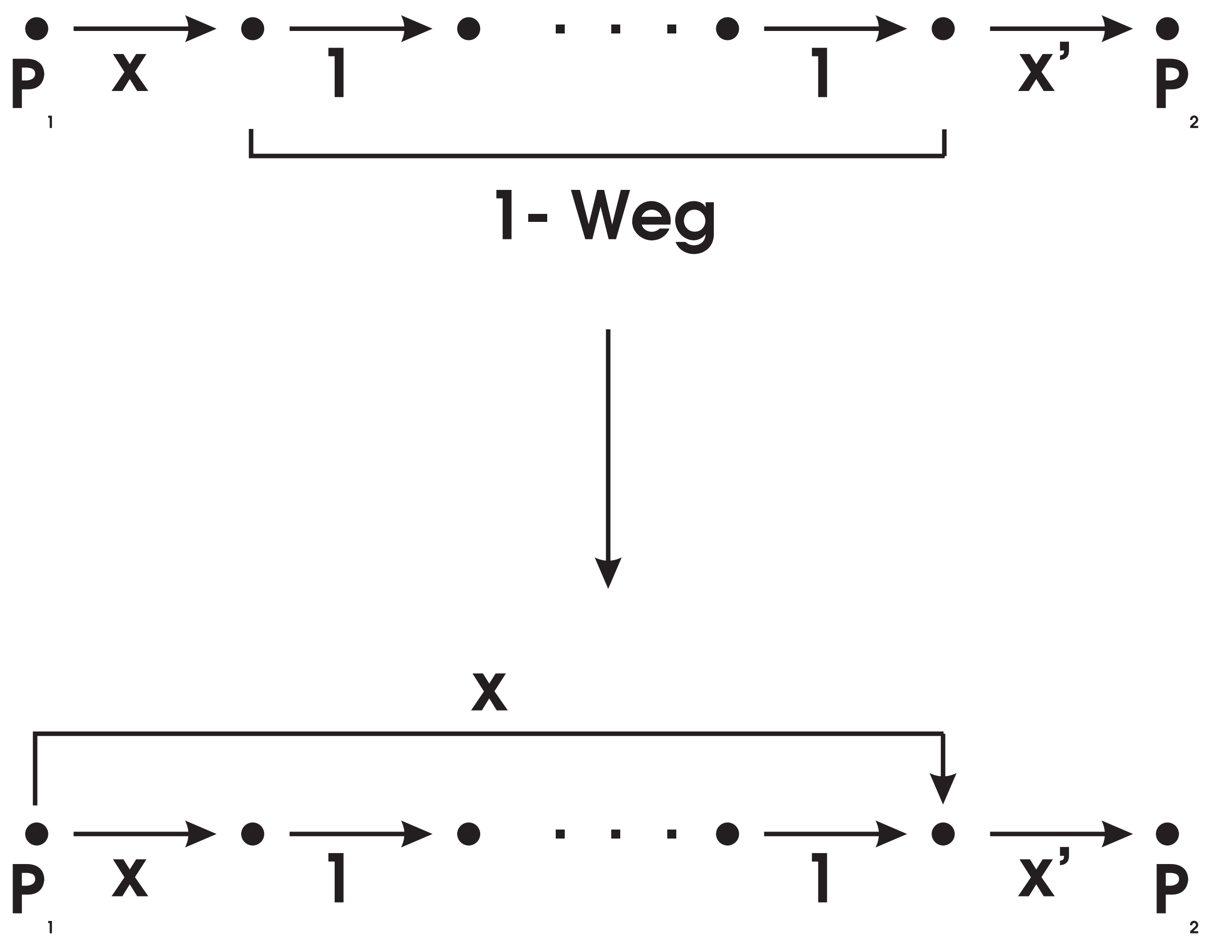
(b) Verfahre analog mit T_1.

Resultat: Akzeptor a_2 mit $L_{a_2} = L_{a_1}, \sharp S_1 = \sharp T_1 = 1$.

Endliche Akzeptoren (11)

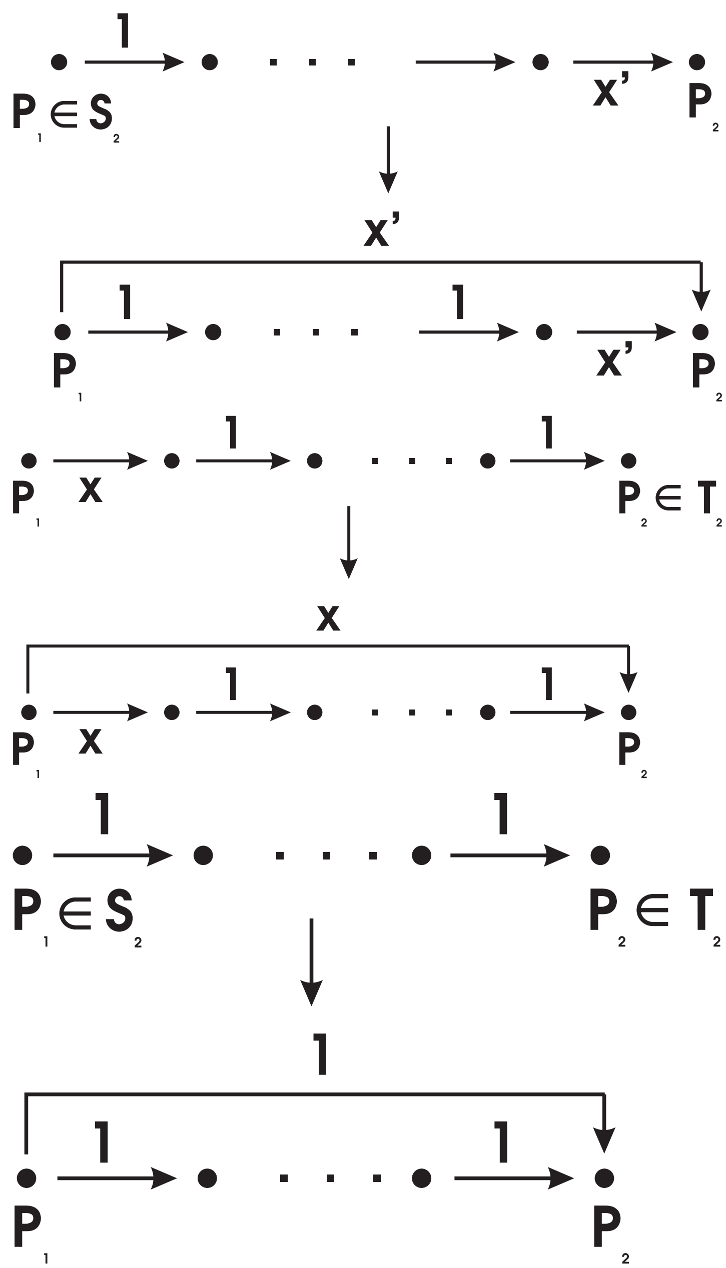
Schritt 3: Überbrücke alle „maximalen“ $1-$Wege w von a_2 durch Kanten.

1. Allgemeiner Fall $x, x' \in X$.



Endliche Akzeptoren (12)

2. Sonderfälle



(a)

(b)

(c)

Endliche Akzeptoren (13)

Iteriere diese Schritte so lange wie möglich, ohne Parallelkanten mit gleicher Markierung zu erzeugen.

Resultat: $a_3, L_{a_3} = L_{a_2}$; gibt es im Akzeptor a_3 einen Weg w, $|w| \geq 1$ mit $\mu_3(w) \in X$, dann gibt es auch eine Kante s in a_3 mit $Q(w) = Q(s)$, $Z(w) = Z(s)$ und $\mu(w) = \mu(s)$.

Schritt 4: Eliminiere alle Kanten s von a_3 mit $\mu_3(s) = 1$ außer der eventuell existierenden Kante

$$S_3 \xrightarrow{1} T_3.$$

Resultat: $a' := a_4$ erfüllt die Aussage des Lemmas.

Endliche Akzeptoren (14)

Definition: Der Akzeptor a heißt **vollständig** $:\Longleftrightarrow$ es gelten

(1) $\mu(s) \in X \cup \{1\}$ für alle Kanten s. Aus $\mu(s) = \mu(s') = 1$ folgt $s = s'$ und $Q(s) \in S, Z(s) \in T$.

(2) Zu jedem Knoten P und jedem $x \in X$ gibt es eine Kante s mit $Q(s) = P, \mu(s) = x$.

Lemma: Zu jedem Akzeptor a gibt es einen vollständigen Akzeptor a' mit $L_a = L_{a'}$.

Beweis: Falls a nicht vollständig ist, führe einen neuen Knoten P_∞ ein und von jedem Knoten P aus, von dem keine Kante s mit $\mu(s) = x$ ausgeht, eine Kante

$$P \xrightarrow{x} P_\infty$$

Endliche Akzeptoren (15)

Weiter auch Kanten

$$P_\infty \xrightarrow{x} P_\infty$$

für $x \in X$ (d.h. P_∞ kann man nicht verlassen). Die Iteration dieses Schrittes führt zu einem vollständigen Automaten a' mit $L_{a'} = L_a$.

Definition: a heißt deterministischer Akzeptor $:\Longleftrightarrow$ zu jedem Knoten P und $x \in X$ gibt es genau eine Kante s mit $Q(s) = P$ und $\mu(s) = x$.

Lemma: Zu jedem Akzeptor a gibt es einen vollständigen deterministischen Akzeptor $\tilde{a}$ mit $L_a = L_{\tilde{a}}$.

Beweis: Wir dürfen annehmen, daß a vollständig ist und daß $\mu(s) \in X$ für $s \in E(G)$ gilt.

Endliche Akzeptoren (16)

Konstruktion von $\tilde{a}$.

Wir setzen

$$\tilde{K} := 2^K.$$

Sind $Z, Z' \in \tilde{K}$, so gibt es genau dann eine Kante $\tilde{s}$ mit $\tilde{\mu}(\tilde{s}) = x \in X$, wenn es zu jedem $z' \in Z'$ eine Kante $s \in K$ gibt mit $Q(s) \in Z$, $Z(s) = z'$ und $\mu(s) = x$. Da wir voraussetzen, daß a vollständig ist, gibt es dann auch zu jedem $z \in Z$ eine Kante s mit $Q(s) = z$ und $Z(s) \in Z'$ und $\mu(s) = x$.

$$Z = \left\{ \begin{array}{c} z_1 \\ \vdots \\ z_i \\ \vdots \\ z_k \end{array} \right\} \xrightarrow[\tilde{s}]{x} \left\{ \begin{array}{c} z'_1 \\ \vdots \\ z'_j \\ \vdots \\ z'_l \end{array} \right\} = Z'$$

Endliche Akzeptoren (17)

Es gibt genau dann eine Kante $\tilde{s}$

$$Q(\tilde{s}) = S \in \tilde{K},\ Z(\tilde{s}) = T \in \tilde{K},\ \tilde{\mu}(\tilde{s}) = 1,$$

wenn es eine Kante s in E gibt mit $Q(s) \in S$, $Z(s) \in T$ und $\mu(s) = 1$.

Damit haben wir den Graphen $\tilde{G}$ und die Markierung $\tilde{\mu}$ definiert. Wir setzen weiter

$$\tilde{S} := S \text{ und } \tilde{T} = \{Z \in \tilde{K} \mid Z \cap T \neq \emptyset\}$$

und

$$\tilde{a} := (\mathcal{W}(\tilde{G}), \tilde{S}, \tilde{T}, \tilde{\mu}, X)$$

und verwenden die Bezeichnung $2^a := \tilde{a}$.

Endliche Akzeptoren (18)

Wir beweisen $L_{\tilde{a}} = L_a$.

(1) Zu jedem $u \in L_a$ gibt es ein $w \in \mathcal{W}(G)(S,T)$ mit $\mu(w) = u$. Es gibt also $s \in S$ und $t \in T$ mit

$$s \underset{s_1}{\overset{x_1}{\longrightarrow}} z_1 \underset{s_2}{\overset{x_2}{\longrightarrow}} z_2 \underset{s_3}{\overset{x_3}{\longrightarrow}} \ldots \underset{s_n}{\overset{x_n}{\longrightarrow}} t,$$

worin $u = x_1 \cdot \ldots \cdot x_n$ und $\mu(s_i) = x_i$ ist. Hieraus folgt unmittelbar, daß es einen Weg $\tilde{w} = (\tilde{s}_1, \ldots, \tilde{s}_n) \in \mathcal{W}(\tilde{G})(\tilde{S}, \tilde{T})$ mit $\tilde{\mu}(\tilde{w}) = u$ gibt. Also ist $L_a \subset L_{\tilde{a}}$.

Endliche Akzeptoren (19)

(2) Zu $u \in L_{\tilde{a}}$, $|u| = n$, gibt es einen Weg $\tilde{w} = (\tilde{s}_1, \ldots, \tilde{s}_n) \in \mathcal{W}(\tilde{G})(\tilde{S}, \tilde{T})$ mit $\tilde{\mu}(\tilde{w}) = u$. Da a als vollständig vorausgesetzt wird, gibt es auch zu jedem $P \in S$ einen Weg $w = (s_1, \ldots, s_n)$ in $\mathcal{W}(G)$ mit $Q(w) = P$, und $\mu(w) = u$. Aufgrund der Definition von $\tilde{a}$ folgt $Z(w) \in Z(\tilde{w}) \in \tilde{T}$. Es braucht aber **nicht** $z(w) \in T$ zu gelten! Wir finden einen geeigneten Weg w mit $z(w)$, indem wir $\tilde{z} \in T \cap Z(\tilde{w})$ auswählen. Nun gibt es aufgrund der Definition von $\tilde{G}$ eine Kante s_n mit $\mu(s_n) = x$ und $Z(s_n) = \tilde{z}$. Ebenso findet man s_{n-1} als Kante mit $\mu(s_{n-1}) = x_{n-1}$ und $Z(s_{n-1}) = Q(s_n) \in Z(\tilde{s}_{n-1})$ usw. Der so zu $\tilde{w}$ konstruierte Weg w liegt in $\mathcal{W}(G)(S, T)$ und erfüllt die Forderung $\mu(w) = u$. Also ist $u \in L_a$ und also $L_{\tilde{a}} \subset L_a$.

Aus (1) und (2) folgt $L_a = L_{\tilde{a}}$.

Endliche Akzeptoren (20)

Bemerkung: Die Konstruktion hat $\#S = 1$ nicht verwendet und auch nicht, daß es keine Kante gibt, die in S einmündet. Unter Verwendung dieser Eigenschaft gilt für jeden Weg $w \in \mathcal{W}(\tilde{G})(\tilde{S}, Z)$ mit $|w| \geq 1$, daß $Z(w) \cap \tilde{S} = \emptyset$ ist. Das erlaubt es uns, die im Falle $1 \in L_{\tilde{a}}$ vorhandene $1-$Kante zu eliminieren, indem wir $\tilde{T}$ durch $\tilde{T} \cup \tilde{S}$ ersetzen.

Übung: Kontrolliere, ob sich dieser Beweis ohne die Voraussetzung der Vollständigkeit von a ebenso durchführen läßt.

Endliche Akzeptoren (21)

Lemma: Sei a vollständig und deterministisch und sei $\tilde{a}$ ein Akzeptor, der sich von a nur dadurch unterscheidet, daß $\tilde{T} = K - T$ ist. Es gilt dann $L_{\tilde{a}} = X^* - L_a$.

Beweis: Ist $u \in L_a$, dann gibt es in a einen Weg $w \in \mathcal{W}(G)(S,T)$ mit $\mu(w) = u$ mit $Z(w) \in T$. Da a deterministisch ist, ist w der einzige solche Weg. Also gibt es keinen Weg $w' \in \mathcal{W}(G)(S, K - T)$ mit $\mu(w') = u$. Also ist $u \notin L_{\tilde{a}}$. Da die Definitionen von a und $\tilde{a}$ bezüglich der Komplementbildung symmetrisch sind, gilt auch umgekehrt $u \in L_{\tilde{a}} \Rightarrow u \notin L_a$. Also ist $L_{\tilde{a}} = X^* - L_a$.

Fassen wir die gesamte Konstruktion ins Auge, dann gilt also: Zu jedem endlichen Akzeptor a gibt es einen vollständigen deterministischen Akzeptor $\tilde{a}$ mit $L_{\tilde{a}} = X^* - L_a$, womit unser Satz auf Blatt 26 bewiesen ist.

Endliche Akzeptoren (22)

Beobachtung: Setzen wir

$$|a| = \#K,$$

dann gilt für unsere Konstruktion

$$|\tilde{a}| = 2^{|a|}.$$

Es mag aber nun sein, daß es zu a einen deterministischen Akzeptor $\tilde{a}$ gibt, mit

$$|a| = |\tilde{a}|.$$

Das ist der Fall, wenn bereits a vollständig und deterministisch ist. In diesem Fall erzeugt unsere Konstruktion sehr viele Zustände, die von keinem von $\tilde{S}$ ausgehenden Weg erreicht werden. Diese Zustände sind also überflüssig. Hierdurch stellt sich die

Frage: Gibt es eine Folge $a_1, a_2, a_3, \ldots$ von Akzeptoren über dem Alphabet X, so daß für jede Folge $\tilde{a}_1, \tilde{a}_2, \tilde{a}_3, \ldots$ von vollständigen deterministischen Akzeptoren mit $L_{a_i} = L_{\tilde{a}_i}$ für $i = 1, 2, 3, \ldots$

$$|\tilde{a}_i| \approx 2^{|a_i|}$$

gilt.

Endliche Akzeptoren (23)

Wir betrachten hierzu ein Beispiel von M. Fischer. (Siehe: G. Hotz: Einführung in die Informatik, Teubner-Verlag)

Sei

$$L_n = \{u \cdot v \in \{a,b\}^* \mid |u| = |v| = n, u \neq v\}.$$

Wir wollen die Akzeptoren aus einfacheren Akzeptoren aufbauen. Hierzu definieren wir

$$L_n^i = \{u_1 a u_2 v_1 b v_2,\ u_1 b u_2 v_1 a v_2 \mid \\ i = |u_1| = |v_1|, |u_2| = |v_2| = n - i - 1\}$$

Es gilt offensichtlich

$$L_n = L_n^0 \cup L_n^1 \cup \cdots \cup L_n^{n-1}$$

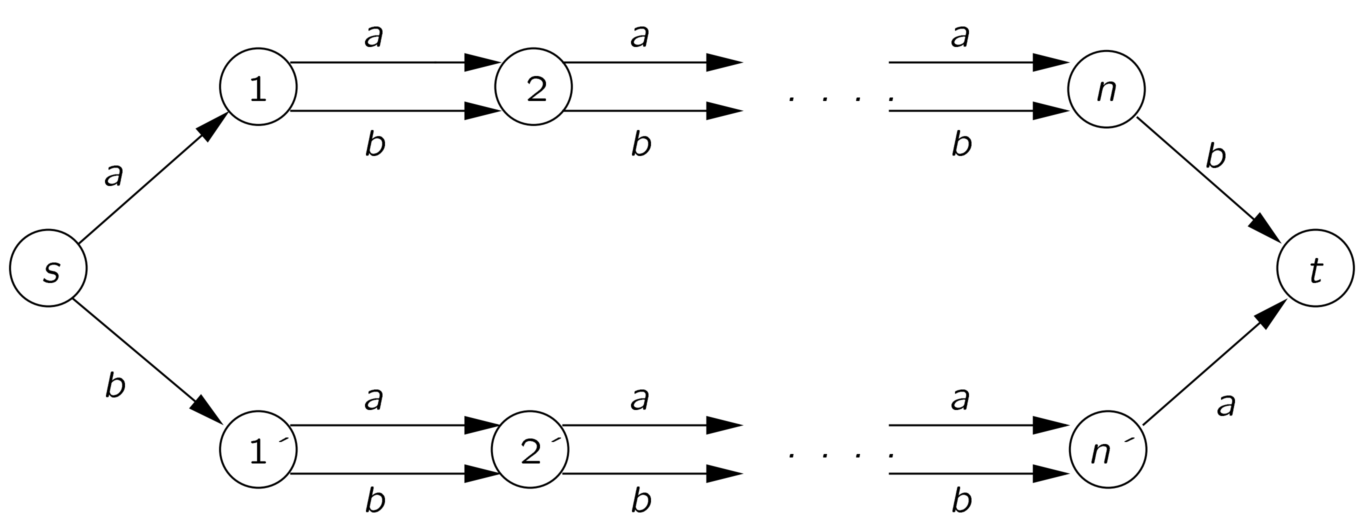
Endliche Akzeptoren (24)

Weiter gilt für $i = 0, \ldots, n-1$

$$\begin{aligned} L_n^i &= \{a,b\}^i \cdot a \cdot \{a,b\}^{n-1} \cdot b \cdot \{a,b\}^{n-i-1} \\ &\cup \{a,b\}^i \cdot b \cdot \{a,b\}^{n-1} \cdot b \cdot \{a,b\}^{n-i-1} \\ \\ &= \{a,b\}^i \cdot \Big(a \cdot \{a,b\}^{n-1} \cdot b \cup \\ & \quad b \cdot \{a,b\}^{n-1} \cdot a\Big) \cdot \{a,b\}^{n-i-1} \end{aligned}$$

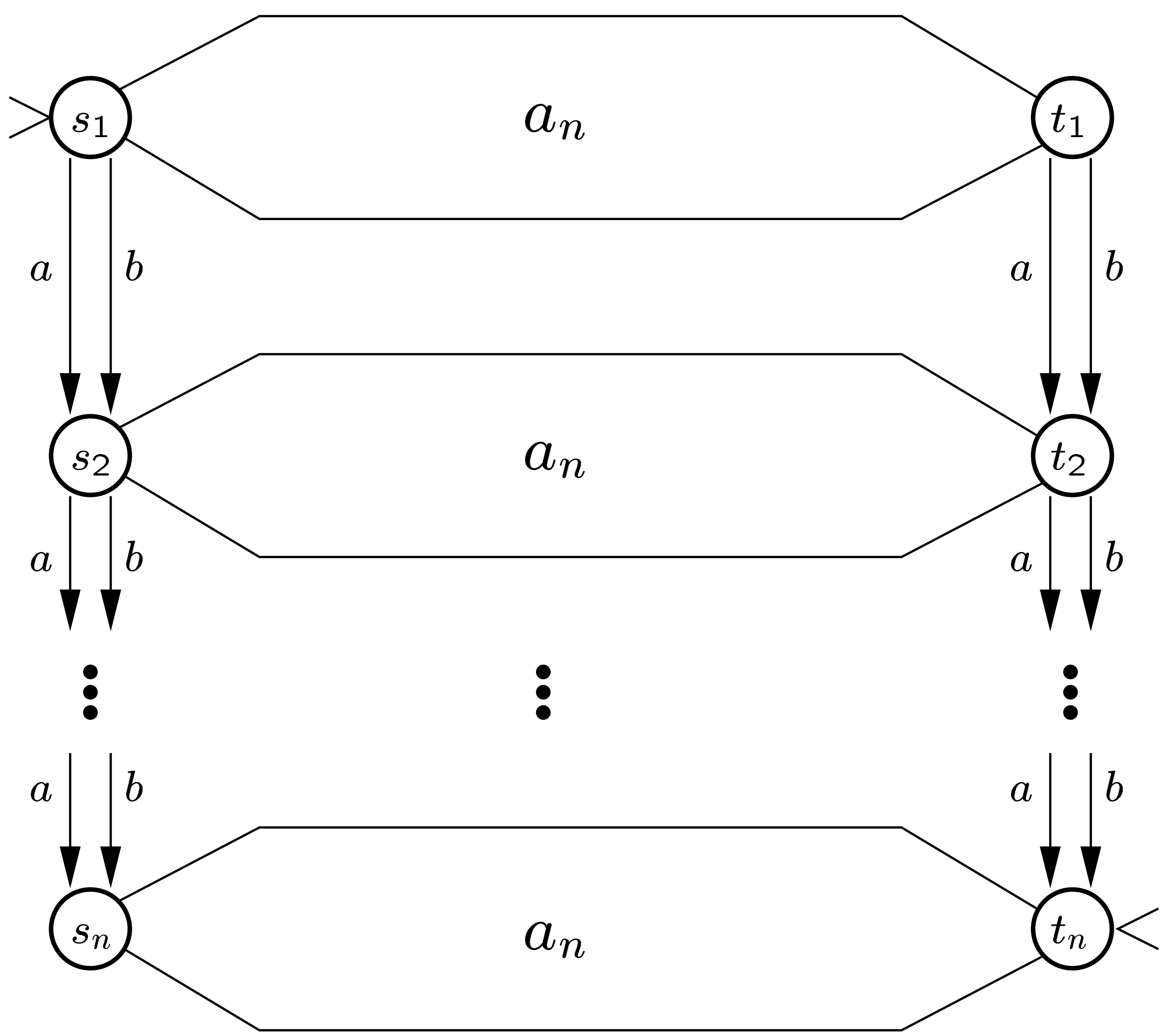
Weiter gilt natürlich $\{a,b\}^{i+1} = \{a,b\}^i \cdot \{a,b\}$.

Wir geben zunächst einen Akzeptor a_n für $a \cdot \{a,b\}^{n-1} \cdot b \cup b \cdot \{a,b\}^{n-1} \cdot a$ an:



Endliche Akzeptoren (25)

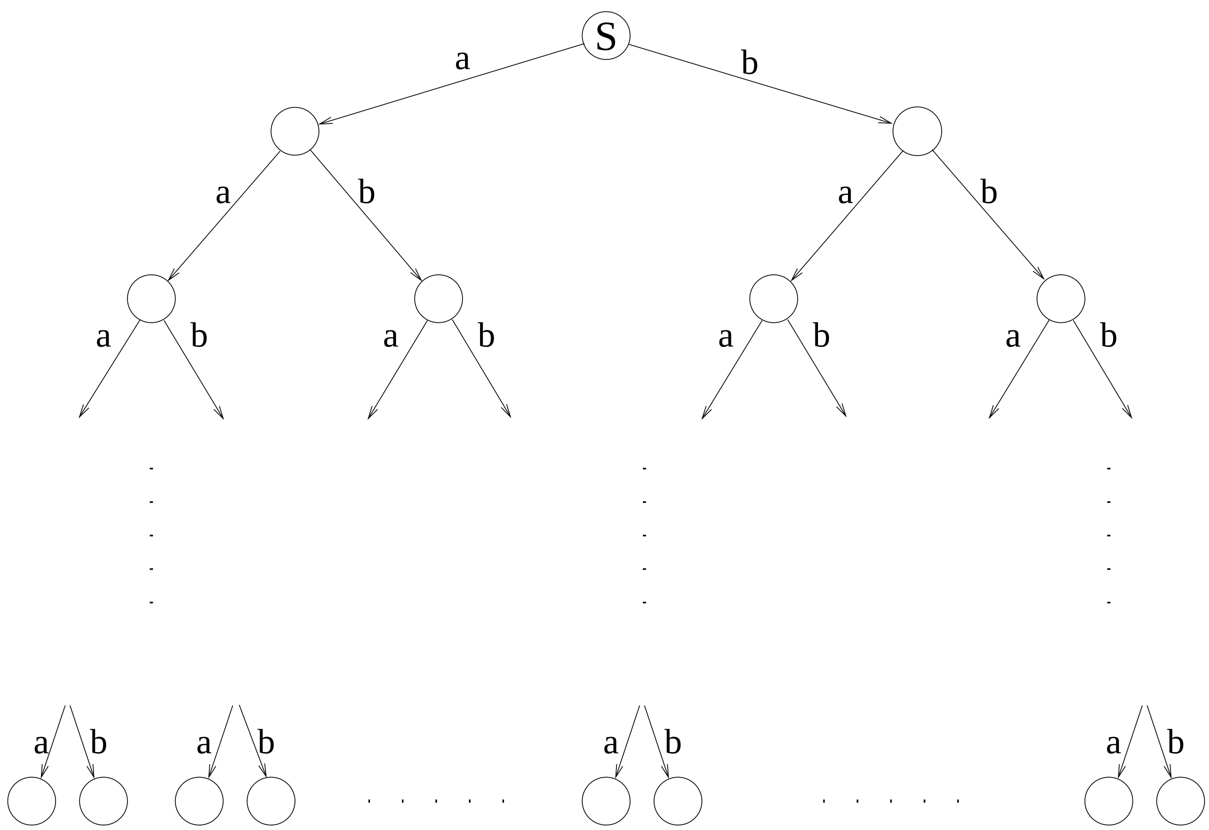
Nun verwenden wir n paarweise disjunkte Exemplare von a_n, um einen Akzeptor für L^n zu konstruieren. Hierbei verwenden wir die Assoziativität des Komplexproduktes und erhalten einen matrixartigen Akzeptor, der unten schematisch dargestellt wird.



Endliche Akzeptoren (26)

Offensichtlich gilt $|A| = n \cdot (2 \cdot n + 2)$. A ist nichtdeterministisch.

Der am einfachsten zu beschreibende deterministische Akzeptor B ist ein balancierter binärer Baum der Tiefe $2n$, dessen Wurzel der Startknoten ist.



Endliche Akzeptoren (27)

Markieren wir die Endknoten des Baumes als Terminalknoten, die zu den Wegen $w \in \mathcal{W}(G)$ mit $\mu(w) \in L_n$ gehören, dann erhalten wir einen Akzeptor für L_n.

Es stellt sich die Frage, ob einige der Kanten von B überflüssig sind.

Zwei Knoten P_1 und P_2 des Baumes kann man identifizieren, wenn die von P_1 und P_2 ausgehenden Wege bei gleicher Markierung zu gleichem Ergebnis führen: akzeptiert bzw. abgelehnt. Das ist dann der Fall, wenn in Punkt P_1 und P_2 schon entschieden ist, daß alle sich an P_1 und P_2 anschließenden Wege akzeptiert werden. Dieser Fall liegt genau dann vor, wenn der Weg von $w \in \mathcal{W}(S, P_1)$ an zwei „korrespondierenden“ Stellen verschiedene Markierungen enthält.

Endliche Akzeptoren (28)

Diese Frage kann aber nicht entschieden werden, wenn es noch zu keiner Stelle des Weges eine korrespondierende Stelle gibt. Hieraus ergibt sich, daß man kein Punktepaar eines deterministischen Akzeptors identifizieren darf, das von S einen Abstand $\leq n$ besitzt. Hieraus folgt, daß für jeden vollständigen deterministischen Akzeptor $\tilde{a}$ die Abschätzung $|\tilde{a}| \geq 2^n$ gilt. Wir haben damit ein Beispiel gefunden, für das

$$|\tilde{a}| \geq 2^{\sqrt{|a|}}$$

gilt und zwar für **alle** deterministischen Akzeptoren von L_n, $n = 1, 2, \ldots$

Der minimale deterministische Akzeptor einer regulären Sprache L kann also wesentlich größer sein als ein nichtdeterministischer Akzeptor von L.

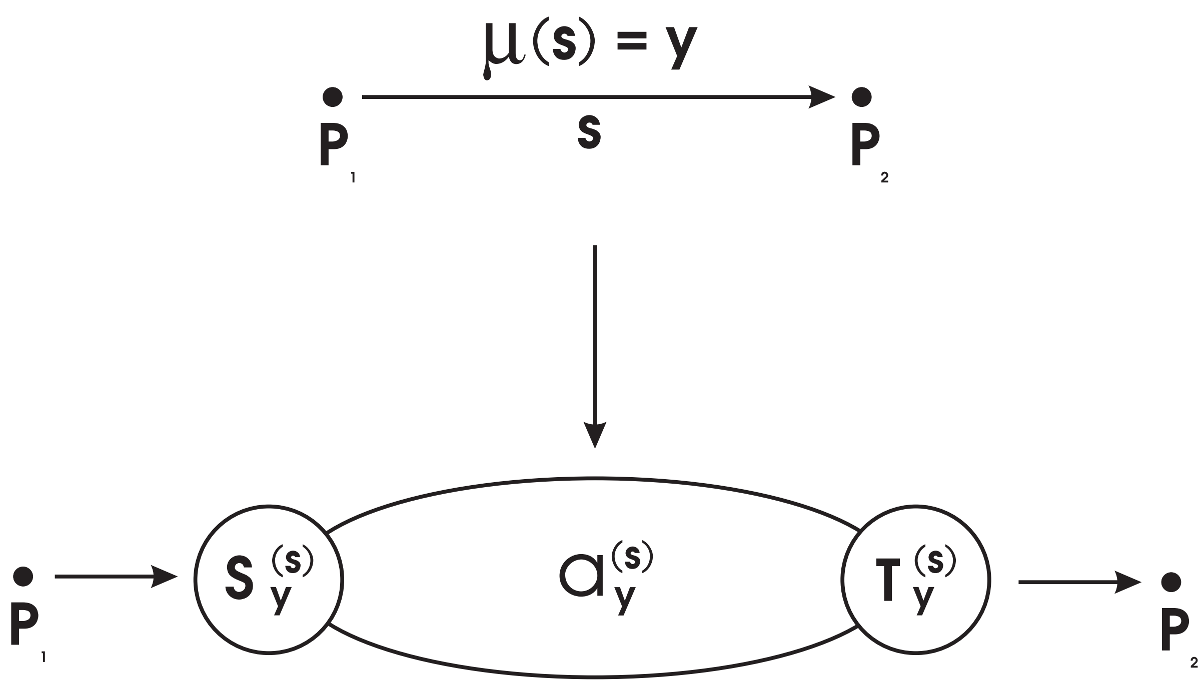
Endliche Akzeptoren (29)

Lemma: Ist $h : Y^* \longrightarrow REG(X^*)$ ein Monoidhomomorphismus und ist $L \in REG(Y^*)$, dann ist $h(L) \in REG(X^*)$; hierin ist $h(L) := \bigcup_{w \in L} h(w)$.

Beweis: Wir wählen zu L einen vollständigen Akzeptor a mit $L_a = L$. Zu jedem $y \in Y$ wählen wir einen Akzeptor a_y für $h(y)$ mit $\#S_y = \#T_y = 1$. Zu jeder Kante s von a mit $\mu(s) = y$ wählen wir ein zu a_y isomorphes Exemplar $a_y^{(s)}$, so daß $a_y^{(s)}$ und $a_y^{(s')}$ keine Knoten gemeinsam haben.

Nun hängen wir in den Graphen von a für jede Kante s, $\mu(s) = y$, $a_y^{(s)}$ ein, wie es das folgende Diagramm beschreibt.

Endliche Akzeptoren (30)



Resultat $\tilde{a}$. Es gilt $L_{\tilde{a}} = h(L)$.

Durch Homom. erkennbare Mengen (1)

Definition: Sei a ein Akzeptor und P_1, P_2 zwei Knoten in a. P_1 und P_2 heißen genau dann äquivalent in Bezug auf a, wenn es zu jedem Weg $w_1 \in \mathcal{W}(G)(P_1, T)$ einen Weg $w_2 \in \mathcal{W}(G)(P_2, T)$ gibt, mit $\mu(w_1) = \mu(w_2)$ und umgekehrt auch.

Lemma: Seien P_1, P_2 bezüglich a äquivalente Punkte. a' ergebe sich aus a durch Identifikation von P_1 und P_2. Es gilt dann $L_a = L_{a'}$.

Beweis:

(1) $L_a \subset L_{a'}$, da jeder Weg in a auch ein Weg in a' ist.

(2) $L_{a'} \subset L_a$. Betrachte

$$\begin{array}{cccc} S \xrightarrow{w_1} & P_1 & & \\ & P_2 & \xrightarrow{w_2} & T. \end{array}$$

$w_1 \cdot w_2$ ist ein Weg in $\mathcal{W}(G')$. Aus der Äquivalenz von P_1, P_2 folgt leicht die Behauptung.

Durch Homom. erkennbare Mengen (2)

Definition: a' heißt eine Reduktion von a, wenn a' aus a durch Identifikation äquivalenter Knoten erhalten werden kann.

Korollar: Ist a' eine Reduktion von a, dann gilt $L_a = L_{a'}$.

Definition: a heißt reduziert $:\Longleftrightarrow$ sind P_1, P_2 äquivalente Knoten aus a, dann ist $P_1 = P_2$.

Korollar: Zu jedem Akzeptor a gibt es einen reduzierten Akzeptor a' mit $L_a = L_{a'}$.

Lemma: Ist a ein vollständiger deterministischer Akzeptor und ist a' ein zu a gehöriger reduzierter Akzeptor, dann ist auch a' vollständig und deterministisch.

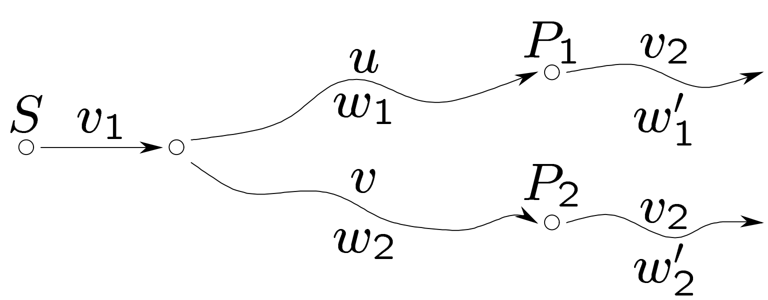
Zeige: Sind P_1, P_2 äquivalent in a, dann gilt das auch für die Nachfolgeknoten P_1', P_2'

$$P_1 \xrightarrow{x} P_1', \quad P_2 \xrightarrow{x} P_2'.$$

Durch Homom. erkennbare Mengen (3)

Lemma: Sei a ein vollständiger deterministischer reduzierter Akzeptor. Ist $u \equiv v\ (L_a)$, dann gilt für Wege w_1, w_2 mit $Q(w_1) = Q(w_2)$ und $\mu(w_1) = u$ und $\mu(w_2) = v$, daß $Z(w_1) = Z(w_2)$ ist.

Erläuterung:



Für alle v_2 gilt $Z(w_1') \in T \Leftrightarrow Z(w_2') \in T$, d.h. P_1 äq. P_2.

Durch Homom. erkennbare Mengen (4)

Satz: $L \subset X^*$ ist regulär $\iff$ $synt(L)$ ist endlich.

Beweis:

(1) Wir haben $synt(L)$ endlich $\Rightarrow L \in REG$ bereits gezeigt.

(2) $L \in REG \Rightarrow synt(L)$ endlich folgt aus dem vorhergehenden Lemma.

Hinweis: Es gilt unter den Voraussetzungen des Lemmas stets: Aus $u \equiv v(L_a)$ folgt

$$\begin{tikzcd} \circ \arrow[r, bend left, "u"] \arrow[r, bend right, "v"'] & \circ \end{tikzcd}$$

Durch Homom. erkennbare Mengen (5)

Korollar: Zu jeder Sprache $L \in REG(X^*)$ gibt es ein endliches Monoid M und einen Homomorphismus $h : X^* \to M$, so daß für eine geeignete Teilmenge $U \subset M$

$$L = h^{-1}(U)$$

gilt.

Beweis: Wähle $M = synt(L)$ und $U = \{[u] \mid u \in L\}$.

Die Umkehrung dieses Korollars haben wir bereits gezeigt. Somit gilt der

Satz: $L \in REG(X^*) \iff$ es gibt ein endliches Monoid M, eine Teilmenge $U \subset M$ und einen Monoidhomomorphismus $h : X^* \to M$, mit $h^{-1}(U) = L$.

Korollar: Ist $h : Y^* \to X^*$ ein Homomorphismus, dann gilt $h^{-1}(L) \in REG(Y^*)$. für $L \in REG(X^*)$.

Durch Homom. erkennbare Mengen (6)

Diese Charakterisierung der regulären Mengen gibt Anlaß zu der folgenden Definition.

Definition: Ist M ein beliebiges Monoid und $L \subset M$, dann heißt L *recognizable* (durch einen Homomorphismus erkennbar) $:\Longleftrightarrow$ Es gibt einen Homomorphismus $h : M \to M'$ in ein endliches Monoid M' und eine Teilmenge $U \subset M'$ mit $h^{-1}(U) = L$.

Notation: $L \in REC(M)$.

Wir haben gezeigt, daß der

Satz: $REG(X^*) = REC(X^*)$ gilt.

Rationale Mengen über X^* (1)

Definition: Ist M ein Monoid, dann heißt $R \subset 2^M$ **rational abgeschlossen** $:\Longleftrightarrow$ R ist abgeschlossen unter der Vereinigung $\bigcup$ von Mengen, dem Komplexprodukt $\cdot$ und dem $*-$Operator.

$RAT(M)$ ist die kleinste, rational abgeschlossene Teilmenge von 2^M, die die Mengen $\{m\}, m \in M$ und $\emptyset$ enthält.

Wir haben bereits gezeigt, daß

$$RAT(X^*) \subset REG(X^*)$$

gilt. Wir wollen nun auch die umgekehrte Relation zeigen.

Satz (Kleene): $REG(X^*) = RAT(X^*)$.

Rationale Mengen über X^* (2)

Beweis: Wir haben

$$L \in REG(X^*) \Rightarrow L \in RAT(X^*)$$

zu zeigen.

$L \in REG(X^*) \Rightarrow$ Es gibt einen endlichen Akzeptor a mit $L = L_a$.

Man zeigt, daß L aus $\emptyset, \{\{x\} \mid x \in X\}$ mittels $\bigcup, \cdot, *$ erzeugt werden kann.

Der Beweis erfolgt durch Induktion nach der Anzahl k der Kanten bei fester Knotenzahl. O.B.d.A. $\#S = 1$, $S \cap T = \emptyset$.

(1) $k = 0 \Rightarrow L_a = \emptyset$, d.h. $L_a \in RAT(X^*)$.

(2) Induktionsschritt: Sei die Behauptung für Akzeptoren mit k Kanten bewiesen, und sei a_k ein solcher Akzeptor. G_k sei der Graph von a_k. Wir fügen zwischen zwei Knoten P_1, P_2 von G_k eine neue Kante s ein, mit $Q(s) = P_1$, $Z(s) = P_2$, $\mu(s) = x$. Der neue Akzeptor sei a_{k+1}.

Rationale Mengen über X^* (3)

Wir haben dann

$$\mathcal{W}(G_{k+1})(S,T) = \mathcal{W}(G_k)(S,T) \cup$$
$$\mathcal{W}(G_k)(S,P_1)\cdot(s\cdot\mathcal{W}(G_k)(P_2,P_1))^*\cdot s\cdot\mathcal{W}(G_k)(P_2,T)$$

$$\begin{tikzcd}
 & P_1 \arrow[r, "s"'] & P_2 \arrow[l, bend right, "\mathcal{W}(G_k)(P_2,P_1)"'] \arrow[dr, bend left, "\mathcal{W}(G_k)(P_2,T)"] & \\
S \arrow[ur, bend left, "\mathcal{W}(G_k)(S,P_1)"] \arrow[rrr, "\mathcal{W}(G_k)(S,T)"'] & & & T
\end{tikzcd}$$

Nach Induktionsannahme gilt

$$\begin{array}{rcl}
\mu(\mathcal{W}(G_k)(S,T)) & \in & RAT(X^*), \\
\mu(\mathcal{W}(G_k)(S,P_1)) & \in & RAT(X^*), \\
\mu(\mathcal{W}(G_k)(P_2,T)) & \in & RAT(X^*), \\
\mu(\mathcal{W}(G_k)(P_2,P_1)) & \in & RAT(X^*), \\
\mu(s) & \in & RAT(X^*).
\end{array}$$

Da $RAT(X^*)$ abgeschlossen ist unter $\cup, \cdot, *$, folgt die Behauptung.

Rechtslineare Sprachen $r - LIN(X^*)$ (1)

N und X sind Alphabete, $A := N \cup X$. Wir setzen $N \cap X = \emptyset$ voraus. Im folgenden heißen endliche Teilmengen $\mathfrak{P} \subset N^* \times A^*$ Produktionssysteme. Weiter ist $S \in N$, und $G = (N, X, \mathfrak{P}, S)$ bezeichnen wir als Chomsky-Grammatik. N heißt Variablen- und X Terminalalphabet.

Ist $(p, q) \in \mathfrak{P}$, dann schreiben wir auch

$$p \xrightarrow{\mathfrak{P}} q \quad \text{oder kurz} \quad p \longrightarrow q.$$

Wir setzen die Relation $\mathfrak{P}$ wie folgt zu einer Relation $\mathfrak{P}^*$ fort. $\mathfrak{P}^*$ ist minimal mit

(1) $w \xrightarrow{\mathfrak{P}^*} w$, für $w \in A^*$,

(2) $u \xrightarrow{\mathfrak{P}^*} v \Rightarrow wuw' \xrightarrow{\mathfrak{P}^*} wvw'$,

(3) $w_1 \xrightarrow{\mathfrak{P}^*} w_2,\ w_2 \xrightarrow{\mathfrak{P}^*} w_3 \Rightarrow w_1 \xrightarrow{\mathfrak{P}^*} w_3$.

Anstelle von $w \overset{\mathfrak{P}^*}{\to} w'$ schreiben wir oft $w \overset{*}{\to} w'$.

Rechtslineare Sprachen $r - LIN(X^*)$ (2)

Im Falle

$$w = upv \xrightarrow{*} uqv = w'$$

mit $(p,q) \in \mathfrak{P}$ schreiben wir auch $\longrightarrow$, um auszudrücken, daß w durch eine **einmalige** Anwendung von $\mathfrak{P}$ in w' übergeführt werden kann.

Eine Folge

$$w = w_1 \longrightarrow w_2 \longrightarrow \ldots \longrightarrow w_k = w'$$

heißt **Ableitung** von w' aus w. $k-1$ heißt die **Länge** der Ableitung.

$$L(G) := \{w \in X^* \mid S \xrightarrow{*} w\}$$

heißt „die von G erzeugte Sprache“.

Notiz: Chomsky versuchte, Grammatiken natürlicher Sprachen mittels „Chomsky“-Grammatiken zu modellieren.

Hier betrachten wir zunächst nur den Sonderfall

$$\mathfrak{P} \subset N \times N \cdot X \cup N \times (X \cup \{1\}).$$

G heißt in diesem Fall **rechtslinear**.

Rechtslineare Sprachen $r - LIN(X^*)$ (3)

Definition: $L \subset X^*$ heißt rechtslineare Sprache $:\iff$ Es gibt eine rechtslineare Chomsky-Grammatik G mit $L = L(G.)$ $r - LIN(X^*) := \{L \subset X^* | \ L$ ist rechtslineare Sprache$\}$.

Satz: $r - LIN(X^*) = REG(X^*)$.

Beweis: (1) $r - LIN(X^*) \subset REG(X^*)$.

Setze:

$$\begin{array}{rcl} K & := & N \cup \{t\}, t \notin N, \\ E & := & \{(n, n', x) \mid (n, x \cdot n') \in \mathfrak{P}\} \\ & \cup & \{(n, t, x) \mid (n, x) \in \mathfrak{P}\}. \end{array}$$

$$Q(n,t,x) := n, Z(n,t,x) := t, \mu(n,t,x) := x$$

$$Q(n,n',x) := n, Z(n,n',x) := n', \\ \mu(n,n',x) := x.$$

$$G := (E, K, Q, Z),$$

$$a := (\mathcal{W}(G), S, t, \mu, X).$$

Rechtslineare Sprachen $r - LIN(X^*)$ (4)

Man zeigt nun $L_a = L(G)$.

(2) $REG(X^*) \subset r - LIN(X^*)$.

Sei $L \in REG(X^*)$.

Wir konstruieren zu a eine rechtslineare Grammatik: O.B.d.A. nehmen wir an, daß a vollständig ist. Wir setzen

$$\begin{array}{rcl} N & := & K, \\ \mathfrak{P} & := & \{(z, x \cdot z') \in N \times N \cdot X \mid \text{ Es gibt } s \in E \\ & & \text{mit } Q(s) = z, Z(s) = z', \mu(s) = x\} \\ & \cup & \{(z, x) \in N \times X \mid \text{ Es gibt } s \in E \\ & & \text{mit } Q(s) = z, Z(s) \in T, \mu(s) = x\} \\ & \cup & \{(s, 1) \mid \text{ für } 1 \in L_a\}. \end{array}$$

Für die Grammatik

$$G := (N, X, \mathfrak{P}, S)$$

gilt

$$L(G) = L_a.$$

Rechtslineare Sprachen $r - LIN(X^*)$ (5)

Beispiel 1: $S \longrightarrow x \cdot n$, $n \longrightarrow y \cdot n$, $n \longrightarrow x \cdot n'$, $n' \longrightarrow y \cdot n'$, $n' \longrightarrow x$

Akzeptor:

$$\begin{tikzcd} S \arrow[r, "x"] & n \arrow[loop above, "y"] \arrow[r, "x"] & n' \arrow[loop above, "y"] \arrow[r, "x"] & t \end{tikzcd}$$

Die Sprache: $L = xy^*xy^*x$.

Beispiel 2: Ergänze die obige Grammatik durch die Produktion $t \longrightarrow S$.

Zugehöriger Akzeptor:

$$\begin{tikzcd} S \arrow[r, "x"] & n \arrow[loop above, "y"] \arrow[r, "x"] & n' \arrow[loop above, "y"] \arrow[r, "x"] & t \arrow[lll, bend left, "1"] \end{tikzcd}$$

Die akzeptierte Sprache: $L = (xy^*xy^*x)$

Rechtslineare Sprachen $r - LIN(X^*)$ (6)

Korollar: $REG(X^*) = l - LIN(X^*)$.

Beweis: Die Definition von linkslinear ist hinsichtlich der Produktionen $z \longrightarrow z \cdot x$ spiegelbildlich aufgebaut. Ersetzt man in dem Beweis zu obigem Satz überall $z \longrightarrow z' \cdot x$ durch $z \longrightarrow x \cdot z'$, dann ergibt sich der Beweis für das Korollar.

Zusammenfassung

Hauptsatz (1)

$REG(X^*) = RAT(X^*) = REC(X^*) = r - LIN(X*) = l - LIN(X^*)$. $REG(X^*)$ ist abgeschlossen unter

$$\cup, \cdot, *, ^-, \cap, hom, hom^{-1}$$

und

$$hom : X^* \longrightarrow REG(X^*).$$

Jede dieser Sprachklassen legt andere Verallgemeinerungen nahe.

Zusammenfassung (1)

(1) REG : Hier liegen Verallgemeinerungen in zwei Richtungen nahe:

(a) Man ersetzt das Monoid X^* durch freie Gruppen oder freie kommutative Monoide oder andere Monoide.
Berstel, J.: Transductions and Contextfree Languages. Teubner, Stuttgart (1979), Hotz, G./Estenfeld, K.: Formale Sprachen. B I Wissenschaftsverlag.

(b) Man verallgemeinert die Wegekategorie. Anstelle der Wege treten z.B. Bäume oder allgemeiner Netze.
Hotz, G.: Hierarchischer Entwurf komplexer Systeme, In: Wegener, I. (Ed.): Highlights aus der Informatik. Springer-Verlag (1996)

(2) RAT : Man setzt anstelle von X^* beliebige Monoide oder Algebren.

Zusammenfassung (2)

(3) REC : Man ersetzt in den Darstellungen $hom^{-1}(M)$, das endliche Monoid durch einfach berechenbare unendliche Monoide. Das syntaktische Monoid $synt(D(Z))$ der Dycksprache ist dazu ein Beispiel.

(4) $r - LIN$: Hier geht man von der rechts- oder linkslinearen Chomskysprache zu allgemeineren Grammatikformen über.

Die für $REG(X^*)$ festgestellten Gleichheiten zwischen den aufgeführten Konzepten zur Definition von Sprachen bleiben bei diesen Verallgemeinerungen i.a. nicht erhalten.

Komplexitätsaspekte (1)

Wir haben gezeigt, daß $REG(X^*) = RAT(X^*)$ ist. Also kann man jede reguläre Sprache durch eine Folge rationaler Ausdrücke definieren.

Formal ist ein rationaler Ausdruck über einer Kleene-Algebra ein Wort über dem Alphabet

$$A = X \cup \{\emptyset\} \cup \{(,)\} \cup \{\cup, \cdot, *\},$$

das induktiv wie folgt definiert wird.

- $\emptyset, x$ für $x \in X$ sind rationale Ausdrücke.

- Sind ϱ_1, ϱ_2 rationale Ausdrücke, dann auch $(\varrho_1 \cup \varrho_2), (\varrho_1 \cdot \varrho_2), (\varrho_1^*)$.

- Jeder rationale Ausdruck läßt sich aus den elementaren Ausdrücken durch die Anwendung der im vorigen Punkt angegebenen Schritte erzeugen.

Komplexitätsaspekte (2)

Wir messen die Größe der rationalen Ausdrücke durch die Anzahl der Positionen in dem Ausdruck, auf denen Operationszeichen stehen.

Formal: Wir definieren einen Homomorphismus $\lambda : A^* \longrightarrow A^*$ durch

$$\lambda(\alpha) = \begin{cases} 1 & \text{für} \quad \alpha \in X \cup \{\emptyset\} \cup \{(,)\} \\ \alpha & \text{für} \quad \alpha \in \{\cup, \cdot, *\}. \end{cases}$$

Nun definieren wir für rationale Ausdrücke u

$$|u|_\lambda := |\lambda(u)|.$$

Uns interessiert nun der Zusammenhang zwischen $|u|_\lambda$ und $|a|$, wenn u und a die gleiche Sprache definieren. Hierbei legen wir unsere Konstruktionen in den Beweisen zu

$REG(X^*) \subset RAT(X^*)$ und $RAT(X^*) \subset REG(X^*)$

zugrunde. Man erkennt

$$\begin{aligned} |a \cup b| &= |a| + |b| + 2, \\ |a \cdot b| &= |a| + |b|, \\ |a^*| &= |a| + 2. \end{aligned}$$

Komplexitätsaspekte (3)

Hieraus folgt:

Die angegebene Konstruktion von Akzeptoren a_u zu vorgegebenen rationalen Ausdrücken erfüllt

$$|a_u| \leq 2 \cdot |u|_\lambda.$$

Wir betrachten nun den Beweis zu $REG \subset RAT$, und geben eine Abschätzung für die Größe des in dem Beweis zu a konstruierten rationalen Ausdrucks u_a an.

Zu jeder Kante s aus dem Graphen von a haben wir einen rationalen Ausdruck der Form

$$\mathcal{W}_{k+1} = \mathcal{W}_k(S,T) \cup \mathcal{W}_k(S,P) \cdot (s \cdot \mathcal{W}_k(R,P))^* \cdot s \cdot \mathcal{W}_k(R,T)$$

angegeben. Dieser Ausdruck v_{k+1} hat die Größe

$$|v_{k+1}|_\lambda = 6,$$

wenn wir die $\mathcal{W}_k(P,R)$ einmal als Alphabetelemente ansehen.

Komplexitätsaspekte (4)

Indem man von $v_N, N = \#E(a)$ ausgehend rekursiv die jeweils benötigten Ausdrücke einsetzt, dann erhält man eine Folge von Ausdrücken, deren Größe exponentiell mit der Rekursionstiefe zunimmt.

Zur Erläuterung betrachten wir ein einfaches Beispiel. Sei a definiert durch

$$\begin{tikzcd} S \arrow[r, "s_1"] \arrow[rr, bend right, "s_4"'] & Q \arrow[r, "s_2"] \arrow[loop above, "s_3"] & T \end{tikzcd}$$

mit $\mu(s_1) = \mu(s_2) = x$, $\mu(s_3) = s(s_4) = y$.

Komplexitätsaspekte (5)

Wir nehmen der Reihe nach die Kanten s_1, s_2, s_3, s_4 auf. Wir benötigen

$$
\begin{aligned}
\mathcal{W}_4(S,T) &= \mathcal{W}_3(S,T) \cup \mathcal{W}_3(S,S) \\
&\quad \cdot (s_4 \cdot \mathcal{W}_3(T,S))^* \cdot s_4 \cdot \mathcal{W}_3(T,T) \\
\mathcal{W}_3(S,S) &= \mathcal{W}_2(S,S) \cup \mathcal{W}_2(S,Q) \\
&\quad \cdot (s_3 \cdot \mathcal{W}_2(Q,Q))^* \cdot s_3 \cdot \mathcal{W}_2(Q,T) \\
\mathcal{W}_3(S,T) &= \mathcal{W}_2(S,T) \cup \mathcal{W}_2(S,Q) \\
&\quad \cdot (s_3 \cdot \mathcal{W}_2(Q,Q))^* \cdot s_3 \cdot \mathcal{W}_2(Q,T) \\
\mathcal{W}_3(T,S) &= \mathcal{W}_2(T,S) \cup \mathcal{W}_2(T,Q) \\
&\quad \cdot (s_3 \cdot \mathcal{W}_2(Q,Q))^* \cdot s_3 \cdot \mathcal{W}_3(Q,S) \\
\mathcal{W}_3(T,T) &= \mathcal{W}_2(T,T) \cup \mathcal{W}_2(T,Q) \\
&\quad \cdot (s_3 \cdot \mathcal{W}_2(Q,Q))^* \cdot s_3 \cdot \mathcal{W}_2(Q,T).
\end{aligned}
$$

Wenn wir nun die für den Index $k = 3$ berechneten Ausdrücke in $\mathcal{W}_4(S,T)$ einsetzen und diese Konstruktion rekursiv fortsetzen, dann erhalten wir rationale Ausdrücke von exponentiell wachsender Größe.

Komplexitätsaspekte (6)

Betrachten wir aber für $k = 1$

$$\begin{aligned}
\mathcal{W}(S,Q) &= \mathcal{W}_0(S,Q) \cup \mathcal{W}_0(S,S) \\
&\quad \cdot (s_1 \cdot \mathcal{W}_0(Q,S)) \cdot s_1 \cdot \mathcal{W}_0(Q,Q) \\
&= \emptyset \cup \emptyset^* \cdot (s_1 \cdot \emptyset)^* \cdot s_1 \cdot \emptyset^*,
\end{aligned}$$

dann sehen wir, daß schließlich Ausdrücke entstehen, die sehr viele überflüssige Bestandteile enthalten. Und es ist leicht zu sehen, daß

$$\mathcal{W}_a(S,T) = s_1 \cdot s_3^* \cdot s_2 \cup s_4$$

ist. Unsere Konstruktion eines rationalen Ausdrucks, die einen leicht verständlichen Beweis liefert, taugt also nicht für praktische Anwendungen, und sie gibt uns keine interessante Auskunft über die Ausdruckskraft von rationalen Ausdrücken unter komplexitätstheoretischen Aspekten.

Komplexitätsaspekte (7)

Wir beobachten aber in dem Beispiel $\mathcal{W}(S,Q)$, daß man in den rationalen Ausdrücken Kürzungen zulassen sollte, die durch die Axiome

$$\emptyset \cup u = u, \quad \emptyset \cdot u = \emptyset, \quad \emptyset^* = 1, \quad 1 \cdot u = u$$

beschrieben werden und durch assoziative „Umklammerungen“

$$(u \cdot (v \cdot w)) = ((u \cdot v) \cdot w)$$

und distributives Ausmultiplizieren

$$(u \cdot (v \cup w)) = ((u \cdot v) \cup (u \cdot w)).$$

Diese hier nur angedeutete Algebra der rationalen Ausdrücke ist noch nicht hinreichend untersucht worden.

Komplexitätsaspekte (8)

Eine Bemerkung zur Erweiterung der Kleene-Algebra durch Hinzunahme der Komplementbildung als zusätzliche Operation.

Wir betrachten also $(REG(X^*), \cup, \cdot, *, ^{-})$, worin

$$\overline{L} = X^* - L$$

ist.

Was erwarten wir hier, wenn wir unseren Beweis zu $L \in REG(X^*) \Rightarrow \overline{L} \in REG(X^*)$ anschauen:

Ist a gegeben und nichtdeterministisch, dann müssen wir damit rechnen, daß für jeden Akzeptor $\overline{a}$ für $\overline{L}$

$$|\overline{a}| \geq c \cdot 2^{|a|}$$

gilt, wie in den Übungen gezeigt wurde.

Komplexitätsaspekte (9)

Unsere Konstruktion für $a \cup b$ führt zu einem nichtdeterministischen Akzeptor. Iteriert man also diese Konstruktion nach der Idee

$$a,\ b,\ \overline{a},\ \overline{b},\ \overline{a} \cup \overline{b},\ \overline{\overline{a} \cup \overline{b}}, \ldots,$$

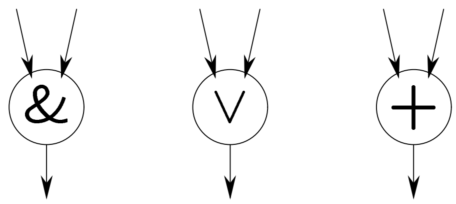
dann liefert unsere Konstruktion Akzeptoren a_i, deren Größe wie

$$n, 2^n, 2^{2^n}, \ldots, 2^{2^{\cdot^{\cdot^{\cdot^{2^n}}}}}$$

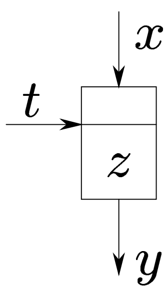
wächst.

Man kann Beispiele dafür angegeben, daß dieses Wachstum von Akzeptoren relativ zu rationalen Ausdrücken über $(REG(X^*), \cup, \cdot, *, \_)$ unvermeidlich ist.

Ein Schaltwerk ist ein Schaltkreis, der neben Gattern wie „und“, „oder“, „Addition mod 2“



auch Speicherelemente enthält. Wir verwenden hier das D-Flipflop, das durch die Tabelle definiert wird.



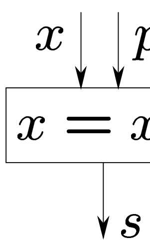
t,	x,	z	z',	y
0	x,	z	z,	z
1	x,	z	x	z

Diese Funktion wird genau dann aktiv, wenn „Taktsignal“ $t = 1$ anliegt; sonst ändern sich sein „Inhalt“ z und seine Ausgabe y nicht.

Wir können hier auf diesen Gegenstand nicht eingehen. Hierzu und zur Ergänzung des Folgenden sehe man G. Hotz: Einführung in die Informatik.

Sei nun $a = (\mathcal{W}(G), S, T, \mu, X)$ ein endlicher Akzeptor, der o.B.d.A. keine 1-Kante enthält und keine Kante, die in S endet.

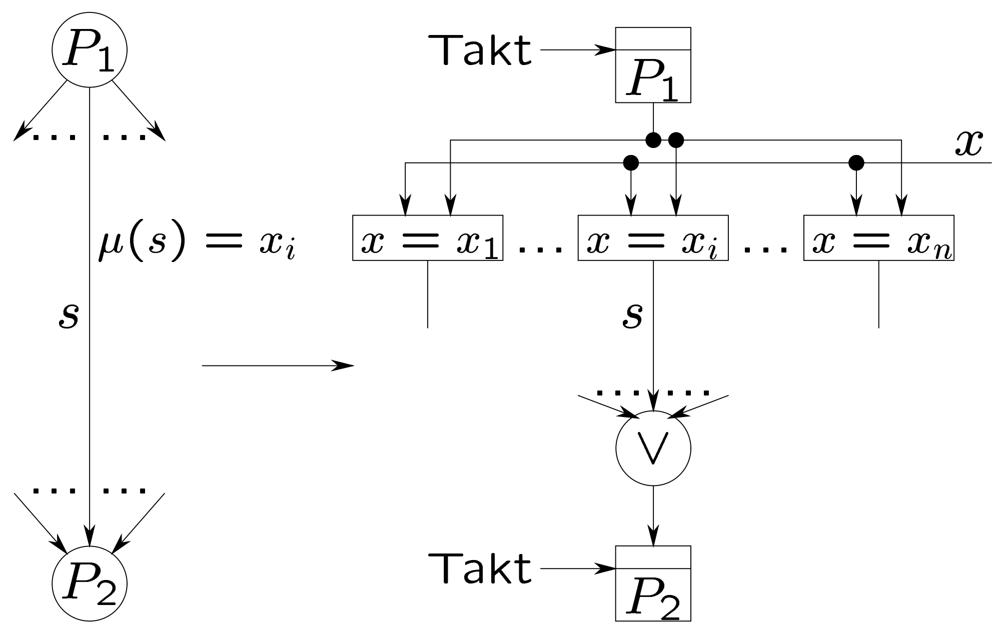
Wir ordnen jedem Knoten P von a ein Flipflop p zu und jeder Kante s von a einen Schaltkreis



$X = \{x_1, \ldots, x_n\}$, $x \in X$, $\mu(s) = x_i$.

Den Ausgang des Schaltkreises bezeichnen wir der Einfachheit wegen wieder mit s. s ist hier also eine Variable, die wie p Werte aus $\{0, 1\}$ annehmen kann. Der Schaltkreis gibt genau dann eine 1 aus, wenn das Flipflop p den Wert 1 und hat und als Eingabe x_i vorliegt.

Nun sind wir in der Lage, das Schaltwerk zu a vollständig zu definieren: Das geschieht durch das Diagramm



Zu Beginn setzen wir das zu S gehörige Flipflop auf 1. Die Ausgänge der zu T gehörigen Flipflops werden disjunktiv zu einer Funktion t zusammengefaßt. S wird nach der ersten Eingabe auf 0 gesetzt.

Ist $u \in X^*$ unsere Eingabe, dann werden genau die Flipflops aktiviert, zu deren Knoten P in a es einen Weg w mit der Beschriftung $u = \mu(w)$ gibt. Ist $u \in L_a$, dann wird mindestens eins der Flipflops aus T aktiviert und t zu 1. Wird umgekehrt $t = 1$, dann gilt $u \in L_a$.

Enthält a also N Knoten, k Kanten, und ist $\#X = n$, dann besteht das Schaltwerk aus N Flipflops und höchstens k Schaltkreisen des Typs „$x = x_i$?“ und höchstens k binären Oder-Gattern. Kodiert man die x_i durch Einheitsvektoren $e_i = (0 \cdots 010 \cdots 0)$, dann sind die Schaltkreise „$x = x_i$?“ durch ein &-Gatter realisierbar.
Setzen wir $\|a\| := \#E(a)$, dann gilt also der

Satz: Der Akzeptor a läßt sich durch ein Schaltwerk mit Kosten $\leq c \cdot \|a\|$ realisieren, wobei c eine von a unabhängige (kleine) Konstante ist.

Beispiel 1: Wir betrachten unsere Beispiele

$$L_n = \{u \cdot v \in \{a,b\}^* \mid |u| = |v| = n, u \neq v\}$$

für $n = 1$ und 2.

(Abb. 3.65, S. 298 in: G. Hotz, Einführung in die Informatik, B. G. Teubner, Stuttgart, 1990)

Eine einfache Realisierung von L_3.

(Abb. 3.66, S. 299 in: G. Hotz Einführung in die Informatik,

B. G. Teubner, Stuttgart, 1990)

Deklaration der Variablen und Prozeduren:

Beide Summanden stehen in binären n-stelligen Registern $x = (x_{n-1}, \ldots, x_0)$ und $y = (y_{n-1}, \ldots, y_0)$. Das Resultat stehe nach Abschluß der Berechnung in x. $i = (i_{k-1}, \ldots, i_0)$ ist ein binäres Register. $i := i + 1)$ bezeichnet eine Prozedur, die im Dualsystem modulo $2^k = n$ zählt. z ist eine boolesche Hilfsvariable.

Anweisungsteil

Schulmethode Addition: $Arg(x, y)$, $Res(x, z)$.

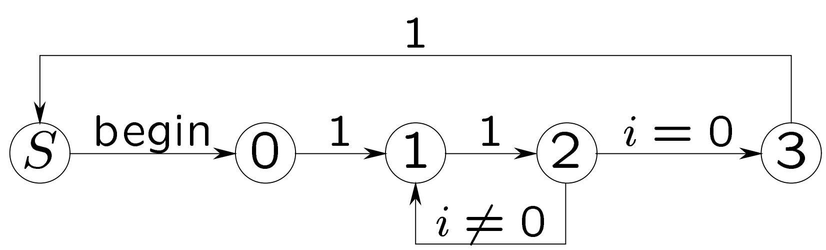
$$
\begin{array}{ll}
0: & i := 0, z := 0; \\
1: & x_i := x_i \oplus y_i \oplus z, \\
 & z := (x_i \cdot y_i) \vee ((x_i \oplus y_i) \cdot z), i := i + 1; \\
2: & \text{if } i = 0 \text{ then goto 3 else goto 1;} \\
3: & \text{end}
\end{array}
$$

Erläuterung: In Zeile 0 des Programmes werden der Zähler i und die Variable z zugleich auf Null gesetzt. i zählt die Bitposition, deren Summe und Übertrag berechnet wird. z enthält den jeweiligen Übertrag.

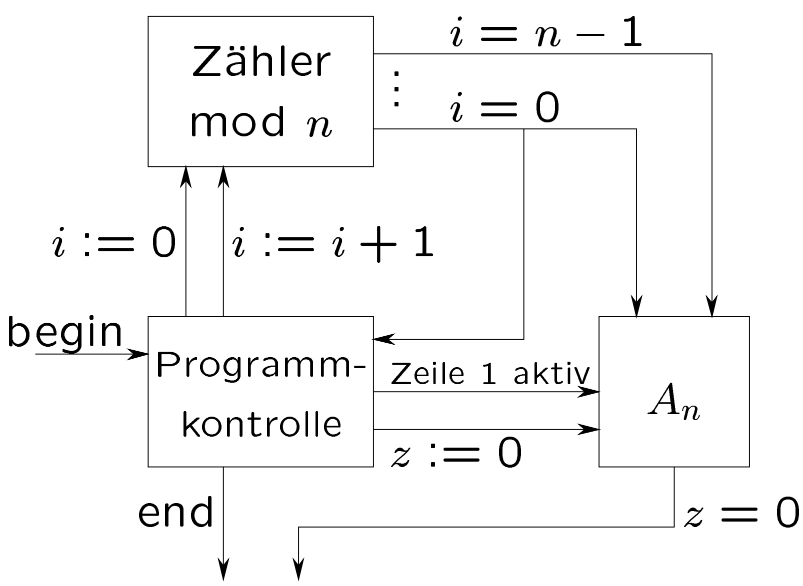
In Zeile 1 addieren wir die beiden i-ten Stellen und den Übertrag der vorhergehenden Stelle modulo 2. Das Ergebnis wird als i-tes Bit der Summe nach x_i geschrieben. Zugleich wird der Übertrag, der an der i-ten Stelle erzeugt wird, berechnet. Weiter inkrementieren wir den „Indexzähler“. In Zeile 2 fragen wir, ob alle Stellen abgearbeitet sind. Dies ist genau dann der Fall, wenn $i = 0$ ist. Dann wird in Zeile 3 das Mikroprogramm beendet. Andernfalls wird das Programm in Zeile 1 fortgesetzt.

Wir geben einen endlichen Automaten a an, der den Programmablauf steuert.

$$K := \{S, 0, 1, 2, 3\},\ T := \{3\},$$
$$X := \{\text{begin}, i = 0, i \neq 0\}.$$



In diesem Akzeptor dürfen wir die 1-Kanten **nicht** eliminieren, da mit diesen Kanten wesentliche Ausgaben verbunden sind. Über das Gesamtnetzwerk gibt das folgende Diagramm eine Übersicht:



Die Realisierung des Zählers $mod\ n$ und des Addierers A_n dürfen wir hier als bekannt voraussetzen.

Das letzte Beispiel zeigt deutlich, daß wir die Diskussion endlicher Automaten nicht vollständig auf die Diskussion endlicher Akzeptoren verlagern können. Aus diesem Grund ergänzen wir nun den endlichen Akzeptor durch eine Ausgabe.

Definition: $\underline{t} = (\mathcal{W}(G), S, T, \mu, X, \beta, Y)$ heißt **endlicher Transduktor** mit Eingabealphabet X und Ausgabealphabet Y $:\Longleftrightarrow$

$a = (\mathcal{W}(G), S, T, \mu, X)$ ist ein endlicher Akzeptor

und $\beta : \mathcal{W}(G) \longrightarrow Y^*$ ist ein Funktor.

$\underline{t}$ heißt längenerhaltend $:\Longleftrightarrow$ $|\mu(s)| = |\beta(s)|$ für alle $s \in E(G)$.

Wir sagen: $\underline{t}$ definiert die endliche Transduktion

$$\begin{aligned}\mathcal{T}(\underline{t}) := \{(u, v) \in X^* \times Y^* \mid \\ \exists\, w \in \mathcal{W}(S, T) :\ \mu(w) = u \wedge \beta(w) = v\}.\end{aligned}$$

Ist $(u, v) \in \mathcal{T}(t)$, dann reagiert also der Transduktor $\underline{t}$ auf die Eingabe u mit der Ausgabe v. I.a. wird v durch u nicht eindeutig bestimmt.

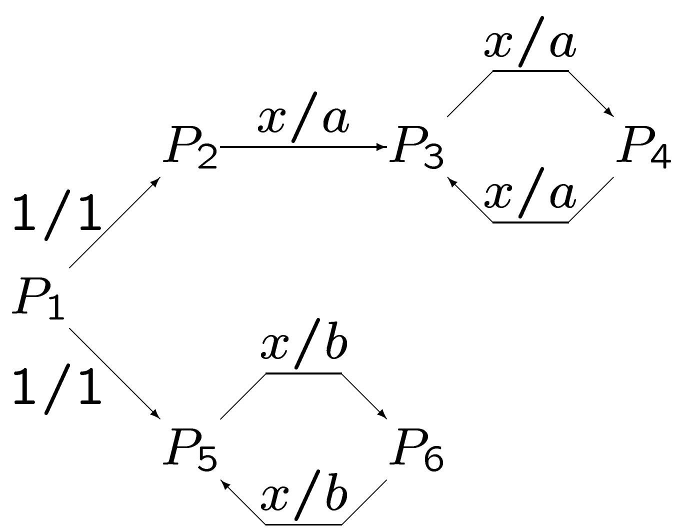
Im internationalen Sprachgebrauch heißt $\mathcal{T}(\underline{t})$ eine rationale Transduktion und $\underline{t}$ rationaler Transduktor. Da wir das Konzept der Akzeptoren, wie wir gesehen haben, in natürlicher Weise auf unendliche Akzeptoren verallgemeinern können, ist der Zusatz „endlich" nicht immer überflüssig.

Beobachtung: Zu jedem längenerhaltenden Transduktor $\underline{t}$ kann man einen Transduktor $\underline{t}'$ angeben mit $|\mu'(s)| = |\beta'(s)| \leq 1$ und $\mathcal{T}(\underline{t}) = \mathcal{T}(\underline{t}')$.

Der rationale Transduktor (3)

Beispiel:

$$\underline{t} := (\mathcal{W}(G), \{P_1\}, \{P_3, P_5\}, \mu, \{x\}, \beta, \{a, b\})$$



Notation: $\mu(s) = x,\ \beta(s) = y$ schreiben wir in dem Diagramm wie

$$\circ \xrightarrow[s]{x/y} \circ$$

$$\begin{aligned} \mathcal{T}(\underline{t}) \ = \ & \{(x,a)^n \mid n \cong 1 \ mod \ 2\} \cup \\ & \{(x,b)^n \mid n \cong 0 \ mod \ 2\} \end{aligned}$$

Definition: Für $u \in X^*$ definieren wir

$$\begin{aligned}\mathcal{T}(\underline{t})(u) \;\; := \;\; & \{v \in Y^* \mid \text{ Es gibt } w \in \mathcal{W}(G)(S,T) \\ & \text{mit } \mu(w) = u, \beta(w) = v\}\end{aligned}$$

In anderen Worten

$$\mathcal{T}(\underline{t})(u) = \{v \in Y^* \mid (u,v) \in \mathcal{T}(\underline{t})\}.$$

Wir können die Funktionen μ und β auch auf E^* fortsetzen. Dann ist $\mu^{-1}(u)$ die Menge der Urbilder von $u \in X^*$ in E^*. Es gilt für $a = (\mathcal{W}(G), S, T, \mu, X)$

$$u \in L_a \Longleftrightarrow \mu^{-1}(u) \cap \mathcal{W}(G)(S,T) \neq \emptyset$$

und also

$$\mathcal{T}(\underline{t})(u) = \beta(\mu^{-1}(u) \cap \mathcal{W}(G)(S,T)).$$

Nun ist aber $\mathcal{W}(G)(S,T) \in REG(E^*)$ (s. Übgn.). Also ist auch $\mu^{-1}(u) \cap \mathcal{W}(G)(S,T) \in REG(E^*)$. Hieraus folgt weiter

$$\mathcal{T}(\underline{t})(u) \in REG(Y^*).$$

Wir definieren für $L \subset X^*$

$$\mathcal{T}(\underline{t})(L) := \bigcup_{u \in L} \mathcal{T}(\underline{t})(u).$$

Satz: $L \subset REG(X^*) \Rightarrow \mathcal{T}(\underline{t})(L) \in REG(Y^*)$.

Beweis: Es gilt

$$\begin{aligned} & \mu^{-1}(L) \in REG(E^*) \\ & \mathcal{W}(G)(S,T) \in REG(E^*) \\ \Rightarrow \quad & \mu^{-1}(L) \cap \mathcal{W}(G)(S,T) \in REG(E^*) \\ \Rightarrow \quad & \mathcal{T}(\underline{t})(L) = \beta(\mu^{-1}(L) \cap \mathcal{W}(G)(S,T)) \\ & \in REG(Y^*). \end{aligned}$$

Lemma: $\mathcal{T}^{-1}(\underline{t}) = \{(u,v) \mid (v,u) \in \mathcal{T}(\underline{t})\}$ ist eine rationale Transduktion.

Satz: Sind

$$\mathcal{T}_1 : X^* \longrightarrow Y^*,\ \mathcal{T}_2 : Y^* \longrightarrow Z^*$$

rationale Transduktionen ohne 1-Kanten und längenerhaltend, dann ist auch $\mathcal{T}_1 \circ \mathcal{T}_2$ eine rationale Transduktion.

Beweis: Das folgende Diagramm gibt eine Übersicht über den Beweis

$$\begin{tikzcd}
 & & E_3^* \cap R_3 \arrow[dl, "\chi"'] \arrow[dr, "\rho"] & & \\
 & E_1^* \cap R_1 \arrow[dl, "\mu_1"'] \arrow[dr, "\beta_1"] & & E_2^* \cap R_2 \arrow[dl, "\mu_2"'] \arrow[dr, "\beta_2"] & \\
X^* \arrow[rr, "\tau_1"] & & Y^* \arrow[rr, "\tau_2"] & & Z^*
\end{tikzcd}$$

Hierin ist $R_i := \mathcal{W}(G_i)(S_i, T_i)$ für $i = 1, 2$. O.B.d.A. $\mu_1(E_1) \subset X$, $\mu_2(E_2) \subset Y$, $\beta_1(E_1) \subset Y$, $\beta_2(E_2) \subset Z$.

Wir setzen: $\chi(s_1, s_2) = s_1, \varrho(s_1, s_2) = S_2$ und

$E_3 := \{(s_1, s_2) \in E_1 \times E_2 \mid \beta_1(s_1) = \mu_2(s_2)\}$

und

$R_3 := \{w \in E_3^* \mid \chi(w) \in R_1$ und $\varrho(w) \in R_2\}$.

Ist also

$$w = ((s_1, s_1'), \ldots, (s_k, s_k')) \in R_3,$$

dann gilt

$$(s_1, \ldots, s_k) \in \mathcal{W}(G_1)(S_1, T_1)$$

und

$$(s_1', \ldots, s_k') \in \mathcal{W}(G_2)(S_2, T_2).$$

Setzen wir

$$\mu := \mu_1 \circ \chi, \ \ \beta := \beta_2 \circ \varrho,$$

dann ist

$$\beta(\mu^{-1}(u) \cap R_3) = \mathcal{T}_1 \circ \mathcal{T}_2.$$

Also ist $\mathcal{T}_1 \circ \mathcal{T}_2$ eine längenerhaltende rationale Transduktion ohne 1-Kanten.

Korollar: Die längenerhaltenden rationalen Transduktionen bilden eine Kategorie.

Definition: $\underline{t} := (\mathcal{W}(G), S, T, \mu, X, \beta, Y)$ heißt deterministisch, wenn $a = (\mathcal{W}(G), S, T, \mu, X)$ deterministisch ist.

Satz: Ist a ein endlicher Akzeptor, dann sind die Fragen $L_a = \emptyset$? und $L_a = X^*$? entscheidbar.

Beweis: (1) $L_{\underline{a}} = \emptyset \iff \mathcal{W}(G)(S,T) = \emptyset$. Das ist das Erreichbarkeitsproblem in einem endlichen Graphen und ist damit entscheidbar.

(2) $L_a = X^* \iff L_{\overline{a}} = \emptyset$, wenn $\overline{a}$ ein Akzeptor ist, der die Komplementmenge von L_a akzeptiert. $\overline{a}$ ist aus a effektiv konstruierbar, so daß also $L_{\overline{a}} = \emptyset$? entscheidbar.

Definition: a heißt (schwach) äquivalent zu $b :\iff$

$$L_a = L_b.$$

Satz: Für endliche Akzeptoren a und b ist $L_a = L_b$ entscheidbar.

Definition: Seien a, b endliche Automaten mit den Alphabeten X bzw. Y. Sei

$$\varphi_1 : X^* \longrightarrow Y^* \text{ ein Homomorphismus}$$

und

$$\varphi_2 : \mathcal{W}(G_a) \longrightarrow \mathcal{W}(G_b) \text{ ein Funktor.}$$

$\varphi = (\varphi_1, \varphi_2)$ heißt Akzeptorhomomorphismus $:\Longleftrightarrow$ Es gelten (A1) und (A2).

(A1) Das Diagramm

$$\begin{array}{ccc} \mathcal{W}(G_a) & \xrightarrow{\mu_a} & X^* \\ \varphi_2 \downarrow & & \downarrow \varphi_1 \\ \mathcal{W}(G_b) & \xrightarrow{\mu_b} & Y^* \end{array}$$

ist kommutativ.

(A2) $\varphi(S_a) \subset S_b$, $\varphi(T_a) \subset T_b$.

Beobachtung: $\varphi : a \longrightarrow b$ Akzeptorhomomorphismus $\Rightarrow \varphi_1(L_a) \subset L_b$.

φ heißt längenerhaltend $:\iff \varphi_2(E_a) \subset E_b$.

φ heißt nicht expandierend $:\iff \varphi_2(E_a) \subset E_b \cup \{1p \mid P \in K_b\}$.

Satz: Ist $\varphi : a \to b$ ein Akzeptorhomomorphismus und ist

$$\varphi_2(\mathcal{W}_a(S_a, T_a)) = \mathcal{W}_b(s_b, T_b),$$

dann gilt $\varphi_1(L_a) = L_b$.

Definition: Der Homomorphismus $\varphi : a \to b$ heißt **Reduktion** von a auf $b :\iff$ Es gilt $\varphi_2(\mathcal{W}_a(s_a, T_a)) = \mathcal{W}_b(S_b, T_b)$ und $\varphi_1 = id_X$.

Satz: Ist $\varphi : a \to b$ eine Reduktion, dann gilt $L_a = L_b$.

Satz: Sind a, b endliche Akzeptoren und ist $L_a = L_b$, dann gibt es eine Kette von Reduktionen, die a in b überführen.

Erläuterung: Sei $a = a_1, a_2, \ldots, a_n = b$ eine Folge von Automaten und es gebe zu a_i, a_{i+1} stets eine Reduktion $\varphi^i : a_i \longrightarrow a_{i+1}$ oder $\varphi^i : a_{i+1} \longrightarrow a_i$. Da die Hintereinanderausführung von zwei Reduktionen wieder eine Reduktion ist und da die Identität ebenfalls eine Reduktion ist, können wir uns auf die Situation

$$\begin{tikzcd}
 & a_2 \arrow[dl] \arrow[dr] & & \arrow[dl] \cdots & \arrow[dl] a_{n-1} \arrow[dr] & \\
a_1 & & a_3 & & & a_n
\end{tikzcd}$$

beschränken.

Der Beweis verwendet das folgende

Lemma: Sind a, b, c vollständige deterministische Akzeptoren, und sind $\varphi_a : c \longrightarrow a$ und $\varphi_b : c \longrightarrow b$ Reduktionen mit $\varphi_{a,2}^{-1}(S_a) = S_c$, $\varphi_{b,2}^{-1}(S_b) = S_c$, $\varphi_{a,2}^{-1}(T_a) = T_c$, $\varphi_{b,2}^{-1}(T_b) = T_c$, dann gibt es einen vollständigen deterministischen Akzeptor d und Reduktionen $\gamma_a : a \longrightarrow d$, $\gamma_b : b \longrightarrow d$, so daß das Diagramm

$$\begin{tikzcd}
 & c \arrow[dl, "\varphi_a"'] \arrow[dr, "\varphi_b"] & \\
a \arrow[dr, "\gamma_a"'] & & b \arrow[dl, "\gamma_b"] \\
 & d &
\end{tikzcd}$$

kommutativ ist.

Beweisidee: Man definiere auf K_c und E_c Äquivalenzrelationen, indem man solche Elemente zusammenfaßt, die unter φ_a oder φ_b auf das gleiche Element abgebildet werden. Hat man also

$$\begin{tikzcd} s_1 \arrow[dr, "\varphi_a"'] & & s_2 \arrow[dl, "\varphi_a"] \arrow[dr, "\varphi_b"'] & & s_3 \quad \dots, \arrow[dl, "\varphi_b"] \\ & s & & s & \end{tikzcd}$$

dann bildet man die Klasse $[s_1] = \{s_1, s_2, s_3, \dots,\}$.

Diese Klassenbildung führt zu einem Akzeptor d mit der gewünschten Eigenschaft.

Bemerkung: Die Voraussetzung „vollständig“ ist wesentlich dafür, daß es die Reduktionen γ_a und γ_b gibt.

Folgerung: Jede Kette von Reduktionen vollständiger deterministischer Automaten läßt sich auf ein Diagramm reduzieren der Form

$$\begin{tikzcd} a \arrow[dr] & & b \arrow[dl] \\ & c & \end{tikzcd} .$$

Definition: Sei $a = (\mathcal{W}(G), S, T, \mu, X)$ ein endlicher Akzeptor und $P_1, P_2 \in K(a)$. Weiter sei $a_i := (\mathcal{W}(G), P_i, T, \mu, X)$ für $i = 1, 2$. P_1 heißt **äquivalent** zu P_2 bezüglich $a :\Longleftrightarrow L_{a_1} = L_{a_2}$. a heißt **reduziert** $:\Longleftrightarrow$ Für alle Punktepaare $\{P_1, P_2\}$ aus $K(a)$ gilt P_1 nicht äquivalent zu P_2 bezüglich a.

Reduktion von a

Sei $P_1 \neq P_2$, aber P_1 äquivalent zu P_2 bezüglich a. Wir konstruieren den Automaten a' aus a, indem wir die Knoten P_1 und P_2 identifizieren. Den so entstandenen Knoten bezeichnen wir mit $[P_1]$. Die Kanten s, die in P_i enden bzw. beginnen enden bzw. beginnen nun in $[P_1]$. Der neue Graph sei G'. Die Beschriftung der Kanten wird nicht verändert. Bei dieser Identifikation mögen T in T' und S in S' übergehen. Wir definieren : $a' := (\mathcal{W}(G'), S', T', \mu, X)$.

Lemma: $L_a = L_{a'}$.

Beweis: (1) Sei nun $\varphi : \mathcal{W}(G) \longrightarrow \mathcal{W}(G')$ der Funktor, der durch die identische Abbildung auf E erzeugt wird.

(2) Ist $w \in \mathcal{W}(G)$, dann folgt $w \in \mathcal{W}(G')$. Da sich die Beschriftung beim Übergang von a zu a' nicht geändert hat, gilt

$$\mu(w) = \mu(\varphi(w)).$$

Aus $u \in L_a$ folgt also $u \in L_{a'}$, d.h. $L_a \subset L_{a'}$.

(3) I.a. ist $\mathcal{W}(G) \subsetneq \mathcal{W}(G')$: Es gilt ja nach Konstruktion

$$\begin{array}{ccccc} \xrightarrow{w_1} & P_1 & & & \\ & & & \Longrightarrow & \xrightarrow{w_1} [P_1] \xrightarrow{w_2} \\ & P_2 & \xrightarrow{w_2} & & \\ & & & & \\ & a & & & a' \end{array}$$

Ist $w' \in \mathcal{W}(G')(S', T')$, dann zerlegen wir $w' = w'_1 \cdot w'_2 \cdot \ldots \cdot w'_m$, worin kein w'_i den Knoten $[P_1]$ im Inneren enthält.

Wir betrachten die Situation

$$\begin{array}{lll} \xrightarrow{w'_{m-1}} & P_1 & \\ & P_2 & \xrightarrow{w'_m} T \end{array}$$

Da P_1 äquivalent zu P_2 bezüglich a ist, gibt es $\tilde{w}_m \in \mathcal{W}(G)(P_1, T)$:

$$P_1 \xrightarrow{\tilde{w}_m} T,\ \mu(\tilde{w}_m) = \mu(w'_m).$$

Nun ersetzen wir $w' = w'_1 \cdot \ldots \cdot w'_m$ durch den Weg

$$\tilde{w} := w'_1 \cdot \ldots \cdot w'_{m-1} \cdot \tilde{w}_m.$$

Also „springt“ $\tilde{w}$ einmal weniger als w' von P_1 nach P_2 oder umgekehrt, und es gilt $\mu(w') = \mu(\tilde{w})$. Durch die Iteration dieser Konstruktion erhalten wir schließlich einen Weg $w \in \mathcal{W}(G)(S, T)$ mit $\mu(w) = \mu(w')$. Also aus $u \in L_{a'}$ folgt $u \in L_a$, d.h. $L_{a'} \subset L_a$.

Aus (1) und (2) ergibt sich das Lemma.

Sei nun $a = (\mathcal{W}(G), S, T, \mu, X)$ ein endlicher Akzeptor und sei $P \in K(a)$. Wir definieren

$$[P]_a := \{P' \mid PqP' bzgl. a\}.$$

G' gehe aus G durch die Identifikation der jeweils äquivalenten Punkte hervor. Dabei gehe S in S' und T in T' über. Die Kanten $s \in E$ werden bei dieser Identifikation in natürlicher Weise verlagert, ohne ihre Identität einzubüßen. Der so erhaltene Automat sei a'.

Definition: a' heißt der zu a gehörige **zustandsreduzierte** Akzeptor.

Lemma: Ist a' der zu a gehörige zustandsreduzierte Akzeptor, dann gilt $L_a = L_{a'}$. Die natürliche Abbildung $P \mapsto [P]$, $s \mapsto s$ definiert einen Akzeptorhomomorphismus φ.

Beweis: a' läßt sich offfensichtlich durch die Iteration des Reduktionsschrittes aus dem vorigen Lemma konstruieren. Also gilt $L_{a'} = L_a$.

Die natürliche Abbildung φ führt Wege aus $\mathcal{W}(G)(S,T)$ in Wege aus $\mathcal{W}(G')(S',T')$ über. Die Beschriftung von a' überträgt sich mit $s \mapsto s$ identisch. Also ist φ ein Akzeptorhomomorphismus.

Lemma: Ist a ein endlicher Akzeptor und geht a' aus a durch Zerlegung der Kante s mit $\mu(s) = x_1 \cdot \ldots \cdot x_k$ in einen Weg $s_1 \cdot \ldots \cdot s_k$ und $\mu'(s_i) := x_i$ hervor, dann definiert die Abbildung $s \mapsto s_1 \cdot \ldots \cdot s_k$ und $s' \mapsto s'$ für $s' \in E$, $s' \neq s$ einen Akzeptorhomomorphismus von a auf a'.

Beweis: klar.

Beobachtung: Diese Reduktion erhöht die Anzahl der Kanten.

Definition: Der Akzeptor a heißt reduziert $:\Longleftrightarrow$

(1) a ist knotenreduziert

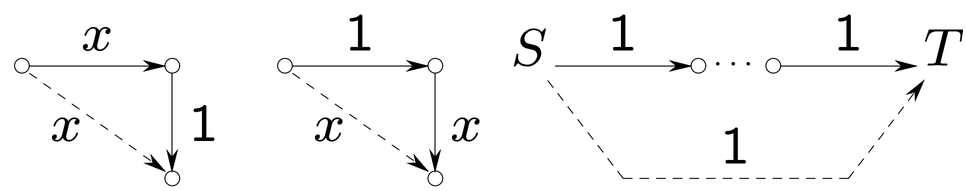
(2) $|\mu(s)| \leq 1$ für $s \in E$

(3) $s, s' \in E \ Q(s) = Q(s'),\ Z(s) = Z(s'),$
$\mu(s) = \mu(s') \Rightarrow s = s'.$

Satz: Zu jedem Akzeptor a gibt es einen reduzierten Akzeptor a' und eine Reduktion $\varphi : a \to a'$.

Beweis: Folgt aus den beiden vorigen Lemmata und der Beobachtung, daß die Identifikation paralleler, gleichbeschriftetet Kanten eine Reduktion definiert.

Die Elimination von 1-Kanten beruhte auf der Einführung von Brücken, wie es die folgenden Diagramme zeigen: Die Brücken sind hervorgehoben.



Nachdem alle erforderlichen Brücken eingezogen waren, konnten wir die 1-Kanten (außer einer von s nach T) löschen. Sei a' der Automat, der aus a durch diese Konstruktion gewonnen wurde.

Nun definieren wir eine Reduktion $\varphi : a' \to a$, indem wir jede neu gewonnene Kante auf den Weg abbilden, den sie überbrückt. Diese Abbildung ist offensichtlich ein Homomorphismus und wie wir bereits gezeigt haben, läßt dieser Homomorphismus die Sprache invariant. Also gilt das

Lemma: Geht a durch die 1-Kantenelimination durch Überbrückungen in a' über, dann gibt es eine Reduktion $\varphi : a' \to a$.

Wir betrachten nun noch das „vollständig und deterministisch machen“.

Für den in dem vorigen Lemma konstruierte Automat a' gilt $|\mu'(s)| = 1$ für alle bis auf höchstens eine Kante von S nach T mit Beschriftung $= 1$. Das Vollständigmachen bestand in der Einführung eines neuen Knotens P, zu dem alle erforderlichen neuen Kanten hingeführt wurden. Zu jedem $x \in X$ haben wir weiter Kanten $P \xrightarrow{s_x} P$ mit $\mu(s_x = x)$ eingeführt. Ist a'' der so zu a' konstruierte vollständige Akzeptor, dann definiert die Inklusion von a' in a'' eine Reduktion.

Wir fassen die erhaltenen Resultate in einem Diagramm zusammen.

$$a_1 \xrightarrow{\varphi_1} a_2 \xleftarrow{\varphi_2} a_3 \xrightarrow{\varphi_3} a_4 \xrightarrow{\varphi_4} a_5.$$

Hierin ist a_1 der beliebig vorgegebene Akzeptor.

φ_1 bezeichnet das „Kantenbrechen“.
φ_2 bezeichnet die „Entfernung der 1-Kanten“.
φ_3 führt a_3 in den „zugehörigen reduzierten“ Akzeptor über.
φ_4 liefert die Einbettung in den vervollständigten Akzeptor.

Wir können $\varphi_3 \circ \varphi_4$ durch eine Reduktion φ_3' überbrücken. Also gilt das

Lemma: Jeder endliche Akzeptor a_1 läßt sich durch eine Kette von drei Reduktionen in einen Akzeptor a_4 überführen.

$$a_1 \xrightarrow{\varphi_1} a_2 \xleftarrow{\varphi_2} a_3 \xrightarrow{\varphi_3'} a_4' = a_5.$$

Korollar: Ist a_1 ein 1-kantenfreier Akzeptor,dann läßt sich a_1 durch eine Reduktion in einen reduzierten vollständigen Akzeptor überführen.

Wir gehen nun von dem vollständigen Akzeptor a aus und betrachten die Konstruktion des zugehörigen vollständigen deterministischen Akzeptor a'.Wie sich aus unserem Beweis für $L_a = L_{a'}$ ergibt, können wir jedem Weg $w \in \mathcal{W}(G)$ aufgrund seiner Beschriftung eindeutig einen Weg $w' \in \mathcal{W}(G')$ mit der gleichen Beschriftung zuordnen. Wir haben darüber hinaus gezeigt, daß es zu jedem $w' \in \mathcal{W}(G')(S', T')$ einen Weg $w \in \mathcal{W}(G)(S, T)$ gibt, der bei dieser Konstruktion auf w' abgebildet wird.

Diese Konstruktion ist aber i.a. **kein** Akzeptorhomomorphismus. Der Weg

$$w := P_1 \xrightarrow{x_1, s_1} P_2 \xrightarrow{x_2, s_2} P_3 \longrightarrow \cdots \xrightarrow{x_{k-1}, s_{k-1}} P_k$$

kann z.B. übergehen in

$$\overline{P}_1 := \{P_1\} \xrightarrow{x_1} \overline{P}_2 := \{\ldots, P_2, \ldots\} \xrightarrow{x_2} \cdots \xrightarrow{x_{k-1}} \overline{P}_k := \{\ldots, P_k, \ldots\},$$

aber auch in einen Weg

$$\tilde{w} := \tilde{P}_1 := \{\ldots, P_1, \ldots\} \xrightarrow{x_1} \cdots \xrightarrow{x_{k-1}} \tilde{P}_k := \{\ldots, P_k, \ldots\},$$

wenn w z.B. ein Weg w' vorgeschaltet wird, der von einem Knoten $\{P\}$ zu dem Knoten $\tilde{P}_1$ von a' führt.

Wir erhalten aber, wie schon gesagt, eine surjektive Abbildung von $\mathcal{W}(G)(S,T)$ auf $\mathcal{W}(G')(S',T')$. Man sagt: Die Wege $\mathcal{W}(G)(S,T)$ **überlagern** $\mathcal{W}(G')(S',T')$.

Wir betrachten zum Abschluß die Frage, wie vollständige deterministische Akzeptoren a, b mit $L_a = L_b$ zusammenhängen.

Hierzu definieren wir zunächst für $u, v \in X^*$

$$u \equiv_r v(L) :\Longleftrightarrow \forall_w (uw \in L \iff vw \in L).$$

Man sieht: Für $w \in X^*$ gilt

$$u \equiv_r v(L) \Rightarrow uw \equiv_r vw(L).$$

Wir bezeichnen mit $[u]_r$ diese Rechtskongruenzklassen. Die Menge dieser Klassen sei X^*/L. Nun ist

$$\#(X^*/L) < \#Synt(L)$$
$$\Rightarrow \ (L \in REG(X^*) \Rightarrow X^*/L \text{ ist endlich}).$$

Wir ordnen L vermittels X^*/L einen endlichen Akzeptor a mit $L_a = L$ zu, indem wir definieren

$$[u]_r \xrightarrow{x} [u']_r :\Longleftrightarrow u' = u \cdot x,$$
$$s = [1]_r, \ T = \{[u]_r \mid u \in L\}.$$

a ist vollständig und deterministisch. Wir können die Wege W in G_a als durch Angabe des Anfangspunktes und ihre Beschriftung $\mu(w)$ eindeutig bestimmen. Offensichtlich ist $L_a = L$. Nach Konstruktion ist a reduziert.

Sei nun b ein beliebiger vollständiger deterministischer Akzeptor mit $L_b = L_a$. Ist P ein Knoten von b und $b_P = (\mathcal{W}(G_b), P, T, \mu, X)$ und $L_P := L_{b_P}$, dann definieren wir

$$\Psi(P) := [u]_r \quad \text{für} \quad S \xrightarrow{u} P.$$

Diese Abbildung ist wohldefiniert...

Ist $s \in E_b$ mit $\mu(s) = x$, $Q(s) = P$, dann sei

$$\varphi_2(s) = s' \;:\Longleftrightarrow\; Q(s') = \Psi(P) \text{ und } \mu(s') = x$$

φ_2 ist wohldefiniert aufgrund der Determiniertheit von a und b.

Setze auf $\mathcal{W}(G_b)$ fort. $\varphi_1 := id_X$.

Also ist $\varphi = (\varphi_1, \varphi_2)$ eine Reduktion von b auf a.

Satz: Jeder vollständige deterministische Akzeptor a läßt sich auf den durch die Rechtskongruenz X^*/L_a definierten Akzeptor reduzieren.

Korollar: Vollständige deterministische reduzierte Akzeptoren sind isomorph.

Zusammenfassung: Sind a und b endliche Akzeptoren und ist $L_a = L_b$, dann lassen sich a und b durch Reduktionen und Überlagerungen verbinden, wie das durch das folgende Diagramm beschrieben wird

$$a = a_1 \xrightarrow[\varphi_1]{} a_2 \xleftarrow[\varphi_2]{} a_3 \overset{}{\underset{\psi}{\dashrightarrow}} \tilde{a} \cong \tilde{b} \overset{}{\underset{\psi'}{\dashleftarrow}} b_3 \xrightarrow[\varphi_2']{} b_2 \xleftarrow[\varphi_1']{} b_1 = b$$

Hierin sind φ_1, φ_2, φ_1', φ_2' Reduktionen (Kantenbrechen, 1-Kantenelimination) und Ψ, Ψ' sind Überlagerungen. Dieses Diagramm weicht von unserem Diagramm auf Seite 107 insofern ab, als wir auf die dort gegebene „vollständige“ Reduktion verzichten und das Vervollständigen in die Überlagerung mit hereingezogen haben.

Realisierung von Akzeptoren

Auf die Frage nach der Realisierung endlicher Akzeptoren durch Schaltwerke oder sequentielle Schaltkreise sind wir nur grundsätzlich eingegangen. Die angegebene Konstruktion liefert eine Lösung für Akzeptoren a mit $|\mu(s)| = 1$, deren Kosten linear mit der Anzahl der Kanten von a wächst. Die Zustandsmenge dieser Realisierung hat $2^{|a|}$ Elemente. Macht man a zunächst deterministisch um dann Standardmethoden zur Realisierung boolescher Funktionen auf der Basis einer Kodierung der Zustände durch eine Numerierung mit Dualzahlen minimaler Länge, dann mögen wesentlich aufwendigere Realisierungen entstehen. Eine befriedigende systematische Theorie zur Lösung dieses Problems gibt es noch nicht.

Schlußbemerkung zu dem Kapitel „Endliche Akzeptoren“

Eine natürliche Weiterführung der Theorie der endlichen Akzeptoren besteht in der Ergänzung des Akzeptors durch ein „Eingabeband“, auf dem sich der Lesekopf in beiden Richtungen bewegen darf. Das hier behandelte Modell läßt das Lesen nur in einer Richtung zu. Man kann zeigen,daß diese Erweiterung nicht über reguläre Mengen hinausführt. Allerdings kann man zeigen, daß es zu gewissen regulären Mengen L 2-Wege-Akzeptoren gibt, die wesentlich kleiner sind, als jeder 1-Wege-Akzeptor von L. Die Größe des deterministischen 1-Wege-Akzeptors mag $2^{|a|^2}$ betragen, wenn $|a|$ die Größe des 2-Wege-Akzeptors ist.

Theorie der Berechenbarkeit

Erarbeitung einer Übereinkunft, was unter berechenbaren Funktionen zu verstehen ist. Hat man sich darüber geeinigt, stellen sich z. B. die folgenden Fragen:

(1) Gegeben seien zwei berechenbare Funktionen g, h. Ist $f = h$. Hierbei spielt es natürlich eine Rolle, in welchen Sprachen f und h definiert sind.

(2) Wir schwächen (1) ab zu der Frage:
   (a) Ist Definitionsbereich(f) = Definitionsbereich(g)? Oder noch schwächer:
   (b) Ist Definitionsbereich(f) $\subset M$ und $u \in M$, gilt $u \in$ Definitionsbereich(f)?
   (c) Haben folgende Fragen die gleiche Qualität?

$$\begin{array}{lcl} u & \in & \text{Definitionsbereich}(f) \text{ und} \\ u & \notin & \text{Definitionsbereich}(f) \end{array}$$

(3) Kann man jede Funktion, die man „effektiv“ definieren kann, auch berechnen?
Beispiel: Sei $\chi : \mathbb{R}^2 \longrightarrow \{0,1\}$ berechenbar.

$$M := \{(x,y) \in \mathbb{R}^2 \mid \chi(x,y) = 1\}$$

Die Frage $(x,y) \in M$ ist entscheidbar, da wir vorausgesetzt haben, daß $\chi(x,y)$ für alle $(x,y) \in \mathbb{R}^2$ berechnet werden kann.
Nun projizieren wir M auf die X-Achse. Die Projektion sei P_1. Also

$$P_1(x,y) = x.$$

Frage: Kann man die Frage

$$x \in P_1(M)$$

entscheiden? Die Funktion

$$\chi_1 : \mathbb{R} \longrightarrow \{0,1\}$$

mit $\chi_1(x) = 1 \iff x \in P_1(M)$
ist wohldefiniert. Kann man χ_1 berechnen?

(4) Kann man von manchen „effektiv“ definierten Funktionen zeigen, daß sie nicht berechenbar sind?

(5) Kann man für berechenbare Funktionen abschätzen, wie „schwierig“ es mindestens ist, sie zu berechnen?

(6) Gibt es eine Klassifikation der berechenbaren Funktionen, in der sich diese Fragen widerspiegeln?

Diese Fragen sind älter als der heute allgemein akzeptierte Begriff der Berechenbarkeit. Z. B. hat B. Riemann bereits vor 150 Jahren die möglichen Definitionsbereiche von Funktionen klassifiziert, die auf Teilmengen von $\mathbb{C}$ durch Potenzreihen und deren möglichen Fortsetzungen definiert werden können. In dem Fortsetzungsbegriff ist auch bereits ein algorithmischer Aspekt enthalten.

Das Konzept der Berechenbarkeit

Wir gehen von einem Alphabet A aus und einer Sprache $L \subset A^*$, deren Elemente wir Programme nennen. Zu jedem Programm $u \in L$ gehört eine Menge $\Delta(u)$ auf der das Programm u operiert. Z. B. für

$$\begin{aligned} u \ := \ & (\text{Deklaration: } x \text{ integer;} \\ & \text{Anweisung:} \\ & 1: \ \text{while } x > 0 \text{ do } x := x - 1; \\ & 2: \ \text{end}), \end{aligned}$$

ist $\Delta(u) = \mathbb{Z} \times \{1, 2\}$. $\Delta(u)$ wird durch den Deklarationsteil von u definiert. Der Anweisungsteil definiert auf $\Delta(u)$ eine Abbildung $\alpha(u)$, wenn das Programm „korrekt“ ist. Wir haben also

$$\alpha(u) : \Delta(u) \longrightarrow \Delta(u).$$

Die zu u gehörige Berechnung besteht in der Iteration von $\alpha(u)$. Ist also $\xi_0 \in \Delta(u)$, dann ist

$$\xi_0, \xi_1 := \alpha(u)(\xi_0), \ldots, \xi_{i+1} := \alpha(u)(\xi_i), \ldots$$

die zu (ξ_0, u) gehörige Berechnung.

Ist für ein $t \in \mathbb{N}$

$$\xi_t = \xi_{t+1}, \text{ dann ist } \xi_t = \xi_{t+l} \text{ für } l \in \mathbb{N}.$$

In diesem Fall sagen wir, daß die Berechnung **abbricht** oder **terminiert**. Ist t minimal mit dieser Eigenschaft, dann heißt t die **Länge** der Berechnung. ξ_t heißt das **Resultat** der Berechnung.

Wir schreiben in diesen Fällen

$$T(u, \xi_0) := t, \ Res(u, \xi_0) := \xi_t.$$

Die durch u definierte Funktion F_u wird definiert durch:

$$\begin{array}{lcl} \Omega(F_u) & := & \{\xi_0 \in \Delta(u) \mid \text{die zu } (u, \xi_0) \\ & & \text{gehörige Berechnung terminiert}\} \\ F_u(\xi_0) & := & \xi_t. \end{array}$$

Also u definiert die Abbildung

$$F_u : \Omega(F_u) \longrightarrow \Omega(F_u).$$

Im allgemeinen ist der Raum $\Delta(u)$ wesentlich größer als der Definitionsbereich von F_u, da $\Delta(u)$ gewöhnlich Komponenten enthält,die nur für die Organisation der Berechnung wichtig sind. Hierauf nehmen wir Rücksicht, indem wir zwei **einfache** Funktionen **in** = „Eingabe“ und **out** = „Ausgabe“ hinzufügen, so daß wir L-berechenbar wie folgt definieren

$$
\begin{array}{cccccc}
 & M & \xrightarrow{f} & M' & \\
\text{in} & \downarrow & & \uparrow & \text{out} \\
 & \Delta(u) & & \Delta(u) & \\
 & \cup & & \cup & \\
 & \Omega(u) & \xrightarrow{F_u} & \Omega(u) &
\end{array}
$$

kommutativ ist.

Dieser Berechenbarkeitsbegriff enthält die folgende Schwierigkeit: Starten wir die Berechnung mit ξ_0, dann wissen wir nicht, ob die Berechnung abbricht. Bricht die Berechnung ab, dann haben wir Glück gehabt, bricht sie nicht ab, dann warten wir ∞-lange vergeblich. Es stellt sich daher die Frage, ob man das nicht vorher wissen kann. Wir fragen ja gar nicht nach dem Resultat, sondern nur danach, ob wir eines erhalten werden.

Haben wir uns für die Sprache L entschieden, dann lautet diese Frage: Gibt es ein Programm $u \in L$, so daß dieses Programm für alle Programme $v \in L$ und $\xi_0 \in \Delta(v)$ entscheiden kann, ob die zu (v, ξ_0) gehörige Berechnung abbricht. Dazu benötigen wir offensichtlich eine Kodierung von (v, ξ_0) durch ein Element aus $\Delta(u)$. Wir wollen uns über die Konstruktion von γ nicht den Kopf zerbrechen, bevor wir wissen, ob es eine solche Kodierung überhaupt geben kann.

Wir nehmen an, daß $L \subset \{0,1\}^*$ ist. Wir können das ohne Einschränkung der Allgemeinheit tun, da sich X^* für jedes X, das höchstens abzählbar-unendlich groß ist, isomorph auf ein Untermonoid von $\{0,1\}^*$ abbilden läßt. Ist $X = \{x_1, \ldots, x_n\}$, dann definiert

$$h(X_i) := 1\,\underbrace{0\ldots0}_{i}$$

einen solchen Isomorphismus.

Alle Daten werden in unseren Computern binär abgelegt, so daß wir auch annehmen dürfen, daß

$$\Delta(v) = X^* \quad \text{für alle } v \in L$$

und

$$\alpha(v) : X^* \longrightarrow X^*$$

die durch v definierte Abbildung ist, deren Iteration die Berechnung beschreibt.

Für gewisse Argumente mag die Berechnung abbrechen, für andere nicht. Wir fragen nun: Gibt es eine Kodierung

$$\gamma : L \times X^* \longrightarrow X^*$$

und ein Programm u mit den Eigenschaften (1) und (2)?

(1) $F_u(\gamma(v,w)) = 1 \iff \gamma(v,w) \notin \Omega(v)$

(2) $\gamma(v,w) \in \Omega(v) \iff \gamma(v,w) \notin \Omega(u)$.

Das Programm u schreibt also eine 1 aus, wenn v auf w angesetzt nicht terminiert. In allen anderen Fällen terminiert u nicht. Wir werden zeigen

Lemma: $u \notin L$.

Beweis: Annahme $u \in L$.

$$\Rightarrow \quad F_u(\gamma(u,u)) = 1 \Leftrightarrow \gamma(u,u) \notin \Omega_u.$$

Widerspruch!

Beantwortet die Konstruktion von u die gestellte Frage? Wenn wir ein Programm für unsere Maschine haben, das entscheidet, ob $v \in L$ auf w angesetzt hält oder nicht, dann muß sie in jedem Falle anhalten, da wir die gewünschte Information im Falle des Nichtanhaltens nicht bekommen. Die Maschine muß also auch anhalten, wenn die Eingabe keine Kodierung $\gamma(v, w)$ ist. Verlangen wir nun von der Programmiersprache L, daß sie in dem unten definierten Sinne vollständig ist, dann können wir u so abwandeln, daß u die Voraussetzung des Lemmas erfüllt.

Definition: L heißt **schwach vollständig** $:\iff$ Zu jedem Programm $v \in L$ und $c \in X^*$ gibt es ein Programm $v' \in L$ mit folgenden Eigenschaften:

- $F_v(w) = c \quad \Rightarrow \quad w \notin \Omega(v'),$

- $F_v(w) \neq c \quad \Rightarrow \quad F_{v'}(w) = F_v(w).$

Die Annahme, daß L schwach vollständig ist, ist eine plausible Annahme, da sie nur voraussetzt, daß man in dem Fall, daß ein Programm mit bestimmtem Wert terminiert, das Programm durch eine Prozedur erweitern kann, die im Falle dieser Terminierung z. B. die alternierende Folge $0, 1, 0, 1, \ldots$ als Ausgabe erzeugt.

Haben wir also ein Programm $\tilde{u}$, das auf $\gamma(v, w)$ angesetzt eine 1 ausgibt, falls v auf w angesetzt nicht anhält, und in allen anderen Fällen 0, dann gibt es auch ein Programm $u \in L$, das im Falle der Ausgabe 0 nicht anhält und sonst den gleichen Wert wie $\tilde{u}$ liefert. Also gilt

Satz: Schwach vollständige Programmiersprachen L enthalten kein Programm, das das Halteproblem für Programme v aus L entscheidet.

Es kann zu Sprachen L_0 Programme u_0 geben, die das Halteproblem für Programme aus L_0 entscheiden. Erweitert man diese Sprachen, indem man u_0 zu L_0 hinzunimmt, so erhält man eine Sprache L_1, die außer u_0 noch weitere zusätzliche Programme enthält, wenn L_0 z. B. ein Prozedurkonzept einschließt. Indem wir nun den Satz auf L_1 anwenden, kommen wir in dem Fall, daß ein Programm u_1 existiert, das das Halteproblem für L_1 löst, zu einer Sprache L_2 und schließlich zu einer ganzen Hierarchie

$$L_0 \subsetneq L_1 \subsetneq L_2 \subsetneq \dots$$

von Sprachen, die auch ∞-groß werden kann.

Geht man davon aus, daß man eine Sprache L_u vorliegen hat, die universell in dem Sinne ist, daß sich jede berechenbare Funktion in ihr programmieren läßt, dann ist also die Funktion, die das Halteproblem für L_u löst, keine berechenbare Funktion.

Wir erläutern die Idee der Konstruktion einer Funktion, die nicht in L liegt, etwas näher.

Sei M eine Menge, $O \subset 2^M$ und

$$F \subset \{f : M_1 \longrightarrow \{0,1\} \mid M_1 \in O\}$$

Es sei $\tilde{M} \subset M$ und

$$\gamma : \tilde{M} \longrightarrow F$$

eine Abbildung. Ist $m \in \tilde{M}$ und $\gamma(m) = f$, dann sehen wir m als eine Bezeichnung für f an.

Nun betrachten wir die Abbildung $u : \tilde{M} \longrightarrow \{0,1\}$, die durch

$$u(m) := \begin{cases} 1 & \text{für } m \notin Q(\gamma(m)) \text{ oder } \gamma(m)(m) \neq 1 \\ 0 & \text{für } m \in Q(\gamma(m)) \text{ und } \gamma(m)(m) = 1 \end{cases}$$

definiert wird.

Lemma: $u \notin F$ oder γ ist nicht surjektiv.

Beweis: Annahme: $u \in F$ und γ ist surjektiv. Ist $u \in F$, dann gibt es ein $m_0 \in \tilde{M}$ mit $\gamma(m_0) = u$, und also $Q(\gamma(m_0)) = \tilde{M}$.

- Ist $u(m_0) = 1$, dann ist $m_0 \notin \tilde{M} = Q(\gamma(m_0))$,oder es ist $u(m_0) \neq 1$ im Widerspruch zur Annahme.

- Ist $u(m_0) = 0$, dann ist $m_0 \in \tilde{M}$ und $\gamma(m_0)(m_0) = 1$ im Widerspruch zur Annahme.

Also ist die Annahme

$$u \in F \text{ und } \gamma \text{ surjektiv}$$

nicht erfüllt.

Wir betrachten eine zweite Version dieses Lemmas.

Lemma: Sei

$$M_F := \{m \in M \mid m \notin Q(\gamma(m))\}$$

und

$$u_F : M_F \longrightarrow \{1\}.$$

Dann gilt γ ist nicht surjektiv oder $u_F \notin F$.

Beweis: Wir nehmen an: γ ist surjektiv. Ist $u_F \in F$, dann finden wir ein $m_0 \in \tilde{M}$ mit $\gamma(m_0) = u_F$.

Falldiskussion:

(a) $m_0 \in M_F \Rightarrow m_0 \notin (\gamma(m_0)) = M_F$. Widerspruch.

(b) $m_0 \notin M_F \Rightarrow m_0 \in Q(\gamma(m_0)) = M_F$. Widerspruch.

Also ist mindestens eine unserer beiden Annahmen falsch.

Die beiden Konstruktionen enthalten als wesentliches Element

- Die Bezeichnung γ der Abbildungen aus F durch Elemente aus M.

- Die Anwendung von $\gamma(m)$ auf m.

Beide Lemmata geben also Verfahren an, Abbildungen zu konstruieren, die nicht in F liegen. Dazu muß es nur möglich sein, eine Teilmenge $\tilde{M} \subset M$ surjektiv auf F abzubilden.

Die Konstruktion der „Haltefunktion“ u zur Sprache L ist eine Anwendung des ersteren der beiden Lemmata. Zur Erläuterung geben wir noch zwei Anwendungen.

Anwendung 1: Die Menge der reellen Zahlen ist nicht aufzählbar.

Beweis: Wähle $M = \mathbb{N}_0$ und

$\gamma : \mathbb{N}_0 \longrightarrow F := \{a : \mathbb{N}_0 \to \{0,1\} \mid a \text{ Abbildung}\}.$

Wir können F vermöge der Definition

$$\overline{a} := \sum_{i \in \mathbb{N}_0} \frac{a(i)}{2^{i+1}}, \quad a \in F$$

als die Menge der reellen Zahlen aus $[0,1]$ ansehen.

Wir setzen nun wie in dem ersten Lemma

$$u(i) \neq \gamma(i)(i) \quad \text{für } i \in \mathbb{N}_0.$$

Offensichtlich ist $u \in F$. Also ist, wie das Lemma besagt, γ nicht surjektiv. Also ist $[0,1]$ nicht aufzählbar.

Das Lemma ergibt hier also das **Cantor'sche Diagonalisierungsverfahren**.

Anwendung 2: Die Antinomie des Barbiers

Nach griechischer Überlieferung ist ein Barbier ein Mann, der die Athener barbiert, die sich nicht selbst barbieren.

Sei M die Menge der vollständigen Namen der Athener. $\gamma(m)$ ordnet m seine Barbierfunktion zu.

Es sei

$$\gamma(m)(m') = 1 \iff m \text{ barbiert den } m!$$

$\gamma(m)$ ist also für alle Athener definiert und ist surjektiv.

Wir definieren nun den Barbier u :

$$u(m) = 1 :\iff \gamma(m)(m) = 0$$

Unser zweites Lemma ergibt nun $u \notin F$, da γ surjektiv ist.

Wir sehen, daß die Idee der hier verwendeten Konstruktion etwa 2000 Jahre alt ist.

Wir werden nun in den folgenden Abschnitten verschiedene Konzepte der Berechenbarkeit entwickeln und sie miteinander vergleichen. Dabei werden wir auch wiederholt auf die hier geschilderten Grundrezepte, das Konzept der Iteration einer Abbildung und der Diagonalisierung zur Unterscheidung von Klassen von Abbildungen zurückbekommen. Wissen wir nämlich, daß $L \subset L'$ gilt, und gilt für die „Haltefunktion“ u zu L, $u \in L'$, dann ist $L \subsetneq L'$.

μ-rekursive Funktionen

1. Primitiv-rekursive Funktionen

Wir definieren induktiv eine Menge $\mathbb{P}$ von Funktionen mit den Definitionsbereichen und Zielbereichen

$$\mathbb{N}^0, \mathbb{N}^1, \mathbb{N}^2, \ldots, \mathbb{N}^i, \ldots.$$

Wir definieren $\mathbb{P}$, indem wir zunächst die „elementaren“ oder „Basisfunktionen“ von $\mathbb{P}$ definieren und anschließend Operationen auf der Menge der Funktionen. $\mathbb{P}$ ergibt sich dann als der kleinste Abschluß hinsichtlich dieser Operationen, der die Basisfunktion enthält.

Die Menge der Basisfunktionen E besteht aus den folgenden Elementen:

$$\begin{array}{lll}
1 & : & \mathbb{N} \to \mathbb{N},\ 1(x) = x \text{ (Identität)} \\
c_0 & : & \mathbb{N}^0 \to \mathbb{N},\ c_0(\Box) = 0 \text{ (nullstellige Konstante)} \\
N & : & \mathbb{N} \to \mathbb{N},\ N(x) = x + 1 \text{ (Nachfolgerfunktion)} \\
p_0 & : & \mathbb{N} \to \mathbb{N}^0,\ p_0(x) = \Box \text{ (Projektion)} \\
d_1 & : & \mathbb{N} \to \mathbb{N}^2,\ d_1(x) = (x, x) \text{ (Diagonalisierung)} \\
T & : & \mathbb{N}^2 \to \mathbb{N}^2,\ T(x, y) = (y, x) \text{ (Transposition)}
\end{array}$$

Operationen auf der Menge der Abbildungen

Sei $f : \mathbb{N}^i \to \mathbb{N}^j$, $g : \mathbb{N}^l \to \mathbb{N}^m$.

Wir betrachten zwei Kompositionen dieser Funktionen:

(a) $(f \times g) : \mathbb{N}^{i+l} \to \mathbb{N}^{j+m}$
$(f \times g)(x, y) := (f(x), g(y))$
für $x \in \mathbb{N}^i, y \in \mathbb{N}^l$.

(b) für $Q(f) = Z(g)$
$(f \circ g)(x) := f(g(x))$
für $x \in \mathbb{N}^i$.

Folgerung: Ist

$$f : \mathbb{N}^k \to \mathbb{N},\ g_i : \mathbb{N}^m \to \mathbb{N} \text{ für } i = 1, \ldots, k$$

dann läßt sich die durch

$$h(x_1, \ldots, x_m) := f(g_1(x_1, \ldots, x_m), \ldots, g_k(x_1, \ldots, x_m))$$

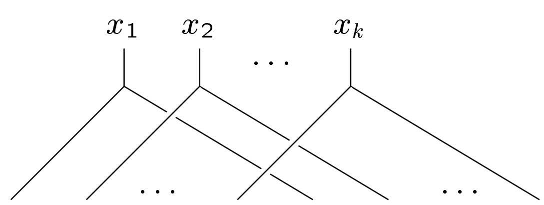
definierte Funktion mittels Basisfunktionen und den Kompositionen a) und b) erzeugen.

Wir setzen

$$1_k := 1_{k-1} \times 1,\ 1_1 = 1$$

und betrachten zunächst $d_k : \mathbb{N}^k \to \mathbb{N}^{2\cdot k}$.

$$d_k(x_1, \ldots, x_k) = (x_1, \ldots, x_k, x_1, \ldots, x_k).$$



Wir definieren d_k induktiv: d_1 ist aus E. Ist d_{k-1} definiert, dann bilden wir der Reihe nach

$$\begin{array}{lcl} \tilde{f} & = & d_{k-1} \times 1, \\ g & = & 1_{2\cdot(k-1)} \times d_1, \\ T_{k-1,1} & := & (T_{k-2,1} \times 1) \circ (1_{k-2} \times T), \end{array}$$

worin $T_{1,1} := T$ ist. Nun erhalten wir

$$d_k = (1_{k-1} \times T_{k-1,1} \times 1) \circ g \circ \tilde{f}$$

$$\begin{aligned} d_k^1 &:= d_k, \\ d_k^{i+1} &:= (d_k^i \times 1_k) \circ d_k, \\ h &:= f \circ (g_1 \times \ldots \times g_k) \circ d_m^{k-1}. \end{aligned}$$

Nun definieren wir weiter das **Schema der primitiven Rekursion.**

Seien

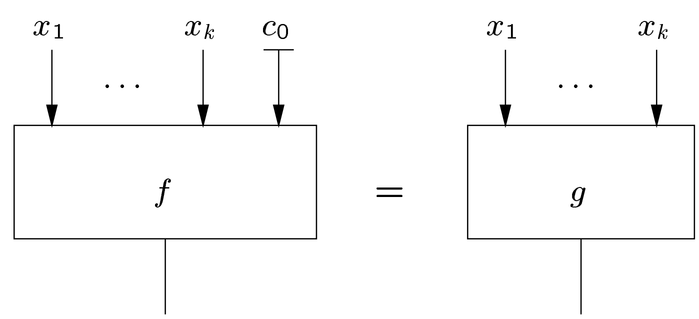
$$g : \mathbb{N}^k \to \mathbb{N},\ \ h : \mathbb{N}^{k+2} \to \mathbb{N}$$

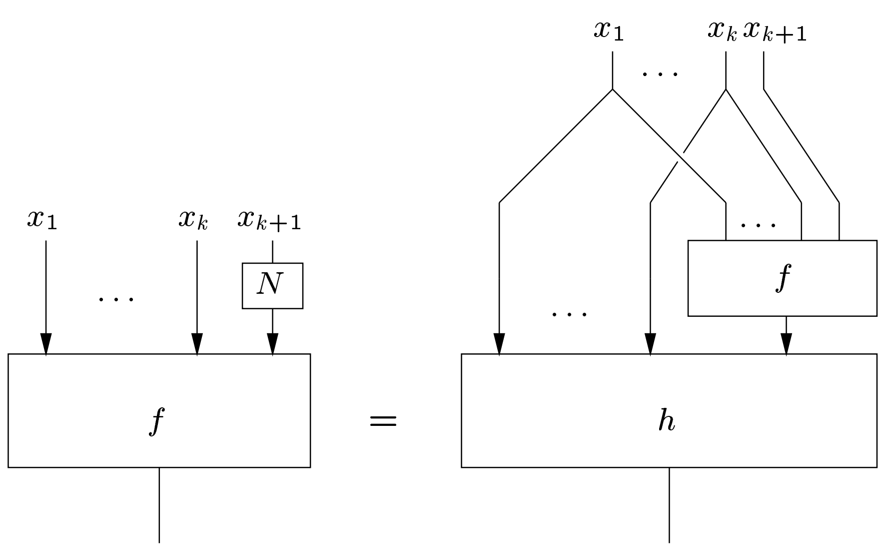
gegeben. Wir sagen: $f : \mathbb{N}^{k+1} \to \mathbb{N}$ wird durch das Schema der primitiven Rekursion erzeugt, wenn (1) und (2) gelten.

(1) $f \circ (1_k \times c_0) = g,$

(2) $f \circ (1_k \times N) = h \circ (1_{k+1} \times f) \circ d_{k+1}.$

Erläuterung durch die Diagramme





Lemma: Es gibt zu g, h genau eine totale Funktion f, die das Rekursionsschema erfüllt.

Beweis: Durch Induktion nach x_{k+1}. Ist $x_{k+1} = c_0$, dann ist $f_{(x_1,\dots,x_{k+1})}$ eindeutig definiert, da $N(x_{k+1}) \neq c_0$ ist für alle x_{k+1}. Ist $x_{k+1} \neq 0$, dann gibt es genau ein x^-_{k+1} mit $x_{k+1} = N(x^-_{k+1})$. Unter der Annahme, daß $f_{(x_1,\dots,x_k,x^-_{k+1})}$ definiert ist, folgt, daß $f_{(x_1,\dots,x_{k+1})}$ definiert ist.

Da die Nachfolgerfunktion ganz $\mathbb{N}$ aus c_0 erzeugt, ist also f für alle $(x_1, \dots, x_{k+1}) \in \mathbb{N}^{k+1}$ wohldefiniert.

Definition: $\mathbb{P}$ ist die kleinste Menge von Abbildungen $f : \mathbb{N}^k \to \mathbb{N}^j$ für $k, j \in \mathbb{N}$, die die Basisfunktionen E enthält und die abgeschlossen ist unter den Kompositionen und dem Schema der primitiven Rekursion.

Beispiele:

1) $1_k = \underbrace{1 \times \ldots \times 1}_{k \text{ mal}} \qquad 1_k \in \mathbb{P}$

Beweis: Es ist $1 \in E$. Setzen wir $1_1 = 1$ und $1_k = 1_k \times 1$, dann folgt induktiv, daß $1_k \in \mathbb{P}$ gilt.

2) $d_k \in \mathbb{P}$.

Beweis: Es ist $d_1 \in E, 1_k \in \mathbb{P}, T \in E$. Aus dem Abschluß von $\mathbb{P}$ gegenüber „$\times$“ folgt

$$\begin{aligned} f &:= d_{k-1} \times 1 \in \mathbb{P}, \\ g &:= 1_{2\cdot(k-1)} \times d_1 \in \mathbb{P}, 1_{k-2} \times T \in \mathbb{P}. \end{aligned}$$

Weiter gilt

$$1,\ T \in \mathbb{P} \Rightarrow 1_1 \times T \in \mathbb{P}.$$

Induktiv folgt nun

$$T_{k-1,1} = (T_{k-2,1} \times 1) \circ (1_{k-2} \times T) \in \mathbb{P},$$

und schließlich $d_k \in \mathbb{P}$.

3) $p_i^n : \mathbb{N}^n \to \mathbb{N}$ **Projektion** auf die i-te Komponente, d. h. $p_i^n(x_1, \ldots, x_n) = x_i$ für $1 \leq i \leq n$.

Es ist

$$p_i^n = \underbrace{p_0 \times \ldots \times p_0}_{i-1} \times 1 \times \underbrace{p_0 \times \ldots \times p_0}_{n-i} \in \mathbb{P}.$$

4) Die **Vorgängerfunktion**

$$V : \mathbb{N} \to \mathbb{N} \text{ mit } V(n) = \begin{cases} n-1 & \text{für } n > 0 \\ 0 & \text{für } n = 0. \end{cases}$$

Es ist $V \in \mathbb{P}$.

Beweis:

$$\begin{aligned} V \circ c_0 &= c_0 \\ V \circ N &= 1 \quad \text{(Identität).} \end{aligned}$$

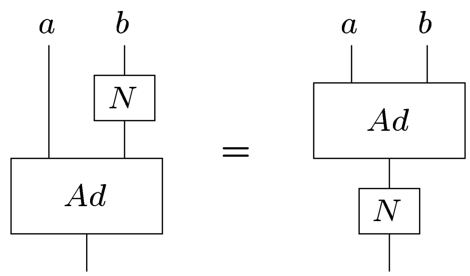
Da $c_0 \in E$ und $\mathbb{P}$ abgeschlossen sind unter der Anwendung der primitiven Rekursion, ist $V \in \mathbb{P}$.

Schreibweise: $V(a, b) = a \dot{-} 1$

5) **Addition:**

$$\begin{aligned} Ad &: \mathbb{N} \times \mathbb{N} \to \mathbb{N} \\ Ad &\circ (1 \times c_0) = 1, \\ Ad &\circ (1 \times N) = N \circ Ad. \end{aligned}$$

Schreibweise: $a + b := Add(a, b)$.



6) **Modifizierte Differenz**

$$Dif : \mathbb{N}^2 \to \mathbb{N}, \quad Dif(x, y) := \begin{cases} x - y & \text{für } x \geq y \\ 0 & \text{sonst.} \end{cases}$$

Schreibweise: $x \dot{-} y := Dif(x, y)$: Es gilt $Dif \in \mathbb{P}$.

Beweis:

$$\left. \begin{aligned} Dif \circ (1 \times c_0) &= 1, \\ Dif \circ (1 \times N) &= V \circ Dif \end{aligned} \right\} \text{Schema der prim. Rekursion}$$

7) **Multiplikation**

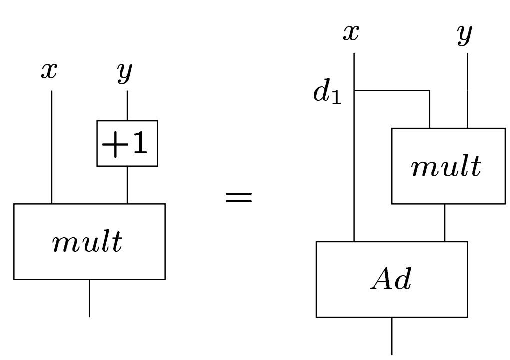
$$\left.\begin{array}{rcl} mult \circ (1 \times c_0) & = & p_0 \times c_0, \\ mult \circ (1 \times N) & = & Ad \circ (1 \times mult) \\ & & \circ (d_1 \times 1) \end{array}\right\} \begin{array}{l}\text{Schema} \\ \text{der prim.} \\ \text{Rek.}\end{array}$$

$mult : \mathbb{N}^2 \to \mathbb{N}.$

Also $mult \in \mathbb{P}$.

Erläuterung:

$$x \cdot 0 = 0$$
$$x \cdot (y + 1) = x + xy.$$



8) $\pi : \mathbb{N}^k \to \mathbb{N}^k$: π permutiert die Komponenten. Man zeigt $\pi \in \mathbb{P}$.

9) **Exponentialfunktion** $ex(x,y) = x^y$.
$ex(x,y)$ wird durch das Schema

$$\begin{aligned} ex \circ (1 \times c_0) &= p_0(\times N \circ c_0), \\ ex \circ (1 \times N) &= mult \circ (1 \times ex) \circ (d_1 \times 1) \end{aligned}$$

definiert. Also $ex \in \mathbb{P}$.

Man beobachtet,daß das Schema zur Definition der Exponentialfunktion relativ zur Multiplikation gleich dem zur Definition der Multiplikation relativ zur Addition ist.

Wir betrachten die Iteration dieses Konstruktionsverfahrens:

$$\begin{aligned} F_0(a,b) &:= a+1, \\ F_1(a,b) &:= a+b, \\ F_2(a,b) &:= a \cdot b, \\ F_3(a,b) &:= a^b \end{aligned}$$

allgemeingültig für $n > 2$

$$\begin{aligned} F_{n+1}(a,0) &:= F_n(a,0) \\ F_{n+1}(a,b+1) &= F_n(a,F_{n+1}(a,b)) \end{aligned}$$

Wir setzen nun

$$F(n,a,b) := F_n(a,b)$$

und erhalten eine Abbildung

$$F : \mathbb{N}^3 \to \mathbb{N},$$

die durch die Rekursion

$$\begin{aligned} F(i,a,b) &= F_i(a,b) \qquad \text{für } i = 1,2,3, \\ F(n+1,a,0) &= F(n,0,0), \\ F(n+1,a,b+1) &= F(n,a,F(n+1,a,b)) \end{aligned}$$

gegeben ist.

Dieses Rekursionsschema verwendet eine simultane Rekursion in zwei Variablen und hat also nicht die Gestalt der primitiven Rekursion. Natürlich folgt daraus noch nicht, daß $F \notin \mathbb{P}$ gilt. Wir kommen hierauf später zurück.

10) Die Funktion

$$sg(x) = \begin{cases} 1 & \text{für } x > 0 \\ 0 & \text{für } x = 0 \end{cases}$$

liegt in $\mathbb{P}$.

Beweis: Das Schema

$$\begin{aligned} sg \circ c_0 &= c_0 \\ sg \circ N &= N \circ c_0 \end{aligned}$$

definiert $sg : \mathbb{N} \to \mathbb{N}$ und hat die Form der primitiven Rekursion.

11) Sei

$$c_<(x,y) = \begin{cases} 1 & \text{für } x < y, \\ 0 & \text{für } x \geq y. \end{cases}$$

Beh.: $c_<(x,y) \in \mathbb{P}$.

Beweis: Es ist $c_< = sg \circ Dif \circ T$, da

$$c_<(x,y) = sg(x \dot{-} x).$$

Primitiv-rekursive Relationen

Sei $R \in \mathbb{N}^k$ und $\chi_R : \mathbb{N}^k \to \mathbb{N}$ sei definiert durch

$$\chi_R(x) = \begin{cases} 1 & \text{für } x \in R, \\ 0 & \text{für } x \notin R. \end{cases}$$

χ_R heißt die charakteristische Funktion von R.

Definition: R heißt **primitiv-rekursive Relation** $:\Longleftrightarrow\ \chi_R \in \mathbb{P}$.

Lemma: Sind $R_1, R_2 \subset \mathbb{N}^k$ prim.-rek. Relationen, dann sind auch

$$R_1 \cup R_2,\ R_1 \cap R_2,\ \overline{R}_1 := \mathbb{N}^k - R_1$$

prim.-rek. Relationen.

Beweis: Wir müssen zeigen, daß die zu diesen Mengen gehörigen charakteristischen Funktionen prim.-rek. sind.

Man erkennt

$$\begin{aligned} \chi_{R_1 \cup R_2} &= sg \circ (\chi_{R_1} + \chi_{R_2}) \circ d_k \\ \chi_{R_1 \cap R_2} &= ((\chi_{R_1} + \chi_{R_2}) \dot{-} 1) \circ d_k \\ \chi_{\overline{R}_1} &= 1 \dot{-} \chi_{R_1}. \end{aligned}$$

Lemma: Ist für $\lambda \in \{<, >, \leq, \geq, =\}$

$$R_\lambda := \{(x, y) \in \mathbb{N}^2 \mid x \lambda y\},$$

dann ist $\chi_{R_\lambda} \in \mathbb{P}$.

Beweis:...

Definition durch Fallunterscheidung

Seien $R_i \subset \mathbb{N}^k$ für $i = 1, \ldots, n$ prim.-rek. Relationen und $f_i : \mathbb{N}^k \to \mathbb{N}$ für $i = 1, \ldots, n$ aus $\mathbb{P}$. Weiter gelte $R_i \cap R_j = \emptyset$ für $i \neq j$.

Lemma: Die durch

$$g(x) := \begin{cases} f_i(x) & \text{für } x \in R_i (i = 1, \ldots, n) \\ 0 & \text{sonst} \end{cases}$$

definierte Abbildung $g : \mathbb{N}^k \to \mathbb{N}$ liegt in $\mathbb{P}$.

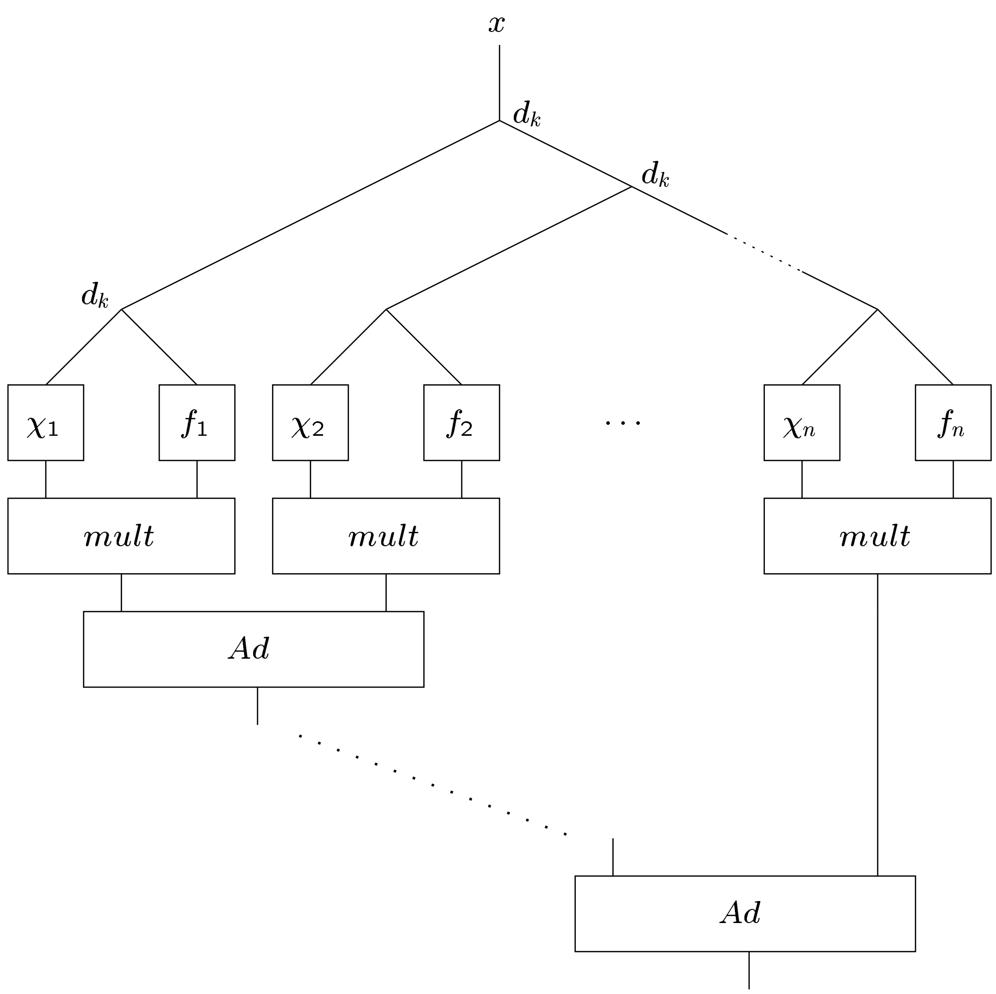
Beweis: Sei x_i die zu R_i gehörige prim.-rek. Funktion. Wegen $R_i \cap R_j = \emptyset$ für $i \neq j$ ist

$$\chi_1(x) + \ldots + \chi_n(x) \leq 1,$$

dann gilt

$$g(x) = \chi_1(x) \cdot f_1(x) + \cdots + \chi_n(x) \cdot f_n(x).$$

Wir geben eine funktionale Darstellung von g durch das folgende Diagramm an.



Die Ackermann'sche Funktion

Auf Hilberts Anregung konstruierte Ackermann 1928 eine rekursiv definierte Funktion, die nicht in $\mathbb{P}$ liegt.

Wir schildern zunächst die Idee dieser Konstruktion. Man konstruiere eine Folge von primitiv-rekursiven Funktionen

$$f_1, f_2, \ldots \qquad\qquad , \ f_i : \mathbb{N} \to \mathbb{N}$$

von der f_{i+1} durch das Schema der primitiven Rekursion aus f_i erzeugt wird. Somit steigt die Rekursionstiefe, die zur Definition der Folge verwendet wird, streng monoton mit i an. Betrachtet man nun

$$F(n,x) := f_n(x) \quad \text{für } x, n, \in \mathbb{N},$$

dann haben wir zur Definition von F beliebig tiefe primitive Rekursionen verwendet. Also sollte i. a. F nicht in $\mathbb{P}$ liegen.

Sei $f_1 : \mathbb{N} \to \mathbb{N}$ aus $\mathbb{P}$. Wir definieren

$$F_1(n,x) := f_1^{(n)}(x) := f_1(f_1^{(n-1)}(x));$$
$$f_1^{(0)}(x) := x, f_1^{(1)}(x) = f_1(x).$$

Nun setzen wir

$$f_2(x) := F_1(x,x).$$

F_1 ist primitiv-rekursiv, da F_1 durch das prim.-rek. Schema

$$\begin{aligned} F_1 \circ (c_0 \times 1) &= f_1 \\ F_1 \circ (N \times 1) &= f_1 \circ F_1 \end{aligned}$$

definiert wird. Weiter gilt

$$f_2 = F_1 \circ d_1.$$

Also ist $f_2 \in \mathbb{P}$.

Wir definieren den Operator Pr durch

$$Pr(f_1) := f_2.$$

Nun iterieren wir die Anwendung von Pr und erhalten die Folge

$$f_1, f_2, f_3, \dots, f_i \dots$$

mit $f_{i+1} := Pr(f_i)$ für $i = 1, 2, \dots$.

Nun setzen wir

$$F(n, x) := f_n(x)$$

und definieren

$$\mathcal{A}(f_1) := F.$$

$\mathcal{A}$ bezeichnen wir als Ackermannoperator.

Die Anwendung von $\mathcal{A}$ auf f_1 enthält ∞-viele Diagonalisierungen und primitiv-rekursive Definitionen. $F(n, x)$ kann aber durch endlich viele Rechenschritte bestimmt werden. Also erzeugt $\mathcal{A}$ aus berechenbaren Funktionen wieder berechenbare Funktionen.

Wir betrachten die Wirkung von $\mathcal{A}$ auf die Nachfolgerfunktion N. Sei also $f_1 = N$.

Wir erhalten

$$F_1(n, x) = N^{(n)}(x) = n + x$$

und

$$\begin{aligned} f_2(x) &= 2 \cdot x, \\ F_2(n, x) &= f_2^{(n)}(x) = f_2(f^{(n-1)}(x)) \\ &= 2 \cdot f^{(n-1)}(x) = 2^n \cdot x, \end{aligned}$$

$$\begin{aligned} f_3(x) &= 2^x \cdot x, \\ F_3(n, x) &= f_3^{(n)}(x) = 2^{f_3^{(n-1)}(x)} \cdot f_3^{(n-1)}(x) \end{aligned}$$

Wir erhalten

$$\begin{aligned} f_3^{(2)}(x) &= 2^{2^x \cdot x} \cdot 2^x \cdot x, \\ f_3^{(3)}(x) &= 2^{2^{2^x \cdot x} \cdot 2^x \cdot x} \cdot 2^{2^x \cdot x} \cdot 2^x \cdot x \end{aligned}$$

Die Funktion $F(n, x)$ wächst also mit n ungeheuer an.

Wie kann man zeigen, daß die Funktion F nicht in $\mathbb{P}$ liegt?

Wir haben eine Methode kennengelernt, die zu einer Menge F von Funktionen eine Funktion u_F konstruiert, die nicht in F liegt.

Wenn wir eine Folge

$$u_1, u_2, u_3, \ldots$$

effektiv konstruieren können, die ganz $\mathbb{P}$ ausschöpft, dann ist $u_{\mathbb{P}}$ eine Funktion, die berechenbar ist und die nicht in $\mathbb{P}$ liegt. Das kann man in der Tat tun. Wir gehen hier einen anderen Weg,der zum Ziel führt, indem man zeigt, daß das Wachstum der Funktion F größer ist als das jeder Funktion aus $\mathbb{P}$. In der Konstruktion von F haben wir ebenfalls das Konzept der Diagonalisierung benutzt. Wir folgen nicht genau der eben angegebenen Konstruktion, sondern ersetzen sie durch eine Konstruktion, die etwas übersichtlicher ist.

Definition: Die durch die folgende Rekursion definierte Abbildung $A : \mathbb{N}^2 \to \mathbb{N}$ heißt Ackermann'sche Funktion.

$$\begin{aligned} A(0,x) &= x^2, \\ A(n+1,0) &= 2, \\ A(n+1,x+1) &= A(n, A(n+1,x)). \end{aligned}$$

Wir verwenden auch die Bezeichnungen

$$A_n(x) := A(n,x),$$

$\forall_x^\infty$: für alle $x \in \mathbb{N}$ bis auf endlich viele Ausnahmen.

Lemma: $A(n,x)$ ist für $(n,x) \in \mathbb{N}^2$ berechenbar.

Beweis: $A(0,x)$ ist für alle x berechenbar.

Annahme: $A(n,x)$ ist für alle x berechenbar. Nun gilt für alle n

$$A(n+1,0) = 2$$

Nehmen wir an, daß $A(n+1,x)$ berechenbar ist, dann ergibt sich aus

$$A(n+1,x+1) := A(n, A(n+1,x))$$

und der Induktionsannahme über n, daß $A(n+1,x+1)$ berechenbar ist. Also ist $A(n+1,x)$ für alle x berechenbar. Also gilt das auch für alle n. Wir betrachten einige Beispiele:

$$A(1,x) = A(0, A(1,x-1)) = A(1,x-1)^2$$
$$= (A(1,x-2))^{2^2} = \ldots = A(1,0)^{2^x} = 2^{2^x}$$

$$\begin{aligned} A(2,x) &= A(1, A(2,x-1)) = 2^{2^{A(2,x-1)}} \\ &= 2^{2^{2^{2^{A(2,x-2)}}}} = 2^{\cdot^{\cdot^{\cdot^{2}}}} \} 2x+1 \end{aligned}$$

Definition: Für $f : \mathbb{N} \to \mathbb{N}$, $g : \mathbb{N}^k \to \mathbb{N}^l$ schreiben wir

$$g \lesssim f :\Longleftrightarrow \forall^\infty x \in \mathbb{N}^k : |g(x)| \leq f(|x|),$$

worin

$$|x| := x_1 + x_2 + \ldots + x_k \quad \text{für } x = (x_1, \ldots, x_k) \in \mathbb{N}^k.$$

Lemma: $\mathcal{A}_n$ hat die folgenden Eigenschaften: 1) $A_{n+1}(0) = 2$ und $A_{n+1}(x+1) = (A_n \circ A_{n+1})(x)$,
2) $A_{n+1}(x) = (A_n^{(p)} \circ A_{n+1})(x-p)$,
3) $A_n(x+1) > A_n(x)$ für $x \in \mathbb{N}$,
4) $A_{n+1}(x) \geq 2^x$ für $x, n \in \mathbb{N}$,
5) $p \in \mathbb{N} : A_n^{(p)}(x) \leq A_{n+1}(x)$ für $\forall^\infty x \in \mathbb{N}$
6) $p \in \mathbb{N} : A_n^{(p \cdot x)}(x) < A_{n+1}^2(x)$ für $\forall^\infty x \in \mathbb{N}$.

Beweis:

1) folgt unmittelbar aus der Definition von A.

2) folgt aus 1)

3) Man zeigt durch verschachtelte Induktion zusätzlich $A_n(x) > x + 1$.

Verankerung:

$$\begin{aligned} A_0(x) &= x^2 > x + 1 \qquad \text{für } x > 1. \\ A_0(x+1) &= (x+1)^2 > x^2 = A_0(x). \end{aligned}$$

Induktionsschritt:

$$A_{n+1}(x+1) = A_n(A_{n+1}(x)).$$

Nehmen wir an

$$A_n(x+1) > A_n(x) > x + 1,$$

dann folgt

$$A_{n+1}(x+1) > A_{n+1}(x) + 1 > A_{n+1}(x).$$

Nun ist

$$A_{n+1}(0) = 2 > 0 + 1.$$

Nehmen wir

$$A_{n+1}(x) > x + 1$$

an, dann folgt wegen

$$A_{n+1}(x+1) > A_{n+1}(x) + 1 > x + 2$$

die Behauptung für alle x. 4) $A_{n+1}(x) \geq 2^x$.
Für $n = 0$ haben wir

$$A_1(x) = 2^{2^x} > 2^x$$

Induktionsschritt:

$$\begin{array}{rcl} A_{n+1}(x+1) & = & A_n(A_{n+1}(x)) \\ & > & 2^{A_{n+1}(x)} > 2^{x+1} \quad \text{(nach 3)).} \end{array}$$

5) $A_{n+1}(x) = A_n^{(p+1)}(A_{n+1}(x \dot{-} p \dot{-} 1))$

$$\forall_x^\infty A_{n+1}(x \dot{-} p \dot{-} 1) \gtrsim 2^{x \dot{-} p \dot{-} 1} \gtrsim x.$$

Also

$$\begin{aligned} \forall_x^\infty A_{n+1}(x) &> A_n^{(p+1)}(x) = A_n(A_n^{(p)}(x)) \\ &> 2^{a_n^{(p)}(x)} > A_n^{(p)}(x). \end{aligned}$$

6) $\forall p \in \mathbb{N},\ \forall^\infty x \in \mathbb{N}$

$$A_n^{(p \cdot x)} < A_{n+1}^2(x)$$

Beweis:

$$A_{n+1}(A_{n+1}(x)) = A_n^{(p \cdot x)}(A_{n+1}(x \dot{-} p \cdot x)) > A_n^{(p \cdot x)}(x)$$

nach 4).

Wir definieren eine Hierarchie von Funktions-klassen

$$E_0 \subset E_1 \subset \ldots \subset \mathbb{P}.$$

Hierbei verwenden wir die Notation

$$Pr(g,h) := f,$$

$$\iff \quad \begin{array}{rcl} f \circ (1_k \times c_0) & = & g, \\ f \circ (1_k \times N) & = & h \circ (1_{k+1} \times f) \circ d_{k+1}. \end{array}$$

Wir setzen $B_0 := E \cup \{+, \cdot, \dot{-}\}$ und

$$\begin{array}{rcl} E_0 & := & \langle B_0, \{\circ, \times\}\rangle \\ & = & \text{aus } B_0 \text{ mittels } \circ, \times \text{ erzeugte Menge} \\ & & \text{von Funktionen.} \end{array}$$

Nun definieren wir

$$\begin{array}{rcl} Pr(E_n) & := & \{f \mid \exists g, h \in E_n \text{ mit } f = Pr(g, \cdot h)\}, \\ B_{n+1} & := & E_n \cup Pr(E_n), \end{array}$$

und

$$E_{n+1} := \langle B_{n+1}, \{\circ, \times\}\rangle.$$

Offensichtlich gilt

$$E_n \subset E_{n+1} \text{ und } \bigcup_{i=0}^{\infty} E_i = \mathbb{P}.$$

$rek(f) := Min\{n \mid f \in E_n\}$ heißt Rekursionszahl von f.

Lemma: $A_n \in E_n$.

Beweis: Es gilt:

(1) $A_0 \in E_0$.
(2) $A_{n+1} \circ N = A_n \circ A_{n+1}$

Hieraus folgt induktiv

$$A_{n+1} \in E_{n+1}$$

Lemma: Zu allen $f \in E_n$ gibt es ein $p \in \mathbb{N}$, so daß

$$|f| \lesssim A_n^{(p)}.$$

Beweis:

(1) $n = 0$

Die Elemente aus E_0 wachsen höchstens wie ein Polynom. Also gibt es $p \in \mathbb{N}$ mit

$$\forall_x^\infty |f(x)| \leq |x|^{2^p} = A_0^{(p)}(|x|)$$

nach Def. äquivalent mit

$$|f| \lesssim A_0^{(p)}.$$

(2) $n > 0$: Abschluß unter „$\circ$" und „$\times$".

Seien $f_1, f_2 \in E_n$ mit $Q(f_1) = Z(f_2)$

$$|f_1| \lesssim A_n^{(p)}, |f_2| \lesssim A_n^{(p)}.$$

Es folgt dann

$$\begin{array}{rcl} \forall_x^\infty |f_1 \circ f_2(x)| & \leq & A_n^{(p)}(|f_2(x)|) \leq A^{(p)} \circ A^{(p)}(|x|) \\ & = & A^{2p}(|x|) \\ |f_1 \times f_2| & = & |f_1| + |f_2| \lesssim 2 \cdot A^{(p)} \end{array}$$

(3) $n \geq 0$: Anwendung der pr.-Rekursion. Seien $g, h \in E_n$,

$f(x, 0) = g(x), f(x, m+1) = h(x, m, f(x, m))$.

Wir betrachten die folgende Iteration:

$$it(x, n, y) := (x, n+1, h(x, n, y)).$$

Funktional geschrieben ergibt das

$$it = (1 \times N \times h) \circ (d_2 \times 1) \in E_n.$$

Nun setzen wir

$$\begin{aligned} \overline{it}(x, m, y, 0) &:= (x, m, y) \\ \overline{(}x, m, y, r+1) &:= it(\overline{it}(x, m, y, r)). \end{aligned}$$

Man sieht, daß jede Komponente von $\overline{it}$ in E_{n+1} liegt und also auch $\overline{it} \in E_{n+1}$ gilt. Die primitive Rekursionm kann also durch eine Iteration und eine Projektion dargestellt werden:

$$\overline{it}(x, 0, g(x), n) = (x, n, f(x, n)).$$

Beh.: $it \lesssim A_m^{(p)} \Rightarrow \overline{it} \lesssim A_{m+1}^{(q)}$ für geeignetes $q \in \mathbb{N}$.

Beweis:

$$\begin{aligned} |\overline{it}(x, 0, g(x), m)| &= |it^{(m)}(x, 0, g(x))| \\ &= |it^{(m)} \circ (1_k, c_0, 1) \\ &\quad \circ (1_k \times g) \circ d_k| \\ &\lesssim a_n^{(p \cdot m)} \circ A_n^{(p')} = A_n^{(p \cdot m + p')}, \end{aligned}$$

worin p' so gewählt wurde, daß für $g \in E_n$

$$|g| \lesssim A_n^{(p')}$$

gilt.

Hieraus folgt:
Satz: $E_0, \subsetneq E_1, \subsetneq \ldots \subsetneq E_n \subsetneq \ldots$

Beweis: Es genügt zu zeigen, daß $A_{n+1} \notin E_n$. Wir nehmen an, daß $A_{n+1} \in E_n$ gilt. Da E_n unter $\{\circ, \times\}$ abgeschlossen ist, folgt $A_{n+1}^{(p)} \in E_n$ für $p \in \mathbb{N}$.

Nun gilt nach dem vorigen Lemma, Punkt 6)

$$A_{n+1}^{(p\cdot x)}(x) \lesssim A_{n+1}^{(2)}(x) \qquad (*)$$

und nach Punkt 4)

$$A_{n+1}(x) \gtrsim 2^x,$$

so daß

$$A_{n+1} \circ A_{n+1}^{(m)} \gtrsim A_{n+1}^{(m)},$$

d. h.

$$A_{n+1}^{(p\cdot x)}(x) \gtrsim A_{n+1}^{(p)}(x) \gtrsim A_{n+1}^{(2)}(x),$$

was zusammen mit $(*)$ zum Widerspruch führt. Also ist die Annahme $A_{n+1} \in E_n$ falsch.

Satz (Ackermann): $A \notin \mathbb{P}$.

Beweis: $A \in \mathbb{P} \Rightarrow \exists_n A \in E_n$

$$\Rightarrow A_{n+1} = A \circ (c_{n+1} \times 1) \in E_n.$$

Wir haben aber gezeigt, daß $A_{n+1} \notin E_n$ gilt.

Sei für $n, m \in \mathbb{N}$.

$$\begin{aligned} & b_2(n,0) := \binom{n+1}{2} \\ \text{und} \quad & b_2(n,m) = b_2(n+m,0) + m. \end{aligned}$$

Damit ist (siehe Übungen)

$$b_2 : \mathbb{N}^2 \longrightarrow \mathbb{N}$$

eine primitiv-rekursive Funktion.

Setzt man

$$s(x) := max\{i \in \mathbb{N} \mid \binom{i+1}{2} \leq x\}$$

und

$$\begin{aligned} b_{2,2}(x) &:= x - \binom{s(x)+1}{2}, \\ b_{2,1}(x) &:= s(x) - b_{2,2}(x), \end{aligned}$$

dann gilt (siehe Übungen)

$$[b_{2,1}, b_{2,2}] \circ d \circ b_2 = 1_{\mathbb{N}^2}, \quad b_2 \circ [b_{2,1}, b_{2,2}] = 1_{\mathbb{N}}.$$

Es ist also

$$b_2^{-1} = [b_{2,1}, b_{2,2}].$$

Nun definieren wir für $k > 2$

$$b_k(x_1, \ldots, x_k) := b_2(x_1, b_{k-1}(x_2, \ldots, x_k))$$

und

$$b_{k,i}(x) := \begin{cases} b_{2,1}(x) & \text{für } i = 1, \\ b_{k-1,i-1}(b_{2,2}(x)) & \text{für } 2 \leq i \leq k. \end{cases}$$

Mittels Induktion zeigt man den

Satz: $b_k, b_{k,i} \in \mathbb{P}$ für $k \geq 2$ und $1 \leq i \leq k$.
Weiter gilt

$$b_k^{-1} = [b_{k,1}, \ldots, b_{k,k}].$$

Definition: $b_k, b_{k,1}, \ldots, b_{k,k}$ heißen Standardreduktionen von $\mathbb{N}^k$ auf $\mathbb{N}$.

Satz: Jede Abbildung

$$f : D \subset \mathbb{N}^k \longrightarrow \mathbb{N}^j$$

läßt sich mittels der Standardreduktionen auf eine Abbildung

$$\tilde{f} : \tilde{D} \subset \mathbb{N} \longrightarrow \mathbb{N}$$

reduzieren. Es ist $f \in \mathbb{P} \iff \tilde{f} \in \mathbb{P}$.

Beweis: Setze

$$\tilde{f} := b_j \circ f \circ b_k^{-1}.$$

Es folgt

$$f := b_j^{-1} \circ \tilde{f} \circ b_k.$$

Korollar: Die Standardreduktionen führen primitiv-rekursive Relationen in primitiv-rekursive Relationen über.

Die angegebenen Reduktionen liegen alle in der Komplexitätsklasse E_1. Es gibt aber viele weitere primitiv-rekursive Reduktionen, d. h. Abbildungen $\beta : \mathbb{N}^k \longrightarrow \mathbb{N}$ mit $\beta, \beta^{-1} \in \mathbb{P}$.

Definition: Abbildungen f, g heißen primitiv-rekursiv äquivalent, wenn es primitiv-rekursive Reduktionen gibt, die f in g überführen. Entsprechend definiert man die primitiv-rekursive Äquivalenz von Relationen.

Die Standardreduktionen führen die Kategorie der primitiv-rekursiven Reduktionen in eine Gruppe $G_{\mathbb{P}}$ primitiv-rekursiver Abbildungen von $\mathbb{N}$ auf $\mathbb{N}$ über. Da die Standardreduktionen von niedriger Komplexität sind (sie liegen in E_1) wird bei dieser Reduktion die Komplexitätshierarchie der Reduktionen höchstens um 1 verschoben. Eine Untersuchung der Äquivalenzklassen unter $G_{\mathbb{P}}$ oder einer Untergruppe von $G_{\mathbb{P}}$ stellt eine Übertragung der Konzepte der Geometrie dar, wie sie im **Erlanger Programm** von Felix Klein formuliert wurde.

Die hier angedeutete Theorie wurde in Bezugnahme auf dieses Programm entwickelt. Die anschließend entwickelte Reduktionstheorie stellte den Komplexitätsaspekt betont in den Vordergrund. Da die Forderung, daß Reduktionen eine Gruppe bilden sollten, nicht gut mit dem Komplexitätsaspekt verträglich ist, hat man in der Weiterentwicklung der Reduktionstheorie die Gruppen aufgegeben und sich auf Kategorien bezogen, deren Elemente bestimmten Komplexitätsklassen angehören. Die bekannteste Klasse ist heute die Klasse der in Polynomialzeit berechenbaren Funktionen. Das berühmte Problem $P = NP$ ist in dem oben erläuterten Sinn ein geometrisches Problem. Wir können hier dieses Thema nicht weiter verfolgen.

Wir behandeln nun eine Erweiterung von $\mathbb{P}$, die auf dem Lösen von Gleichungssystemen beruht. Ist $f : \mathbb{N}^k \longrightarrow \mathbb{N}$ eine Abbildung, dann ist

$$R_f := \{\alpha \in \mathbb{N}^k \mid f(\alpha) = 0\} \subset \mathbb{N}^k$$

eine Relation. Nun kann man R_f auf verschiedene Weisen eine Funktion zuordnen. So setzen wir

$$\chi_f := \chi_{R_f}.$$

χ_f ist also die charakteristische Funktion von R_f. Ist f berechenbar, dann ist auch χ_f berechenbar, da $\chi_f = (1 - sig) \circ f$ gilt.

Nun interessieren wir uns für

$$\begin{aligned} D_f \ &:= \ \{\alpha' \in \mathbb{N}^{k-1} \mid \text{ Es gibt } \alpha_k \in \mathbb{N} \\ &\qquad \text{mit } (\alpha', \alpha_k) \in R_f\} \end{aligned}$$

und definieren $\tilde{f} : D_f \longrightarrow \mathbb{N}$ indem wir

$$\tilde{f}(\alpha') := min\{\alpha_k \mid (\alpha', \alpha_k) \in R_f\}.$$

Für $k \geq 1$ definieren wir den μ-Operator, indem wir

$$\mu f := \tilde{f}$$

setzen.

Für $k = 1$ erhalten wir die kleinste Nullstelle von f falls $R_f \neq \emptyset$ ist.

Ist f eine im intuitiven Sinne berechenbare Funktion und ist $\alpha' \in D_f$, dann läßt sich $\tilde{f}(\alpha')$ berechnen, indem man der Reihe nach die Folge

$$f(\alpha', 0), f(\alpha', 1), \ldots, f(\alpha', i), \ldots$$

berechnet. Diese Folge enthält die 0, da $\alpha' \in D_f$ ist. Ist i die erste Stelle mit $f(\alpha', i) = 0$, dann ist $\mu f(\alpha') = i$. (f ist total!)

Wissen wir allerdings nicht, ob $\alpha' \in D_f$ gilt, dann wird unsere Berechnung eventuell nicht abbrechen, so daß μf nur „partiell“ berechenbar ist. Können wir allerdings χ_{D_f} berechnen, dann können wir die nichtterminierenden Berechnungen vermeiden.

Definition: $f : D \subset \mathbb{N}^k \longrightarrow D' \subset \mathbb{N}^j$ heißt μ-**rekursiv** $:\Longleftrightarrow$ f läßt sich aus den Basisfunktionen E durch endlich häufige Anwendung der

- Kompositionsschemata „$\circ$“ und „$\times$“,
- des Schemas der **primitiven Rekursion** und
- des μ-**Operators**

erzeugen.

- Es gibt eine μ-rekursive Funktion $g : \mathbb{N}^k \longrightarrow \mathbb{N}^j$ mit $g(\mathbb{N}^k) = D'$.

Wir bezeichnen die Menge der μ-rekursiven Funktion durch $\mu\mathbb{P}$.

Definition: $D \subset \mathbb{N}^k$ heißt rekursiv **aufzählbar** $:\Longleftrightarrow$ Es gibt eine μ-rekursive Abbildung $f : \mathbb{N} \longrightarrow \mathbb{N}^k$ mit $f(\mathbb{N}) = D$.

Definition: Ist $f : \mathbb{N}^k \longrightarrow \mathbb{N}$ μ-rekursiv, dann heißt

$$R_f := \{\alpha \in \mathbb{N}^k \mid f(\alpha) = 0\}$$

μ-rekursive Relation.

Wir werden später beweisen, daß der folgende Satz gilt.

Satz: D ist genau dann eine μ-rekursive Relation, wenn D rekursiv aufzählbar ist.

Wir zeigen, daß die Menge der μ-rekursiven Funktionen in gewissem Sinne unvollständig ist. Da die hierbei verwendeten Konstruktionen nur zum Teil vom Begriff der Berechenbarkeit abhängen, geben wir ihnen eine allgemeine Fassung.

Wir gehen von der Menge

$$\mathfrak{F} := \{f : D \longrightarrow D' \mid D \subset \mathbb{N}^k, D' \subset \mathbb{N}, k \in \mathbb{N}\}$$

von Abbildungen aus. Mit Hilfe der Standardreduktion können wir $\mathfrak{F}$ auf

$$\mathfrak{F}_i := \{f : D \longrightarrow D' \mid D \subset \mathbb{N}^k, D' \subset \mathbb{N}, k \leq i\}$$

reduzieren. Mit $\mathfrak{a}_i$ bezeichnen wir abzählbare Teilmengen von $\mathfrak{F}_i$. Es bezeichnet

$$\gamma_i : \mathbb{N} \longrightarrow \mathfrak{a}_i$$

eine Abzählung, d. h. γ_i ist eine surjektive Abbildung. Wir wollen weiter voraussetzen, daß sich $\mathfrak{a}_i$ aus $\mathfrak{a}_{i+1}$ durch eine Standardreduktion erhalten läßt und daß diese Standardreduktionen γ_{i+1} in γ_i überführen.

Erläuterung:

$$(x_1, \ldots, x_k) \longrightarrow (x_1, \ldots, x_{k-2}, b_2^{-1}(x_{k-1}, x_k))$$

ist eine Standardreduktion von $\mathbb{N}^k$ auf $\mathbb{N}^{k-1}$.

Ist $f \in \mathfrak{a}_k$ und $\gamma_k(n) = f$, dann sei

$$\gamma_{k-1}(n) = \gamma_k(n)(x_1, \ldots, x_{k-2}, b_2^{-1}(x_{k-1}, x_k)).$$

Sei nun

$$D_n^1 := \{x \in \mathbb{N} \mid \gamma_1(n)(x) = 0\}.$$

D_n^1 ist also die Menge der Nullstellen der Funktion $\gamma_1(n)$. Wir definieren für $(n, x) \in \mathbb{N}^2$

$$e(n, x) := \begin{cases} 1 & \text{für } x \in D_n^1 \\ 0 & \text{sonst.} \end{cases}$$

Nun setzen wir

$$\tilde{e}(n) := e(n, n).$$

Da e total ist, gilt das auch für $\tilde{e}$.

Lemma: $\tilde{e} \notin \mathfrak{a}_1$.

Beweis: Nehmen wir $\tilde{e} \in \mathfrak{a}_1$ an, dann folgt, daß es ein $\tilde{n} \in \mathbb{N}$ gibt mit $\gamma(\tilde{n}) = \tilde{e}$.

Fallunterscheidung:

(1) $\tilde{e}(\tilde{n}) = 1 \Rightarrow \tilde{n} \in D_{\tilde{n}} \Rightarrow \gamma_1(\tilde{n})(\tilde{n}) = 0$
Nun ist aber

$$\gamma_1(\tilde{n})(\tilde{n}) = \tilde{e}(\tilde{n}).$$

Widerspruch!

(2) $\tilde{e}(\tilde{n}) = 0 \Rightarrow \tilde{n} \notin D_{\tilde{n}} \Rightarrow \tilde{e}(\tilde{n}) \neq 0$.
Widerspruch!

Sei nun

$$\tilde{u}(n,x) := \begin{cases} \gamma_1(n)(x) & \text{für } x \in Q(\gamma_1(n)) \\ 1 & \text{für } x \notin Q(\gamma_1(n)). \end{cases}$$

Es ist offensichtlich

$$e(n,x) = \overline{sig}(\tilde{u}(n,x)),$$

worin $\overline{sig} := 1 \dot{-} sig$ ist.

Hieraus folgt das

Korollar 1: Ist $\mathfrak{a}_2$ abgeschlossen unter „$\circ$“ und Diagonalisierung und enthält $\mathfrak{a}_2$ die Funktion $\overline{sig}$, dann ist $\tilde{u} \notin \mathfrak{a}_2$.

Korollar 2: Liegt

$$u(n,x) := \begin{cases} \gamma_1(n)(x) \text{ für } x \in Q(\gamma_1(n)), n \in \mathbb{N} \\ \text{nicht definiert sonst} \end{cases}$$

in $\mathfrak{a}_2$, dann gibt es unter den Voraussetzungen von Korollar 1 keine Fortsetzung von u mit Werten ≥ 1 zu einer totalen Funktion $\tilde{u} \in \mathfrak{a}_2$.

Definition: Die oben definierte Funktion u heißt universell in $\mathfrak{a}$.

Diese Bezeichnung ist gerechtfertigt,da es zu jeder Funktion $f \in \mathfrak{a}_1$ ein $n \in \mathbb{N}$ gibt, so daß

$$f(x) = u(n,x)$$

ist. Jede in diesem Sinne universelle Funktion definiert natürlich eine Aufzählung. Vermittels der Standardreduktion läßt sich auch jede Funktion aus $\mathfrak{a}_2$ auf u reduzieren.

Das Korollar 2 besagt, daß es unter den Voraussetzungen von Korollar 1, daß $\mathfrak{a}_2$ abgeschlossen ist unter „$\circ$“ und daß $\overline{sig}$ in $\mathfrak{a}_2$ liegt, keine Fortsetzung universeller Funktionen $\tilde{u}$ gibt, so daß $\tilde{u} \in \mathfrak{a}_2$ gilt, wenn diese Fortsetzungen die Nullstellenmengen von u erhalten.

Wir wollen die Ergebnisse auf Mengen berechenbarer Funktionen übertragen.

Die Menge der berechenbaren Funktionen müssen wir als abzählbar ansehen. Die oben definierten Mengen von Abbildungen $\mathfrak{a}_i$ bezeichnen wir im Falle der Berechenbarkeit aller ihrer Elemente mit $\mathcal{B}_i$.

Satz 1: Ist $\mathcal{B}_2$ eine Menge berechenbarer totaler Funktionen und ergibt sich $\mathcal{B}_1$ durch eine Standardreduktion aus $\mathcal{B}_2$, und ist $\mathcal{B}_2$ abgeschlossen unter der Diagonalisierung und ist $\overline{sig} \in \mathcal{B}_1$, dann gibt es in $\mathcal{B}_2$ keine Funktion $e(n, x)$, die die Nullstellen-Mannigfaltigkeiten der Funktionen aus $\mathcal{B}_1$ entscheidet.

Beweis: Angenommen e wäre in $\mathcal{B}_2$, dann würde $\tilde{e} := e \circ d_1 \in \mathcal{B}_1$ folgen, was, wie Lemma 1 zeigt, nicht gilt.

Satz 2: Ist $\mathcal{B}_2$ abgeschlossen unter „$\circ$“ und Diagonalisierung und enthält $\mathcal{B}_2$ die Funktion $\overline{sig}$, dann liegt die Fortsetzung jeder universellen Funktion, die die Nullstellen-Mannigfaltigkeit von u erhält nicht in $\mathcal{B}_2$.

Beweis: Aus Korollar 1 ergibt sich, daß die spezielle Fortsetzung $\tilde{u}$ von u nicht in $\mathcal{B}_2$ liegt. Haben wir irgendeine Fortsetzung $\tilde{u}$, die $x \notin Q(\gamma(n))$ einen Wert $\neq 0$ zuweist, dann erhalten wir auch in diesem Fall

$$e(n, x) = \overline{sig}(\tilde{u}(n, x)).$$

Wir diskutieren diesen Satz für $\mathcal{B}_2 = \mathbb{P} \cap \mathfrak{F}_2$. $\mathbb{P}_2$ erfüllt die Voraussetzungen von Satz 2. Wir erhalten

$$\tilde{u}(n, x) = \gamma_1(n)(x) \text{ für } x \in Q(\gamma_1(n)) = \mathbb{N}.$$

Also $\tilde{u}$ ist in diesem Fall eine totale Funktion. Unser Satz besagt, daß eine zu $\mathbb{P}_1$ universelle Funktion nicht in $\mathbb{P}_2$ liegt. Wenn es also eine universelle berechenbare Funktion zu $\mathbb{P}_1$ gibt, dann liegt sie nicht in $\mathbb{P}$.

Satz 3: Sei $\mu\mathbb{P}_2 = (\mu\mathbb{P}) \cap \mathfrak{F}_2$ die Menge der μ-rekursiven Funktionen höchstens zweier Variablen. Ist $u \in \mu\mathbb{P}_2$ und universell, dann ist u nicht total.

Beweis: Der Satz folgt unmittelbar aus Korollar 2.

Nach unserer Überzeugung läßt sich jede berechenbare Funktion auf einem Computer programmieren. Die Menge der Programme über $\mathbb{N}$ mit Ausgabe in $\mathbb{N}$ läßt sich leicht charakterisieren und also rekursiv aufzählen. Also läßt sich eine universelle Funktion $u(n,x)$ angeben, die berechenbar ist. Aus Korollar 2 folgt, daß $u(n,x)$ nicht total ist. Weiter ergibt sich, daß die charakteristische Funktion des Definitionsbereiches von u nicht berechenbar ist.

Wir haben also den

Satz 4: Die charakteristischen Funktionen der Definitionsbereiche der bezüglich $\mathcal{B}$ universeller berechenbarer Funktionen liegen **nicht** in $\mathcal{B}$.

Bemerkung: Die hier verwendeten Konstruktionen von Funktionen, die nicht in einem Bereich $\mathcal{B}$ liegen, kann man verwenden, um Hierarchien

$$\mathcal{B} = \mathcal{B}_1 \subsetneqq \mathcal{B}_2 \subsetneqq \ldots$$

aufzubauen. Das geschieht, indem man die nicht in $\mathcal{B}_i$ liegende charakteristische Funktion χ_i zu $\mathcal{B}_i$ hinzunimmt und $\mathcal{B}_{i+1}$ als die Funktionsmenge definiert, die sich aus $\mathcal{B}_i \cup \{\chi_i\}$ unter Anwendung von „$\circ$“ und „$\times$“ erzeugen läßt.

Zunächst aber benötigen wir ein besseres Verständnis dessen, was Rechenmaschinen sind.

Wir haben den Berechenbarkeitsbegriff bisher auf der primitiven Rekursion und dem $\mu-$Operator aufgebaut. Wir haben gezeigt, daß im Rahmen dieser Theorie Funktionen definiert werden, die nur partiell berechenbar sind in dem Sinne, daß die Theorie keine Methoden an die Hand gibt, diese Einschränkung der Berechenbarkeit zu beheben. Wir haben darüberhinaus gezeigt, daß dieser Mangel grundsätzlicher Natur ist, daß er nämlich auch in jeder Erweiterung des Berechenbarkeitsbegriffs auftreten wird.

Wir sind aber durchaus nicht sicher, daß wir mit der Definition der μ-rekursiven Funktionen alle im intuitiven Sinne berechenbaren Funktionen eingefangen haben. Daß das so ist, darin besteht heute Einigkeit. Wir müssen uns aber im Rahmen der Vorlesung diese Einsicht noch verschaffen.

Es besteht wohl Konsens darin, daß man jede berechenbare Funktion auf unseren Computern programmieren kann, soweit man von ihrer Endlichkeit absieht. Wir werden deshalb das idealisierte Konzept der Maschine präzisieren und anschließend zeigen, daß jede auf diesen Maschinen berechenbare Funktion μ-rekursiv ist, oder besser gesagt, auf eine solche Funktion reduziert werden kann.

Wir betrachten zunächst **Maschinen mit endlichem Speicher**.

Sei

$$K_1 := \{\xi : \{x_1, \ldots, x_n\} \longrightarrow \mathbb{N}\}$$

die Menge der Belegungen der Variablen $x_1, \ldots, x_n$ mit Werten aus $\mathbb{N}$. K_1 ist für uns die Menge der möglichen Speicherinhalte. Weiter sei $K_2 := [0 : N] \subset \mathbb{N}$; K_2 fassen wir als die Menge der Operationen der Maschine auf.

Wir nennen

$$K = K_1 \times K_2$$

die Konfigurationsmenge der Maschine.

$$K_e := \{(\xi, 0) \mid \xi \in K_1\}$$

heißt die Menge der **Endkonfigurationen**,

$$K_a := \{(\xi, 1) \mid \xi \in K_1\}$$

die Menge der **Anfangskonfigurationen**.

Die Funktion der Maschine wird durch eine Abbildung

$$\Delta : K \longrightarrow K$$

gegeben. Wir verlangen

$$\Delta(k) = k \text{ für } k \in K_e.$$

In einer realen Maschine wird durch jedes $\xi \in K_2$ eine spezielle Operation veranlaßt. Beispiele dafür sind

$$\begin{aligned} x_1 &:= x_2 * x_3, \\ \text{while } x_1 = 0 \text{ do } x_2 &:= x_3 * x_2; x_1 := x_1 - 1 \text{ od}, \end{aligned}$$

oder

$$x_i := x_{\xi(x_j)}.$$

In den beiden ersten Fällen ist klar, daß es sich um μ-rekursive Funktionen handelt. In dem dritten Fall ist das nicht so offensichtlich. Ist n fest, kann man die indizierte Zuweisung auf eine n-fache Fallabfrage zurückführen. Somit ist diese Zuweisung für unsere Maschine, die mit festem n rechnet, μ-rekursiv. Ist n variabel, dann benötigen wir die Zeitvariable t. Nach dem t-ten Rechenschritt steht im Speicher ξ_t.

Wir haben

$$\xi_0 = \text{ Eingabe}$$

bei der Anwendung des indizierten Befehles

$$\xi_{t+1}(x_i) := \begin{cases} \xi_t(x_i) \text{ für } i \neq \xi_t(x_j) \\ \xi_t(\xi_t(x_j)) \text{ sonst.} \end{cases}$$

Also diese Zuweisung ist eine primitiv-rekursive Funktion. Wir sehen also, daß die Operationen einer realen Maschine alle μ-rekursiv sind.

Wir können also davon ausgehen, daß sich Δ durch eine endliche Fallunterscheidung mittels primitiv-rekursiven Prädikaten ($\xi(x_i) \gtreqless 0$) und Funktion aus $\mathbb{P}$ definieren läßt. Also können wir annehmen, daß $\Delta \in \mathbb{P}$ gilt.

Definition: $\mathfrak{M} := (K, K_a, K_e, \Delta)$ heißt $\mathbb{P}$-Maschine $:\Longleftrightarrow \; \Delta \in \mathbb{P}$.

$\mathfrak{M}$ heißt $\mu\mathbb{P}$-*Maschine* $:\Longleftrightarrow \; \Delta \in \mu\mathbb{P}$.

Eine **Berechnung** von $\mathfrak{M}$ ist eine Folge

$$(\xi_1, \zeta_1). \xrightarrow{\Delta} (\xi_2, \zeta_2) \xrightarrow{\Delta} \ldots \xrightarrow{\Delta} (\xi_m, \zeta_m),$$

Eine Berechnung heißt regulär $:\Longleftrightarrow \; \zeta_1 = 1 \;\&\; \zeta_m = 0$.

Wir definieren weiter

$$T(\xi) := \begin{cases} min\{m \mid (\xi,1) \xrightarrow{\Delta} (\xi_1,\zeta_1) \longrightarrow \ldots \\ \qquad\qquad \longrightarrow (\xi_m,\zeta_m) \text{ ist regulär}\} \\ \infty \text{ falls das Minimum nicht existiert.} \end{cases}$$

T heißt **Laufzeitfunktion** von $\mathfrak{M}$.

Die Funktion

$$f : D \longrightarrow \mathbb{N}^j, D \subset \mathbb{N}^k$$

heißt $\mathfrak{M}$-**berechenbar** $:\iff$ Für

$\xi(x_i) = \alpha_i$ für $i = 1, \ldots, k$
und $\xi(x_i) = 0$ für $i = k+1, \ldots, n$
und $\alpha \in D$ ist $T(\xi) \neq \infty$ und es gilt
$f(\alpha) = (\xi_m(x_1), \ldots, \xi_m(x_j))$ für $m = T(\xi)$.

Satz: Jede mit einer $\mu\mathbb{P}$-Maschine berechenbare Funktion ist μ-rekursiv.

Beweis: Wir definieren

$$F(\xi,0) := (\xi,1),$$
$$F(\xi,t+1) := \Delta(F(\xi,i)).$$

Es gilt also $\Delta^{(m)} = F(\xi,m)$. F wird durch eine primitive Rekursion auf einer μ-rekursiven Funktion erzeugt und ist also μ-rekursiv.

Die Funktion T ist μ-rekursiv, denn F ist total und es gilt

$$T = \mu F.$$

Die μ-rekursive Funktion

$$F(\xi, T(\xi))$$

liefert den Endzustand der Maschine. $f(\alpha)$ erhalten wir durch die Projektion von K_1 auf die ersten j Komponenten. Also ist f μ-rekursiv.

Wir erhalten auf die gleiche Weise

Satz: Ist f durch eine $\mathbb{P}$-Maschine berechenbar, dann ergibt sich f durch die einmalige Anwendung des μ-Operators auf eine primitiv-rekursive Funktion und eine anschließende Projektion.

Wir zeigen nun, daß sich jede μ-rekursive Funktion durch eine einmalige Anwendung des μ-Operators und eine nachfolgende Projektion berechnen läßt. Das ergibt sich unmittelbar aus dem folgenden

Satz: Jede μ-rekursive Funktion läßt sich mittels einer $\mathbb{P}$-Maschine berechnen.

Beweis: Ist f μ-rekursiv, dann gibt es eine Folge von Definitionen von Funktionen, die aufbauend auf E durch die Anwendung der zugelassenen Operationen f definieren. Wir führen den Beweis induktiv nach der Länge dieser Konstruktion. O.b.d.A. dürfen wir annehmen, daß jede Maschine nach dem Start mit $\zeta = 1$ nicht mehr in einen Anfangszustand zurückkehrt.

Die Aussage des Satzes gilt trivialerweise für die Funktionen aus E. Wir nehmen an, daß die Behauptung richtig ist für alle Funktionen, deren Konstruktion höchstens l Schritte erfordert. Daraus leiten wir ab, daß die Behauptung auch für $l + 1$ gilt.

(1) Wir betrachten zunächst die Operation

$$f := \tilde{f} \times \overline{f}.$$

Für $\tilde{f}$ und $\overline{f}$ gebe es eine $\mathbb{P}$-Maschine $\tilde{\mathfrak{M}}$ bzw. $\overline{\mathfrak{M}}$, die $\tilde{f}$ bzw. $\overline{f}$ berechnet. Wir definieren

$$\mathfrak{M} := \tilde{\mathfrak{M}} \times \overline{\mathfrak{M}}$$

durch die folgenden Ausdrücke

$$\begin{array}{rcl} K_1 & := & \tilde{K}_1 \times \tilde{K}_2 \times \overline{K}_1 \times \overline{K}_2, \\ K_2 & := & b_2^{-1}(\tilde{K}_2 \times \overline{K}_2), \\ \Delta(\xi,\zeta) & := & ((\tilde{\Delta}(\tilde{\xi},\tilde{\zeta}), \overline{\Delta}(\overline{\xi},\overline{\zeta})), b_2^{-1}(\tilde{\zeta}_1, \overline{\zeta}_1)). \end{array}$$

Dabei sei $\tilde{\zeta}_1 := \tilde{\Delta}_2(\tilde{\xi},\tilde{\zeta})$ und $\bar{\zeta}_1 := \bar{\Delta}_2(\bar{\xi},\bar{\zeta})$. Damit ist $\mathfrak{M}$ eine $\mathbb{P}$-Maschine, da das nach Voraussetzung für $\tilde{\mathfrak{M}}$ und $\overline{\mathfrak{M}}$ gilt.

Wegen

$$b_2^{-1}(\tilde{\zeta}, \overline{\zeta}) = 0 \iff \tilde{\zeta} = 0 \;\&\; \overline{\zeta} = 0$$

hält die Maschine genau dann an, wenn $\tilde{\mathfrak{M}}$ und $\overline{\mathfrak{M}}$ angehalten haben.

Durch eine geeignete Projektion von K_1 erhalten wir $f(\xi) = (\tilde{f}(\tilde{\xi}), \overline{f}(\overline{\xi}))$.

(2) Wir betrachten nun

$$\tilde{f} : \tilde{D}_1 \longrightarrow \tilde{D}_2, \overline{f} : \overline{D}_1 \longrightarrow \overline{D}_2$$

mit $\tilde{D}_2 = \overline{D}_1$ und $f := \overline{f} \circ \tilde{f}$.

Wir definieren zunächst zu vorgegebenen Maschinen $\mathfrak{M}$ eine Maschine $\mathfrak{M}_c$, die primitiv-rekursive Funktionen berechnet. Man erhält $\mathfrak{M}_c$ aus $\mathfrak{M}$, indem man $\mathfrak{M}$ so abändert, daß die Operation von $\mathfrak{M}$ in Abhängigkeit von einem neu eingeführten Parameter r gerade r-mal ausgeführt wird.

Wir setzen

$$K_{c1} := K \times \mathbb{N}^2, K_{c2} := \{0, 1, 2\}$$

und

$$\Delta_c(\xi, \zeta, \varrho, r, \zeta') := \begin{cases} (\xi, \zeta, \varrho, r, \zeta') \text{ für } \zeta' = 0 \\ (\xi, \zeta, r, r, 2) \text{ für } \zeta' = 1 \\ (\Delta(\xi, \zeta), \varrho - 1, r, 2) \text{ für } \varrho \neq 0 \\ (\xi, \zeta, 0, r, 0) \text{ für } \varrho = 0. \end{cases}$$

$\mathfrak{M}_c$ ist eine $\mathbb{P}$-Maschine und berechnet Funktionen in $\mathbb{P}$, da das Anhalten durch den beschränkten μ-Operator bewirkt wird.

Seien nun $\tilde{\mathfrak{M}}$ und $\overline{\mathfrak{M}}$ Maschinen, die $\tilde{f}$ bzw. $\overline{f}$ berechnen. Wir bilden $\tilde{\mathfrak{M}}_c$ und $\overline{\mathfrak{M}}_c$ und kombinieren diese Maschinen so, daß die neue Maschine $\overline{f} \circ \tilde{f}$ berechnet.

Die neue Maschine setzt zunächst $\tilde{\mathfrak{M}}_c$ in einen Anfangszustand, indem sie $\tilde{\zeta} := 1$ ausführt und $\tilde{r}$ und $\overline{r}$ je auf einen vorgegebenen Wert setzt. Die Maschine $\overline{\mathfrak{M}}_c$ setzt die neue Maschine in einen Endzustand, indem sie $\overline{\zeta} := 0$ ausführt und $\overline{r}$ den gleichen Wert zuweist wie $\tilde{r}$. Nun läuft $\tilde{\mathfrak{M}}_c$. Sobald $\tilde{\mathfrak{M}}_c$ in den Endzustand kommt, wird das Resultat $\tilde{f}_r(\zeta)$ von $\tilde{\mathfrak{M}}_c$ an $\overline{\mathfrak{M}}_c$ übergeben und $\overline{\mathfrak{M}}_c$ durch $\zeta := 1$ gestartet. Nach r Rechenschritten erreicht $\overline{\mathfrak{M}}_c$ einen Endzustand. Falls $\overline{\zeta} = \tilde{\zeta} = 0$ ist, wird die Berechnung beendet. Es ist nun $\overline{f}_r(\tilde{f}_r(\xi)) = f(\xi)$.

Ist die Bedingung $\tilde{\zeta} = \overline{\zeta} = 0$ nicht erfüllt, erhöht die neue Maschine in ihren Komponenten $\tilde{\mathfrak{M}}_c$ und $\overline{\mathfrak{M}}_c$ r um 1 und startet den eben beschriebenen Prozeß aufs Neue. Die Maschine hält für alle $\xi \in D$ nach endlicher Zeit an. Man erhält $f(\xi)$ durch eine geeignete Projektion des Speicherinhaltes.

Die eben skizzierte Maschine ist eine $\mathbb{P}$-Maschine, da die Abfragen $\tilde{\zeta} = 0$ und $\overline{\zeta} = 0$ durch die beschränkten Abfragen auf $\varrho = 0$ ersetzt werden.

(3) Die primitive Rekursion behandelt man ebenso, indem man sie nicht durch rekursive Prozeduraufrufe ausführt, sondern iterativ, d. h. der Reihe nach $f(\alpha, 0), f(\alpha, 1), \ldots, f(\alpha, x)$ berechnet. Es kommt im Vergleich zu (2) hinzu, daß man das Resultat von $\overline{\mathfrak{M}}$ an $\tilde{\mathfrak{M}}$ übergeben muß und diesen Zyklus $\xi(x)$-mal betreibt, wenn die primitive Rekursion über die Variable x läuft.

(4) Fall: μ-Operator.

Sei also $\mathfrak{M}$ eine $\mathbb{P}$-Maschine, die

$$f : \mathbb{N}^{k+1} \longrightarrow \mathbb{N}$$

berechnet. Wir konstruieren eine Maschine $\tilde{\mathfrak{M}}$, die $g := \mu f$ berechnet wie folgt.

Die Maschine $\tilde{\mathfrak{M}}$ verhält sich weitgehend wie $\mathfrak{M}$. Zum Start „rettet“sie nur $\xi(x_1, \ldots, x_k)$ in neue Variablen $(y_1, \ldots, y_k)$ und setzt $\xi(x_{k+1}) = 0$. Die Variable ζ von $\mathfrak{M}$ schlagen wir nun den Speichervariablen zu. D. h. wir definieren

$$\begin{aligned} \tilde{K}_1 &:= K_1 \times K_2 \times K_1'; \\ K_1' &:= \{\eta : \{y_1, \ldots, y_{k+1}\} \longrightarrow \mathbb{N}\}. \end{aligned}$$

Wir setzen

$$\tilde{K}_2 := \{0, 1, 2\}.$$

Im Anfangszustand $\tilde{\zeta} = 1$ setzt $\tilde{\mathfrak{M}}$ $\zeta := 1$ und $\eta(y_1, \ldots, y_k) := \eta(x_1, \ldots, x_k)$, $\eta(x_{k+1}) := \xi(x_{k+1}) := 0$. Danach geht $\tilde{\zeta}$ in den Zustand 2 über.

In diesem Zustand verhält sich $\tilde{\mathfrak{M}}$ wie $\mathfrak{M}$, solange wie $\zeta \neq 0$ ist. Ist $\zeta = 0$, $\xi(x_1) \neq 0$, dann setzt $\tilde{\mathfrak{M}}$

$$\begin{aligned}
\xi'(x_1, \ldots, x_k) &:= \eta(y_1, \ldots, y_k), \\
\xi(x_{k+1}) &:= \eta(y_{k+1}) + 1 \\
\eta'(y_{k+1}) &:= \eta(y_{k+1}) + 1, \\
\zeta &:= 1.
\end{aligned}$$

Diese „Runden" laufen solange, bis $\xi(x_1) = 0$ auftritt. Nun wird $\zeta := 0$ gesetzt. Gibt man $\eta(y_{k+1})$ aus, dann erhält man $g(\xi(x_1, \ldots, x_k))$.

$\tilde{\mathfrak{M}}$ ist wieder eine $\mathbb{P}$-Maschine. Also ist g durch eine $\mathbb{P}$-Maschine berechenbar.

Aus (1) bis (4) ergibt sich nun unser Satz.

Es bleibt die Behandlung von Berechnungen auf $\mathbb{N}^*$.

Wir betrachten nun **Maschinen über** $\mathbb{N}^*$.

Wir repräsentieren $\mathbb{N}^*$ durch

$$B := \{\varrho : \mathbb{N} \longrightarrow \mathbb{N} \mid \sum_{i=1}^{\infty} \varrho(i) < \infty\}.$$

Die Elemente von B können wir als Rechenbänder anschauen, die in ∞-viele Felder unterteilt sind; jedes Feld enthält als Inschrift eine natürliche Zahl. Alle bis auf endlich viele Felder sind „leer“.

Wir setzen nun

$$K_1 := B \quad \text{und} \quad K_2 := I := L := \mathbb{N}.$$

Wir lassen also auch zu, daß die Maschine über ∞-viele „interne“ Zustände verfügt. Es seien

$$\Delta_1 : \mathbb{N}^2 \longrightarrow \mathbb{N}^2$$

und

$$\Delta_2 : \mathbb{N}^2 \longrightarrow \{0, 1, -1\}$$

berechenbare Funktionen.

Wir setzen

$$\begin{aligned} K \;\; := \;\; & \{(\varrho, \zeta, i, k) \mid \varrho \in B, \zeta \in \mathbb{N}, i \in \mathbb{N}, k \in \mathbb{N}, i \leq k\} \\ & \text{und } \varrho(l) = 0 \text{ für } l > k \end{aligned}$$

Nun definieren wir

$$\Delta : B \times K_2 \times I \times L \longrightarrow B \times K_2 \times I \times L,$$

$$\begin{array}{lrcl} & (\varrho', \zeta', i', k') & = & \Delta(\varrho, \zeta, i, k) \\ :\Longleftrightarrow & \varrho'(l) & := & \begin{cases} \varrho(l) \text{ für } l \neq i, \\ p_1 \circ \Delta_1(\varrho(i), \zeta); \end{cases} \\ & \zeta' & := & p_2 \circ \Delta_1(\varrho(i), \zeta); \\ & i' & := & i + \Delta_2(\varrho(i), \zeta); \\ & k' & := & \begin{cases} k & \text{falls } i' \leq k \\ k+1 & \text{falls } i' := k+1. \end{cases} \end{array}$$

Erläuterung: i bezeichnet die aktuelle Position der Maschine auf dem Band. Δ_1 berechnet in Abhängigkeit von der Inschrift im i-ten Feld und dem aktuellen Maschinenzustand die neue Inschrift $p_1 \circ \Delta_1(\varrho(i), \zeta)$ und den Folgezustand $p_2 \circ \Delta_1(\varrho(i), \zeta)$.

p_i ist hierin die Projektion auf die i-te Komponente. Δ_2 gibt an, ob und wie sich die Maschine auf dem Band weiterbewegt: eine Position nach rechts $(+1)$, eine Position nach links (-1) oder die Maschine bleibt an ihrem Ort (0).

Die Funktion Δ ist im intuitiven Sinne berechenbar. Wir definieren als Menge der **Anfangskonfigurationen**

$$K_a := \{(\varrho, 1, i, k) \mid (\varrho, 1, i, k) \in K\}$$

und

$$K_e := \{(\varrho, 0, i, k) \mid (\varrho, 0, i, k) \in K\}$$

als Menge der **Endkonfigurationen**.

Definition: $\mathfrak{M} = (K, K_a, K_e, \Delta)$ heißt Maschine über $\mathbb{N}^*$. $\mathfrak{M}$ heißt $\mathbb{P}$-Maschine, falls Δ_1 und Δ_2 aus $\mathbb{P}$ sind.

Nun wollen wir zeigen, daß wir jede Berechnung auf $\mathfrak{M} = (K, K_a, K_e, \Delta)$ auf einer Maschine $\mathfrak{M}'$ mit $K_1' := \mathbb{N}^k$ simulieren können. Über k werden wir weiter unten verfügen.

Zunächst definieren wir, was wir unter einer **Simulation** verstehen. Das folgende Diagramm, in dem $\sigma : K \longrightarrow K'$ eine Abbildung ist, zeigt einen Maschinen-**Homomorphismus**.

$$\begin{array}{ccccc} & K & \xrightarrow{\Delta} & K & \\ \sigma & \downarrow & & \downarrow & \sigma \\ & K' & \xrightarrow{\Delta'} & K' & \end{array}$$

Im Falle der **Simulation** lassen wir verallgemeinernd zu, daß $\mathfrak{M}'$ mehrere Rechenschritte ausführt, wenn $\mathfrak{M}$ einen einzigen ausführt.

Definition: Sind $\mathfrak{M}$ und $\mathfrak{M}'$ Maschinen und ist $\sigma : K \longrightarrow K'$ eine Abbildung mit $\sigma(K_a) \subset K'_a$ und $\sigma(K_e) \subset K'_e$, die das Diagramm

$$\begin{array}{ccccc} & K & \xrightarrow{\Delta} & K & \\ \sigma & \downarrow & & \downarrow & \sigma \\ & K' & \xrightarrow{\Delta'^{(j)}} & K' & \end{array}$$

erfüllt, dann heißt σ **Simulation** von $\mathfrak{M}$ auf $\mathfrak{M}'$. Dabei mag j von dem Argument von Δ abhängen.

Definition: Die Simulation $\sigma : \mathfrak{M} \longrightarrow \tilde{\mathfrak{M}}$ heißt **treu** $:\Longleftrightarrow$ Die Einschränkungen $\sigma \mid K_a$ und $\sigma \mid K_e$ sind bijektiv.

Lemma: Ist $\sigma : \mathfrak{M} \longrightarrow \tilde{\mathfrak{M}}$ eine treue Simulation, und ist f $\mathfrak{M}$ berechenbar, dann läßt sich die Berechnung von f mittels σ auf $\tilde{\mathfrak{M}}$ übertragen.

Beweis: Sei, wie auf Seite 189, T bzw. $\tilde{T}$ die Laufzeitfunktion von $\mathfrak{M}$ bzw. $\tilde{\mathfrak{M}}$. Wir definieren wie dort

$$F(\xi, 0) := (\xi, 1),$$
$$F(\xi, t+1) := \Delta(F(\xi, t)).$$

Analog wird $\tilde{F}$ definiert. Es gilt also

$$F(\xi, T(\xi)) \in K_e, \quad \tilde{F}(\tilde{\xi}, T(\tilde{\xi})) \in \tilde{K}_a.$$

Da σ eine Simulation ist, gibt es $j = j(\xi, t)$, so daß

$$\sigma(F(\xi, t+1)) = \tilde{\Delta}^j(\sigma(F(\xi, t))).$$

Hieraus folgt wegen $\sigma(K_a) \subset \tilde{K}_a$, $\sigma(K_e) \subset \tilde{K}_e$:

$$\sigma(F(\xi, T(\xi))) = \tilde{F}(\sigma(\xi, 1), \tilde{T}(\sigma(\xi))).$$

Da σ bijektiv ist, gilt nun

$$F(\xi, T(\xi)) = \sigma^{-1}(\tilde{F}[\sigma(\xi, 1), \tilde{T}(\sigma(\xi))]).$$

Das heißt aber gerade, daß wir mittels σ die Berechnung von $\mathfrak{M}$ berechenbaren Funktionen auf Berechnungen in $\tilde{\mathfrak{M}}$ zurückführen können.

Satz: Jede $\mathbb{P}$-Maschine über $\mathbb{N}^*$ kann durch eine $\mathbb{P}$-Maschine über $\mathbb{N}^4$ treu simuliert werden.

Beweis: Wir wählen $k = 4$ und definieren

$$\sigma : K \longrightarrow \mathbb{N}^4$$

zunächst, indem wir

$$b_{k+1}(\varrho) := b_{k+1}(\varrho(0), \ldots, \varrho(k))$$

und

$$\sigma(\varrho, \zeta, i, k) := (b_{k+1}(\varrho), \zeta, i, k)$$

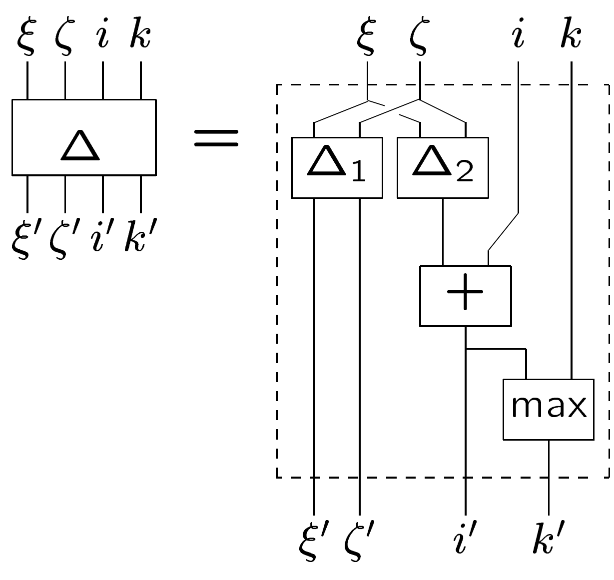
setzen. σ ist bijektiv, da k die Abbildung b_{k+1} eindeutig bestimmt und da b_k für $k = 1, 2, \ldots$ bijektiv ist.

Diese Definition von σ ist naheliegend aber vorläufig. Wir werden mit dieser Definition auf eine Schwierigkeit stoßen, die sich aber durch eine einfache Variation des Ansatzes beheben läßt.

Wir definieren $\tilde{\mathfrak{M}}$ passend zu σ. Sei also

$$(\tilde{\xi}, \zeta, i, k) \in \mathbb{N}^4.$$

Wir fassen die beiden Abbildungen Δ_1 und Δ_2 zu einer Abbildung Δ zusammen, wie es das folgende Netz definiert



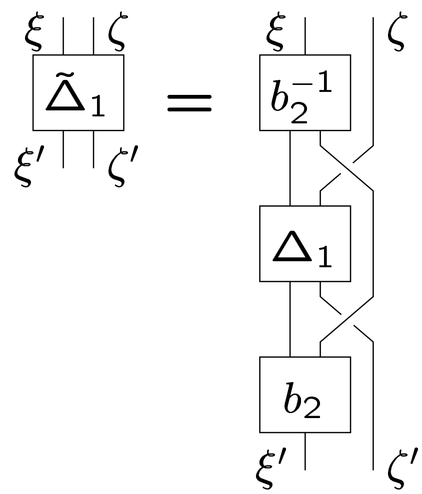
Nun definieren wir $\tilde{\Delta}$ für den Fall $i = 0$ und $k = k'$ durch das Diagramm

$$\begin{tikzcd}
(\tilde{\xi}, \zeta, i, k) \arrow[r, dashed, "\tilde{\Delta}"] \arrow[d, "\sigma^{-1}"'] & (b_{k+1}(\varrho'), \zeta', i', k) \\
(b_{k+1}^{-1}(\xi), \zeta, i, k) \arrow[r, mapsto, "\Delta"] & (\varrho', \zeta', i', k') \arrow[u, "\sigma"']
\end{tikzcd}$$

Da die Maschine auf Feld 0 steht, ergibt sich, wenn wir den ζ-Eingang von $\tilde{\Delta}_1$ und Δ_1 vernachlässigen

$$\begin{aligned}
\tilde{\Delta}_1 &= b_2 \circ (1 \times b_2) \circ \ldots \circ (1_{k-1} \times b_2) \circ (\Delta_1 \times 1_k) \\
&\quad \circ (1_{k-1} \times b_2^{-1}) \circ \ldots \circ (1 \times b_2^{-1}) \circ b_2^{-1} \\
&= b_2 \circ (\Delta_1 \times 1) \circ b_2^{-1}.
\end{aligned}$$

Wir übersetzen das Resultat nun mit Berücksichtigung des ζ-Eingangs in ein Schaltbild.



Offensichtlich liegt für diesen Fall $\tilde{\Delta}$ in $\mathbb{P}$.

Nun führen wir die Fälle $i < k$ auf diesen Sonderfall zurück, indem wir die Reduktionen

$$b_{k_j i} := b_2 \circ (b_i \times b_{k-i+1})$$

einführen. Offensichtlich gilt

$$b_{k,i} \in \mathbb{P} \quad \text{und} \quad b_{k,i}^{-1} \in \mathbb{P} \quad \text{für } i = 0, \ldots, k-1.$$

Für $i < k$ erhalten wir nun das Diagramm

$$\begin{array}{ccc} (\tilde{\xi}, \zeta, i, k) & \stackrel{\tilde{\Delta}}{\longrightarrow} & (b_{k,i}(\varrho'), \zeta', i', k') \\ \sigma^{-1}\downarrow & & \uparrow\sigma \\ (b_{k,i}^{-1}(\tilde{\xi}), \zeta, i, k) & \stackrel{\Delta}{\longrightarrow} & (\varrho', \xi', i', k'), \end{array}$$

das $\tilde{\Delta}$ und σ für $0 \leq i < k$ definiert. Setzen wir diese Transformationen hintereinander, dann ergibt sich, wenn wir vereinfachend den Eingang ζ vernachlässigen

$$\begin{aligned} \tilde{\Delta}_1 &= b_{k,i} \circ (1_i \times \Delta_1 \times 1_{k-i}) \circ b_{k,i}^{-1} \\ &= b_2 \circ (b_i \times b_{k-i+1}) \circ (1_i \times \Delta_1 \times 1_{k-i}) \\ &\quad \circ (b_i^{-1} \times b_{k-i+1}^{-1}) \circ b_2^{-1} \\ &= b_2 \circ (1 \times b_{k-i+1} \circ (\Delta_1 \times 1_{k-i}) \\ &\quad \circ (1 \times b_{k-i+1}) \circ b_2^{-1} \\ &= b_2 \circ (1 \times b_2 \circ (\Delta_1 \times 1) \circ b_2^{-1}) \circ b_2^{-1}. \end{aligned}$$

Die für den Fall $i < k$ definierte Abbildung $\tilde{\Delta}$ liegt offensichtlich in $\mathbb{P}$.

Wir betrachten nun den Fall $i = k$. Dieser Fall gliedert sich in zwei Unterfälle:

$$i = k,\ k' = k \text{ und } i = k,\ k' = k + 1.$$

Im zweiten Fall wird ein neues „Feld des Bandes“ in die Berechnung mit einbezogen.

Fall $i = k,\ k' = k.$

Wir setzen

$$b_{k,k} := b_2 \circ (b_k \times 1)$$

und erhalten in diesem Fall

$$\begin{aligned} \tilde{\Delta}_1 &= b_2 \circ (b_k \times 1) \circ (1_k \times \Delta_1) \circ (b_k^{-1} \times 1) \circ b_2^{-1} \\ &= b_2 \circ (1 \times \Delta_1) \circ b_2^{-1}. \end{aligned}$$

Also auch in diesem Fall liegt $\tilde{\Delta}$ in $\mathbb{P}$. Die Vervollständigung unter Einschluß des ζ-Eingangs ist symmetrisch zu dem Fall $i = 0$, der oben dargestellt wurde.

Fall $i = k, k' = k + 1.$

Hier erhalten wir

$$\begin{aligned}
\tilde{\Delta}_1 &= b_2 \circ (b_{k+1} \times 1) \circ (1_k \times \Delta_1 \times c_0) \\
&\quad \circ (b_k^{-1} \times 1) \circ b_2^{-1} \\
&= b_2 \circ [b_2 \circ (b_k \times 1) \times 1] \\
&\quad \circ (b_k^{-1} \times \Delta_1 \times c_0) \circ b_2^{-1} \\
&= b_2 \circ [b_2 \circ (b_k \circ b_k^{-1} \times 1) \times 1] \\
&\quad \circ (1 \times \Delta_1 \times c_0) \circ b_2^{-1} \\
&= b_2 \circ [b_2 \times 1] \circ (1 \times \Delta_1 \times c_0) \circ b_2^{-1}.
\end{aligned}$$

Auch diese Funktion liegt in $\mathbb{P}$.

Nun sind die drei betrachteten Fälle primitiv-rekursive Prädikate, so daß unsere Definition von $\tilde{\Delta}$ durch Fallunterscheidung schließlich eine primitiv-rekursive Abbildung $\tilde{\Delta}$ liefert.

Da die definierten Abbildungen σ eine treue Simulation von $\mathfrak{M}$ auf $\tilde{\mathfrak{M}}$ liefern, läßt sich jede $\mathfrak{M}$-berechenbare Funktion auch auf $\tilde{\mathfrak{M}}$ berechnen.

Aus dem Simulationssatz folgt der

Satz: Jede durch eine $\mathbb{P}$-Maschine über $\mathbb{N}^*$ berechenbare Abbildung $f : D \subset \mathbb{N}^k \longrightarrow D' \subset \mathbb{N}^j$ läßt sich auch durch eine $\mathbb{P}$-Maschine über $\mathbb{N}^4$ berechnen.

Beweis: Es bleibt nur zu ergänzen, daß man die Eingabe von $\xi \in D$ in $\mathfrak{M}$ durch die Standardreduktionen vornehmen kann. Das Gleiche gilt für die Ausgabe von $\mathbb{N}^4$ in D'. Die Berechnungen über $\mathbb{N}^4$ lassen sich durch die Standardreduktion auf Berechnungen über $\mathbb{N}$ reduzieren.

Zusammenfassung:

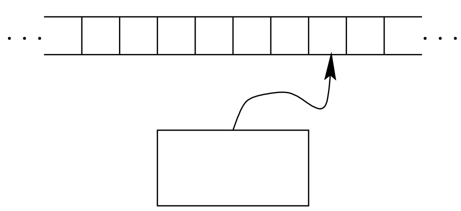
Wir haben abstrakte Maschinen $\mathfrak{M}$ über Mengen $\mathbb{N}^k$ und über $\mathbb{N}$ definiert. Wir haben gezeigt, daß jede über $\mathbb{N}^*$ durch eine $\mathbb{P}$-Maschine berechenbare Funktion auch durch eine $\mathbb{P}$-Maschine über $\mathbb{N}$ berechnet werden kann. Weiter haben wir gezeigt, daß alle die durch diese Maschinen berechenbaren Funktionen μ-rekursiv sind und daß sich jede μ-rekursive Funktion auf diese Weise berechnen läßt.

Die hier betrachteten Maschinen sind Verallgemeinerungen der Turingmaschinen. Somit gilt natürlich auch, daß die turingberechenbaren Funktionen alle μ-rekursiv sind. Da Turingmaschinen zur Allgemeinbildung der Informatiker gehören, werden wir im folgenden Abschnitt die Turingmaschinen einführen und beweisen, daß die μ-rekursiven Funktionen auch bereits auf Turingmaschinen berechenbar sind.

Der hier geschilderte Aufbau der Theorie ist meines Wissens neu. Der Beweis unseres Simulationssatzes ist einfacher und kürzer als der übliche Beweis für die μ-Rekursivität der turingberechenbaren Funktionen.

Turingmaschinen

Bildliche Darstellung



- unbegrenztes Band
- Schreib-/Lesekopf
- endliche Kontrolle

Definition: Eine (nichtdeterministische) *Turingmaschine* (TM) wird spezifiziert durch $\mathcal{M} = (Z, A, \delta, z_0)$, wobei

- Z endliche *Zustandsmenge*;
- $A \supset \{0, 1, \#, \sqcup\}$ *Alphabet* mit $Z \cap A = \emptyset$;
- $\delta \subset Z \times A \times Z \times A \times \{\mathtt{L}, \mathtt{N}, \mathtt{R}\}$ *Übergangsrelation*;
- $z_0 \in Z$ *Anfangszustand*.

$\mathcal{M}$ heißt *deterministisch*, falls $\delta : Z \times A \rightsquigarrow Z \times A \times \{\mathtt{L}, \mathtt{N}, \mathtt{R}\}$ eine (partielle) Funktion ist.

Konfigurationen

Mit $K_{\mathcal{M}} := A^* \cdot Z \cdot A^*$ bezeichnen wir die *Konfigurationen* von $\mathcal{M}$.

Wir definieren eine *Übergangsrelation* $\vdash$ auf Konfigurationen wie folgt, wobei $u, v \in A^*$, $a, b, a' \in A$ und $z, z' \in Z$:

$$ubzav \vdash \begin{cases} uz'ba'v & \text{falls } (z, a, z', a', \mathtt{L}) \in \delta; \\ ubz'a'v & \text{falls } (z, a, z', a', \mathtt{N}) \in \delta; \\ uba'z'v & \text{falls } (z, a, z', a', \mathtt{R}) \in \delta. \end{cases}$$

$$zav \vdash \begin{cases} z'\sqcup a'v & \text{falls } (z, a, z', a', \mathtt{L}) \in \delta; \\ z'a'v & \text{falls } (z, a, z', a', \mathtt{N}) \in \delta; \\ a'z'v & \text{falls } (z, a, z', a', \mathtt{R}) \in \delta. \end{cases}$$

$$ubz \vdash \begin{cases} uz'ba' & \text{falls } (z, \sqcup, z', a', \mathtt{L}) \in \delta; \\ ubz'a' & \text{falls } (z, \sqcup, z', a', \mathtt{N}) \in \delta; \\ uba'z' & \text{falls } (z, \sqcup, z', a', \mathtt{R}) \in \delta. \end{cases}$$

$$z \vdash \begin{cases} z'\sqcup a' & \text{falls } (z, \sqcup, z', a', \mathtt{L}) \in \delta; \\ z'a' & \text{falls } (z, \sqcup, z', a', \mathtt{N}) \in \delta; \\ a'z' & \text{falls } (z, \sqcup, z', a', \mathtt{R}) \in \delta. \end{cases}$$

Berechnungen

Eine Folge $(k_0, \dots, k_T) \in K_{\mathcal{M}}^*$ heißt eine *endliche Rechnung von* $\mathcal{M}$ *angesetzt auf* $w \in A^*$ ($\mathcal{M}$ angesetzt auf w *hält*), falls

i. $k_0 \in \sqcup^* z_0 w \sqcup^*$;

ii. $\forall\, 0 \leq i < T: \; k_i \vdash k_{i+1}$;

iii. $\nexists\, k \in K_{\mathcal{M}}: \; k_T \vdash k'$.

Eine Folge $(k_0, k_1, \dots) \in K_{\mathcal{M}}^{\omega}$ heißt eine *unendliche Rechnung von* $\mathcal{M}$ *angesetzt auf* $w \in A^*$ ($\mathcal{M}$ angesetzt auf w *hält nicht*), falls i. und ii. sinngemäß erfüllt sind. Ist $k_0 = z_0 w$, so heißen diese Rechnungen *kanonisch*.

Sei nun $\mathcal{M}$ *deterministisch*. $time(w) := T$ heißt die *Länge* der endlichen Rechnung. Für kanonische Rechnungen erklären wir den *Bandverbrauch* als

$$space(w) := \max\{|k_i| - 1\}.$$

Sei $f : \mathbb{N} \supset D \to \mathbb{N}$ und $w \in A^*$ beliebig mit $|w| \in D$. $\mathcal{M}$ heißt f*-zeitbeschränkt* bzw. f*-bandbeschränkt*, falls stets $time(w) \leq f(|w|)$ bzw. $space(w) \leq f(|w|)$.

Beispiel (binäres Zählen):

- rechtes Ende suchen (z_0);
- Eins addieren (z_1);
- linkes Ende suchen (z_2);
- fertig (z_3).

(z', a', r)	0	1	$\sqcup$
z_0	$(z_0, 0, \mathtt{R})$	$(z_0, 1, \mathtt{R})$	$(z_1, \sqcup, \mathtt{L})$
z_1	$(z_2, 1, \mathtt{L})$	$(z_1, 0, \mathtt{L})$	$(z_3, 1, \mathtt{N})$
z_2	$(z_2, 0, \mathtt{L})$	$(z_2, 1, \mathtt{L})$	$(z_3, \sqcup, \mathtt{R})$

Kanonische Rechnung angesetzt auf 101 ist

$$\begin{array}{ccccccc} z_0 101 & \vdash & 1 z_0 01 & \vdash & 10 z_0 1 & \vdash \\ 101 z_0 & \vdash & 10 z_1 1 \sqcup & \vdash & 1 z_1 00 \sqcup & \vdash \\ z_2 110 \sqcup & \vdash & z_2 \sqcup 110 \sqcup & \vdash & \sqcup z_3 110 \sqcup & \end{array}$$

mit Länge $time(101) = 8$ und Bandverbrauch $space(101) = 5$. $\mathcal{M}$ ist $(2n + 2)$-zeitbeschränkt und $(n + 2)$-bandbeschränkt.

Mehrband-Turingmaschinen

Definition: Sei $k \in \mathbb{N}$. Eine (nichtdeterministische) *k-Band-Turingmaschine* (k-TM) wird spezifiziert durch $\mathcal{M} = (Z, A, \delta, z_0)$ ähnlich wie zuvor, aber mit

$$\delta \subset Z \times A^k \times Z \times A^k \times \{\mathtt{L}, \mathtt{N}, \mathtt{R}\}^k$$

bzw. – im deterministischen Fall –

$$\delta : Z \times A^k \rightsquigarrow Z \times A^k \times \{\mathtt{L}, \mathtt{N}, \mathtt{R}\}^k.$$

Die Konfigurationsmenge ist

$$K_{\mathcal{M}} := \bigcup_{z \in Z} (A^* z A^*)^k.$$

Die Übergangsrelation $\vdash$ und Berechnungen ergeben sich analog. Den Bandverbrauch messen wir als Summe über die einzelnen Bänder.

Maschinenfunktion über $\mathcal{A}^*$

Sei $\mathcal{A} := A \setminus \{\sqcup\}$ und $h : A^* \to \mathcal{A}^*$ der Homomorphismus, der alle $\sqcup$ löscht.

Falls die 1-TM $\mathcal{M}$ angesetzt auf $w \in \mathcal{A}^*$ hält und $(k_0, \ldots, k_T)$ die zugehörige kanonische Rechnung ist mit $k_T = uzv$, so setzen wir $f_{\mathcal{M}}(w) := h(uv)$. Dies definiert in natürlicher Weise eine Funktion $f_{\mathcal{M}} : \mathcal{A}^* \rightsquigarrow \mathcal{A}^*$.

k-TMs $\mathcal{M}$ definieren eine ebensolche Funktion $f_{\mathcal{M}}$, wobei für Ein- und Ausgaben das erste Band benutzt wird.

Einseitiges Band

Lemma: Sei $t(n) \geq n$ und $\mathcal{M}$ sei eine $s(n)$-band- und $t(n)$-zeitbeschränkte 1-TM, deren Ausgabe keine Felder links von der Startposition der Köpfe belegt. Dann gibt es eine äquivalente $(s(n)+1)$-band- und $O(t(n))$-zeitbeschränkte TM $\mathcal{M}'$, die nie Felder links von der Startposition der Köpfe benutzt. Dies gilt analog für k-TM.

Beweis: Sei $\mathcal{M} = (Z, A, \delta, \zeta)$; numeriere Bandzellen über $\mathbb{Z}$, wobei der Kopf auf Position 0 startet. Sei $\diamond \notin A$ und $A' := A \cup A \times (A \cup \{\diamond\})$ sowie $\zeta_i \notin Z$ und $Z' := Z \times \{\top, \bot\} \cup \{\zeta_i \mid 0 \leq i \leq 3\}$.

Idee:

$\cdots\ \sqcup$	w_{-s}	$\cdots$	w_{-1}	w_0	w_1	$\cdots$	w_s	$\sqcup\ \cdots$

$$\Downarrow$$

w_0	w_1	$\cdots$	w_s	$\sqcup\ \cdots$
$\diamond$	w_{-1}	$\cdots$	w_{-s}	

Neuer Anfangszustand sei ζ_0; wir vereinbaren $\bar{\mathtt{L}} := \mathtt{R}$, $\bar{\mathtt{N}} := \mathtt{N}$ und $\bar{\mathtt{R}} := \mathtt{L}$. Die übergangsrelation δ' enthält für beliebige $x, y \in A$ sowie $(z, a, z', a', r) \in \delta$

- $(\zeta_0, x, \zeta_\top, \genfrac{}{}{0pt}{}{x}{\diamond}, N)$;
- $(z_\perp, \genfrac{}{}{0pt}{}{a}{\diamond}, z_\top, \genfrac{}{}{0pt}{}{a}{\diamond}, \mathtt{N})$; $\begin{cases} (z_\top, \genfrac{}{}{0pt}{}{a}{\diamond}, z'_\top, \genfrac{}{}{0pt}{}{a'}{\diamond}, r) & r \neq \mathtt{L}; \\ (z_\top, \genfrac{}{}{0pt}{}{a}{\diamond}, z'_\perp, \genfrac{}{}{0pt}{}{a'}{\diamond}, \mathtt{R}) & r = \mathtt{L}. \end{cases}$
- $(z_\top, \genfrac{}{}{0pt}{}{a}{x}, z'_\top, \genfrac{}{}{0pt}{}{a'}{x}, r)$; $(z_\perp, \genfrac{}{}{0pt}{}{x}{a}, z'_\perp, \genfrac{}{}{0pt}{}{x}{a'}, \bar{r})$;
- $(z_\top, a, z'_\top, \genfrac{}{}{0pt}{}{a'}{\sqcup}, r)$; $(z_\perp, x, z'_\perp, \genfrac{}{}{0pt}{}{x}{a'}, \bar{r})$ für $a = \sqcup$;
- $(\zeta_1, \genfrac{}{}{0pt}{}{x}{y}, \zeta_1, \genfrac{}{}{0pt}{}{x}{y}, \mathtt{R})$; $(\zeta_1, x, \zeta_2, x, \mathtt{L})$;
- $(\zeta_2, \genfrac{}{}{0pt}{}{x}{y}, \zeta_2, x, \mathtt{L})$; $(\zeta_2, \genfrac{}{}{0pt}{}{x}{\diamond}, \zeta_3, x, \mathtt{N})$;

und für alle $(z, a) \in Z \times A$ mit $\delta(z, a) = \emptyset$

- $(z_\top, \genfrac{}{}{0pt}{}{a}{x}, \zeta_1, \genfrac{}{}{0pt}{}{a}{x}, \mathtt{N})$; $(z_\top, a, \zeta_1, a, \mathtt{N})$;
- $(z_\perp, \genfrac{}{}{0pt}{}{x}{a}, \zeta_1, \genfrac{}{}{0pt}{}{x}{a}, \mathtt{N})$; $(z_\perp, x, \zeta_1, x, \mathtt{N})$ für $a = \sqcup$.

Einseitiges Band

Satz: Zu jeder k-TM $\mathcal{M}$ gibt es eine äquivalente k-TM $\mathcal{M}'$, die keine Felder links von der Startposition der Köpfe besucht. Ist $t(n) \geq n$ und $\mathcal{M}$ $s(n)$-band- und $t(n)$-zeitbeschränkt, so ist $\mathcal{M}'$ $O(s(n))$-band- und $O(t(n) + \delta_{k,1}(s(n))^2)$-zeitbeschränkt.

Beweis: Es gibt eine zu $\mathcal{M}$ äquivalente k-TM $\mathcal{M}''$, deren Ausgabe keine Felder links von der Startposition der Köpfe belegt.

Idee: Teile jedes Band in zwei Spuren, markiere die Startpositionen durch $\diamond$, simuliere $\mathcal{M}$ auf den oberen Spuren (analog zu vorhin, aber ohne Faltung). Verschiebe am Schluß die obere Spur nach rechts, damit keine Felder links von der Startposition der Köpfe belegt sind, und beseitige die Einteilung in Spuren.

Das Verschieben der Bandinschrift der Größe $s(n)$ dauert auf einer 1-TM quadratische und sonst lineare Zeit.

Normalform

Lemma: Sei $\mathcal{M}$ eine k-TM. Dann kann man o.B.d.A. annehmen, daß endliche Berechnungen stets in zwei Phasen verlaufen, so daß

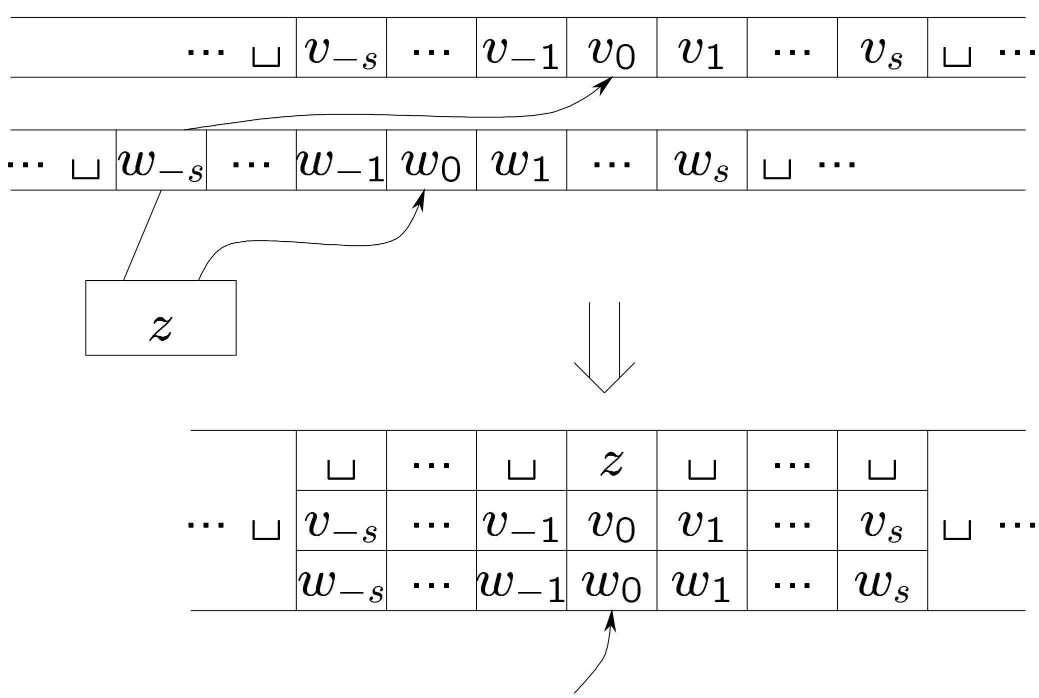
- $\mathcal{M}$ in der ersten Phase kein $\sqcup$ druckt;
- die Köpfe auf allen Bändern am Ende jeder Phase wieder am linken Rand der Bandinschrift stehen.

Beweis: Wenn $\mathcal{M}$ nicht diese Eigenschaften hat, dann führt man das neue Zeichen $\Box$ ein; dieses wird wie $\sqcup$ gelesen und anstelle von $\sqcup$ geschrieben. Sodann führt man neue Übergänge für alle $(z, a) \in Z \times A$ ein, für die $\delta(z, a)$ undefiniert bzw. leer ist. Diese führen zu einem neuen Programmteil, der zunächst alle Köpfe nach links fährt bis zu einem $\sqcup$; damit endet die erste Phase. Dann wird nach rechts das erste $\sqcup$ gesucht und auf dem Rückweg werden alle $\Box$ durch $\sqcup$ ersetzt; damit endet die zweite Phase.

Bandreduktion

Satz: Zu jeder $s(n)$-band- und $t(n)$-zeitbeschränkten k-TM $\mathcal{M}$ gibt es eine äquivalente 1-TM $\mathcal{M}'$, die $O(s(n))$-band- und $O(s(n) \cdot t(n))$-zeitbeschränkt ist.

Idee: Numeriere die Bandzellen über $\mathbb{Z}$ relativ zur *aktuellen* Kopfposition.



Maschinenfunktion über $\mathbb{N}$

$bin : \mathbb{N} \to \mathbb{B}^+$ bilde n ab auf die zugehörige Binärdarstellung ohne führende Nullen; bin ist injektiv mit Umkehrung

$$bin^{-1} : \mathbb{B}^+ \to \mathbb{N},\ w = w_1 \dots w_n \mapsto \sum_{i=1}^{n} w_i 2^{n-i}.$$

Für $k \in \mathbb{N}$ setzen wir bin fort zu $bin_k : \mathbb{N}^k \to \{0, 1, \#\}^*$ durch

$$bin_k(x_1, \dots, x_k) := bin(x_1)\# \dots \# bin(x_k).$$

Die Umkehrung $bin_k^{-1} : \{0, 1, \#\}^* \rightsquigarrow \mathbb{N}^k$ ist partiell, da sie nur für Wörter der Form $bin(x_1)\# \dots \# bin(x_k)$ definiert ist.

Analog zu $f_{\mathcal{M}}$ definiert man jetzt $k, l \in \mathbb{N}$ Funktionen $f_{\mathcal{M}}^{(k,l)} : \mathbb{N}^k \rightsquigarrow \mathbb{N}^l$ durch

$$f_{\mathcal{M}}^{(k,l)} := bin_l^{-1} \circ f_{\mathcal{M}} \circ bin_k.$$

Turingberechenbare Funktionen

Definition: Eine Funktion $f : \mathcal{A}^* \rightsquigarrow \mathcal{A}^*$ bzw. $f : \mathbb{N}^k \rightsquigarrow \mathbb{N}^l$ heißt *turingberechenbar*, falls es eine TM $\mathcal{M}$ gibt mit $f = f_{\mathcal{M}}$ bzw. $f = f_{\mathcal{M}}^{(k,l)}$.

Satz: Die Menge der turingberechenbaren Funktionen $\mathbb{N}^k \rightsquigarrow \mathbb{N}^l$ ist genau die Menge der μ-rekursiven Funktionen $\mathbb{N}^k \rightsquigarrow \mathbb{N}^l$.

Flußdiagramme

Seien die k-TM $\mathcal{M}_i = (Z_i, A_i, \delta_i, z_{0,i})$ $(1 \leq i \leq n)$ gegeben, wobei o.B.d.A. $i \neq j \Rightarrow Z_i \cap Z_j = \emptyset$. Wir wollen nun Flußdiagramme über diesen Maschinen erklären. Dazu bilden wir zunächst $\mathcal{M} := (Z, A, \delta, z_0)$ mit $Z := \bigcup_{1 \leq i \leq n} Z_i$, $A := \bigcup_{1 \leq i \leq n} A_i$ und $\delta := \bigcup_{1 \leq i \leq n} \delta_i$.

Ein Pfeil $\mathcal{M}_i \longrightarrow \mathcal{M}_j$ bedeutet nun, daß die „Endzustände" von $\mathcal{M}_i$ mit dem Anfangszustand von $\mathcal{M}_j$ verbunden werden, daß also Übergänge $(z, a, z_{0,j}, a, \mathtt{N})$ zu δ hinzugefügt werden für alle $(z, a) \in Z_i \times A_i$ mit $\delta_i(z, a) = \emptyset$. Ein markierter Pfeil $\mathcal{M}_i \xrightarrow{\zeta} \mathcal{M}_j$ schränkt dies auf den speziellen Zustand ζ ein. Dann werden nur Übergänge $(\zeta, a, z_{0,j}, a, \mathtt{N})$ zu δ hinzugefügt für alle $a \in A_i$ mit $\delta_i(\zeta, a) = \emptyset$.

Mit $\mathit{start} \longrightarrow \mathcal{M}_i$ drücken wir aus, daß $z_0 := z_{0,i}$ sei. Durch $\mathcal{M}_i \longrightarrow \mathit{stop}$ bzw. $\mathcal{M}_i \xrightarrow{\zeta} \mathit{stop}$ verdeutlichen wir die Endzustände der neuen Maschine.

Beispiel: Sei N die Maschine zur Berechnung der Nachfolgerfunktion und $Test :=$ $(\{z_0, z_1, ja, nein\}, A, \delta, z_0)$ gegeben durch

(z', a', r)	0	1	$\sqcup$
z_0	$(nein, 0, \mathtt{N})$	$(z_1, 1, \mathtt{R})$	$(nein, \sqcup, \mathtt{N})$
z_1	$(nein, 0, \mathtt{L})$	$(ja, 1, \mathtt{L})$	$(nein, \sqcup, \mathtt{L})$

Diese Maschine hält genau dann im Zustand ja, wenn die Bandinschrift mit 11 beginnt, sonst mit $nein$. Sie plaziert den Kopf stets wieder auf dem ersten Zeichen der Eingabe!

Dann bezeichnet das folgende Flußdiagramm eine Maschine, die so lange die Bandinschrift aus $\{0, 1\}^*$ um Eins erhöht, bis diese mit 11 beginnt.

$$\begin{tikzcd}
start \arrow[r] & N \arrow[r] & Test \arrow[r, "ja"] \arrow[l, bend left, "nein"] & stop
\end{tikzcd}$$

Gödelisierung von TM

Idee: Mache Beschreibungen von TM selbst zum Objekt von Berechnungen.

Sei $\mathcal{M} = (Z, A, \delta, z_0)$ eine det. 1-TM mit $Z = \{z_0, \ldots, z_n\}$ und $A = \{a_0, \ldots, a_m\}$; dabei sei $a_0 = 0$, $a_1 = 1$, $a_2 = \#$ und $a_3 = \sqcup$. Ordne nun jedem Tupel

$$(z_i, a_j, z_k, a_l, r) \in Z \times A \times Z \times A \times \{\mathtt{L}, \mathtt{N}, \mathtt{R}\}$$

seine Kodierung $w_{i,j,k,l,\hat{r}} :=$

$$\#\#bin(i)\#bin(j)\#bin(k)\#bin(l)\#bin(\hat{r})$$

über $\mathbb{A} := \{0, 1, \#\}$ zu, wobei

$$\hat{r} := \begin{cases} 0 & \text{falls } r = \mathtt{N}; \\ 1 & \text{falls } r = \mathtt{R}; \\ 2 & \text{falls } r = \mathtt{L}. \end{cases}$$

Jede Konkatenation $w \in \mathbb{A}^*$ all dieser Wörter beschreibt nun die TM $\mathcal{M}$ vollständig. Sei $g : \mathbb{A}^* \to \mathbb{B}^*$ der durch $g(0) := 00$, $g(1) := 01$ sowie $g(\#) := 11$ bestimmte Homomorphismus. Wir nennen $g(w) \in \mathbb{B}^*$ eine *Gödelisierung* von $\mathcal{M}$ und $bin^{-1}(g(w))$ eine *Gödelnummer* von $\mathcal{M}$.

Eigenschaften der Gödelisierung

Satz: Haben $\mathcal{M}$ und $\mathcal{M}'$ eine gemeinsame Gödelisierung, so hält $\mathcal{M}$ angesetzt auf $v \in \mathbb{A}^*$ genau dann, wenn $\mathcal{M}'$ dies tut. Ist $f_{\mathcal{M}}(v) \in \mathbb{A}^*$, so ist $f_{\mathcal{M}}(v) = f_{\mathcal{M}'}(v)$.

Beweis: $\mathcal{M}$ und $\mathcal{M}'$ unterscheiden sich nur in trivialer Weise, nämlich durch die Numerierung von $Z \setminus \{z_0\}$ und $A \setminus \{0, 1, \#, \sqcup\}$.

Lemma: Eine Zeichenreihe $u \in \mathbb{B}^*$ ist genau dann Gödelisierung einer det. 1-TM, falls

- $u = g(w)$ für ein $w \in \mathbb{A}^*$;
- w besteht aus Wörtern $w_{i,j,k,l,\hat{r}}$;
- kommen $w_{i,j,k,l,\hat{r}}$ und $w_{i',j',k',l',\hat{r}'}$ in w vor, so ist $(i, j) \neq (i', j')$.

„Umkehrung“ der Gödelisierung

Wir ordnen nun einem Wort $u \in \mathbb{B}^*$ eine 1-TM $\mathcal{M}_u$ zu: Ist u nicht Gödelisierung einer 1-TM, so sei $\mathcal{M}_u$ eine Maschine, die für keine Eingabe hält. Andernfalls sei $w := g^{-1}(u)$ und $\mathcal{M}_u := (Z, A, \delta, z_0)$, $Z := \{z_0, \ldots, z_n\}$, $A := \{a_0, \ldots, a_m\}$. δ sei die Menge aller Tupel (z_i, a_j, z_k, a_l, r), deren Kodierung $w_{i,j,k,l,\widehat{r}}$ in w vorkommt, und n bzw. m seien die maximale Nummer eines Zustands bzw. Zeichens in diesen Tupeln.

Ist u Gödelisierung irgendeiner 1-TM, so auch von $\mathcal{M}_u$.

Korollar: Ist u Gödelisierung von $\mathcal{M}$, so hält $\mathcal{M}$ angesetzt auf $v \in \mathbb{A}^*$ genau dann, wenn $\mathcal{M}_u$ dies tut. Ist $f_{\mathcal{M}}(v) \in \mathbb{A}^*$, so ist $f_{\mathcal{M}}(v) = f_{\mathcal{M}_u}(v)$.

Universelle 1-TM

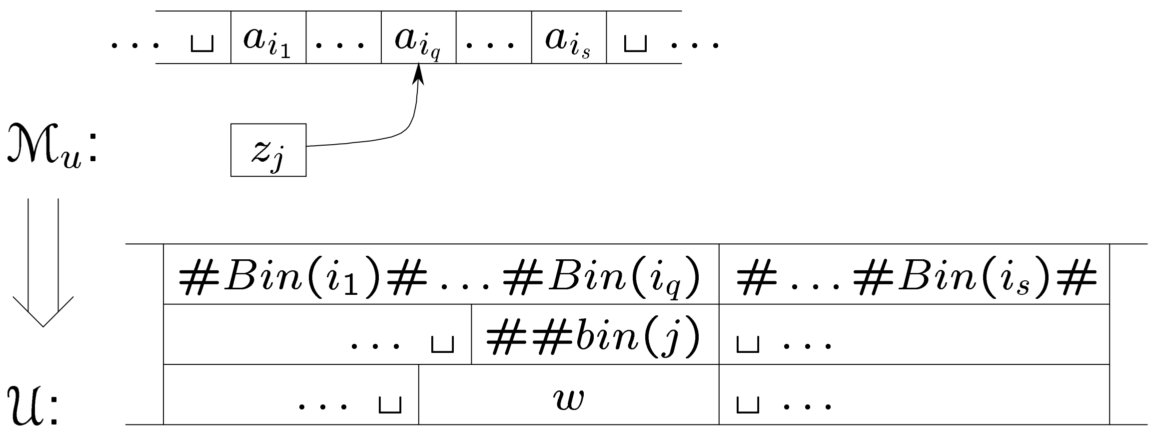
Satz: Es gibt eine 1-TM $\mathcal{U}$, so daß für $u \in \mathbb{B}^*$ und $v \in \mathbb{A}^*$ gilt:

- $\mathcal{U}$ angesetzt auf $u\#v$ hält genau dann, wenn $\mathcal{M}_u$ angesetzt auf v hält. Ist $f_{\mathcal{M}_u}(v) \in \mathbb{A}^*$, so ist $f_{\mathcal{U}}(u\#v) = f_{\mathcal{M}_u}(v)$.
- Benutzt $\mathcal{M}_u$ angesetzt auf v s Felder und macht t Schritte, so benutzt $\mathcal{U}$ angesetzt auf $u\#v$ $O(|u| \cdot s)$ Felder und macht $O(|u|^2(|v|^2 + t + l^2))$ Schritte, wobei $l := |f_{\mathcal{M}_u}(v)|$.

Idee: $\mathcal{U}$ angesetzt auf $u\#v$ testet zunächst, ob u Gödelisierung einer TM ist. Falls ja, simuliert $\mathcal{U}$ die Maschine $\mathcal{M}_u$ angesetzt auf v Schritt für Schritt; ansonsten hält $\mathcal{U}$ nicht.

Beweis: Sei $\mathcal{M}_u = (Z, A, \delta, z_0)$, $A = \{a_0, \ldots, a_m\}$ und $w := g^{-1}(u)$. $\mathcal{U}$ hat ein festes Alphabet und muß daher a_i durch $Bin(i)$ kodieren, die Binärdarstellung von i der Länge $\lfloor \log m \rfloor + 1$.

Die Bandbelegung ist wie folgt:



Simulation eines Schrittes:

- Hänge $\#bin(i_q)\#$ an Spur 2 an.
- Suche $\#\#bin(j)\#bin(i_q)\#$ auf Spur 3 (enthält w ein Wort $w_{j,i_q,j',i_{q'},\hat{r}}$?) und ersetze die Inschrift von Spur 2 durch $\#\#bin(j')\#bin(i_{q'})\#bin(\hat{r})\#$.
- Verschiebe Spur 2, bis $bin(j')$ und w rechtsbündig sind.
- Überschreibe $Bin(i_q)$ auf Spur 1 durch $Bin(i_{q'})$.
- Merke r und lösche $\#bin(i_{q'})\#bin(\hat{r})\#$ auf Spur 2. Falls $r \neq \mathtt{N}$, müssen Spur 2 und 3 entsprechend verschoben werden.

Analyse: $\mathcal{U}$ benutzt höchstens $|u| + s \cdot (\lfloor \log m \rfloor + 2) + 3$ Felder. Wegen $\lfloor \log m \rfloor < |u|$ folgt die Behauptung: $O(|u| \cdot s)$ Felder.

Für den Test, ob u Gödelisierung einer TM ist, genügt Zeit $O(|u|^2)$ (Übung). Für das Präparieren der Bandbelegung (insbesondere $a_i \mapsto Bin(i)$) genügt Zeit $O((|u| \cdot |v|)^2)$.

Die meiste Zeit bei der Simulation eines Schrittes erfordert das Verschieben von Spur 2 und 3; deren Inschriften sind maximal $|u|$ lang und werden um $O(|u|)$ Felder verschoben: Zeit $O(|u|^2)$.

Das Rückgewinnen des Ergebnisses auf Spur 1 dauert analog zum Präparieren $O((|u| \cdot l)^2)$. Insgesamt: $O(|u|^2(|v|^2 + t + l^2))$.

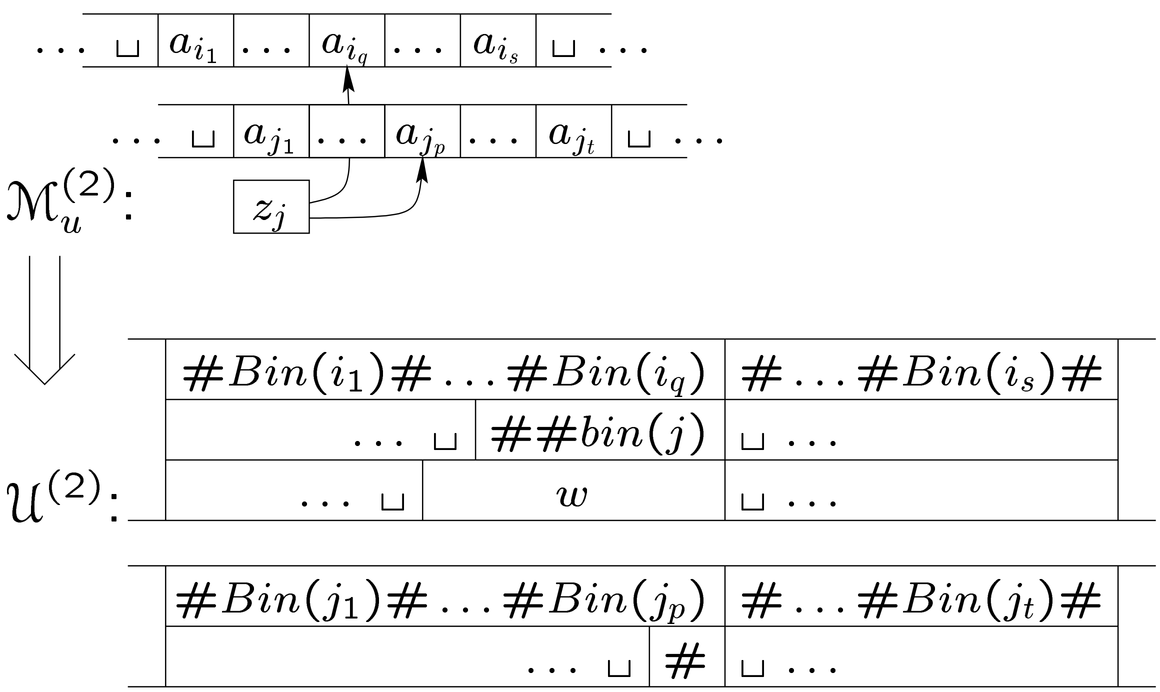
Universelle k-TM

Verallgemeinere Gödelisierungen auf k-TM für $k > 1$ und bezeichne die Umkehrung mit $\mathcal{M}_u^{(k)}$. Dann gilt der

Satz: Es gibt eine k-TM $\mathcal{U}^{(k)}$, so daß für $u \in \mathbb{B}^*$ und $v \in \mathbb{A}^*$ gilt:

- $\mathcal{U}^{(k)}$ angesetzt auf $u\#v$ hält genau dann, wenn $\mathcal{M}_u^{(k)}$ angesetzt auf v hält. Ist $f_{\mathcal{M}_u^{(k)}}(v) \in \mathbb{A}^*$, so ist $f_{\mathcal{U}^{(k)}}(u\#v) = f_{\mathcal{M}_u^{(k)}}(v)$.
- Benutzt $\mathcal{M}_u^{(k)}$ angesetzt auf v s Felder und macht $t \geq |v|$ Schritte, so benutzt $\mathcal{U}^{(k)}$ angesetzt auf $u\#v$ $O(|u| \cdot s)$ Felder und macht $O(|u|^2 \cdot t)$ Schritte.

Idee: Analog zum Fall $k = 1$, hier illustriert für $k = 2$:



Analyse: Mit Hilfe von 2 Bändern kann die Eingabe schon in Zeit $O(|u| \cdot |v|)$ präpariert und die Ausgabe in Zeit $O(|u| \cdot l)$ zurückgewonnen werden. Wegen $|v| \leq t$ und $l \leq |v| + t$ folgt: $O(|u|^2 \cdot t)$.

Entscheidbarkeit und Aufzählbarkeit

Definition: Eine Teilmenge $L \subset M$ (hier: $M = A^*$ oder $M = \mathbb{N}^k$) heißt genau dann *entscheidbar*, wenn ihre *charakteristische Funktion*

$$\chi_L : M \to \mathbb{B},\ x \mapsto \begin{cases} 1 & \text{falls } x \in L; \\ 0 & \text{sonst.} \end{cases}$$

berechenbar ist.

L heißt *rekursiv aufzählbar*, falls $L = \emptyset$ oder

$\exists\, f : M \to M$ berechenbar mit $f(M) = L$.

Sprechweise: f zählt L auf. Äquivalent: berechenbare Surjektion $\mathbb{N} \to L$.

Satz: L ist genau dann entscheidbar, wenn L und $\overline{L} := M \setminus L$ rekursiv aufzählbar sind.

Beweis: Übung.

Halteproblem

Definition: Die Menge

$$H_d := \{u \in \mathbb{B}^* \mid \mathcal{M}_u \text{ angesetzt auf } u \text{ hält}\}$$

heißt das *spezielle Halteproblem*.

Satz: Das spezielle Halteproblem ist unentscheidbar, d.h. $\chi_{H_d} : \mathbb{B}^* \to \mathbb{B}$ ist keine turingberechenbare Funktion.

Beweis: Durch Diagonalisierung. Angenommen, $\mathcal{M}$ sei eine TM mit

$$f_{\mathcal{M}}(u) = \begin{cases} 1 & \text{falls } \mathcal{M}_u \text{ angesetzt auf } u \text{ hält;} \\ 0 & \text{sonst.} \end{cases}$$

Daraus erhält man leicht eine Maschine $\mathcal{M}'$ mit

$$f_{\mathcal{M}'}(u) = \begin{cases} 1 & \text{falls } f_{\mathcal{M}}(u) = 0; \\ \text{undefiniert} & \text{sonst.} \end{cases}$$

[Fortsetzung des Beweis] Sei nun u' eine Gödelisierung von $\mathcal{M}'$:

$$\begin{array}{ll} & \mathcal{M}' \text{ angesetzt auf } u' \text{ hält} \\ \iff & f_{\mathcal{M}}(u') = 0 \\ \iff & \mathcal{M}_{u'} \text{ angesetzt auf } u' \text{ hält nicht} \\ \iff & \mathcal{M}' \text{ angesetzt auf } u' \text{ hält nicht.} \end{array}$$

Korollar: Auch die Menge

$$H := \{u\#v \mid u \in \mathbb{B}^*,\ v \in \mathbb{A}^* \\ \mathcal{M}_u \text{ angesetzt auf } v \text{ hält}\},$$

das (allgemeine) *Halteproblem*, ist unentscheidbar.

Beweis: Die Abbildung $d : \mathbb{B}^* \to \mathbb{A}^*$ mit $d(u) := u\#u$ ist turingberechenbar und $\chi_{H_d} = \chi_H \circ d$.

Reduzierbarkeit

Definition: Sind $L, L' \subset A^*$ und es gibt eine berechenbare Funktion $f : A^* \to A^*$ mit

$$\forall\, w \in A^* : \;\; w \in L \iff f(w) \in L',$$

dann heißt L vermittels f *reduzierbar* auf L'. Schreibweise: $L \leq L'$.

Satz: • Die Relation $\leq$ ist transitiv.

• Ist $L \leq L'$ und L unentscheidbar, so ist auch L' unentscheidbar.

Beispiel: $H_d \leq H$ vermittels d. Auch

$$H_0 := \{u \in \mathbb{B}^* \mid \mathcal{M}_u \text{ angesetzt auf } \varepsilon \text{ hält}\}$$

ist unentscheidbar, wie man durch Reduktion von H auf H_0 zeigt (Übung).

Übrigens: H_0, H_d, H sind rekursiv aufzählbar!

Satz von Rice

Sei $\mathcal{T}$ die Menge aller turingberechenbaren Funktionen und $L \subset \mathcal{T}$ beliebig. Dann ist $\hat{L} := \{u \mid f_{\mathcal{M}_u} \in L\}$ genau dann entscheidbar, wenn $L = \emptyset$ oder $L = \mathcal{T}$.

Beweis: Sei $L \neq \emptyset$ und $L \neq \mathcal{T}$ sowie ω die überall undefinierte Funktion.

$\omega \in L$: Sei $f \in \overline{L}$ mit TM $\mathcal{M}$. $w \in \mathbb{B}^*$ wird eine TM $\mathcal{M}(w)$ zugeordnet: Angesetzt auf $v \in \mathbb{B}^*$ ignoriert $\mathcal{M}(w)$ zunächst die Eingabe und verhält sich wie $\mathcal{M}_w$ angesetzt auf ε. Hält $\mathcal{M}_w$, so verhält sich $\mathcal{M}(w)$ anschließend wie $\mathcal{M}$ angesetzt auf v; dann ist $f_{\mathcal{M}(w)} = f \notin L$ und sonst ist $f_{\mathcal{M}(w)} = \omega \in L$.
Man kann aus w effektiv eine Gödelisierung von $\mathcal{M}(w)$ berechnen, daher ist $\overline{H_0} \leq \hat{L}$.

$\omega \notin L$: Analog: $H_0 \leq \hat{L}$.

Sei $\mathfrak{C} = (O, M, Q, Z, 0)$ eine Kategorie von Abbildungen:

$$Q : M \longrightarrow O,\ \ Z : M \longrightarrow O.$$

M ist die Menge der Abbildungen oder Morphismen und O die Menge der „Objekte“ der Kategorie. Ist $f \in M$, dann ist $f : Q(f) \longrightarrow z(f)$ eine Abbildung.

Beispiel:

(1) $\begin{array}{rcl} O & := & \{\mathbb{N}^k \mid k = 0, 1, 2, \ldots\} \\ M & = & \mathbb{P}. \end{array}$

(2) $\begin{array}{rcl} O & = & \mathbb{N}^*, \\ M & = & \{f\ \mathbb{P}\text{-berechenbar über } \mathbb{N} \mid f \text{ total}\}. \end{array}$

(3) $\begin{array}{rcl} O & = & \mathbb{N}, \\ M & = & \{f \in \mathbb{P} \mid Z(f) = Q(f) = \mathbb{N}\}. \end{array}$

(4) $\begin{array}{rcl} O & = & \mathbb{B}^*, \\ M & = & \{f : \mathbb{B}^* \to \mathbb{B}^* \mid f \text{ turingberechenbar} \\ & & \ \& \ f \text{ total}\}. \end{array}$

(5) $\begin{array}{rcl} O & = & \mathbb{B}^*, \\ M & = & \{f : \mathbb{B}^* \to \mathbb{B}^* \mid f \text{ turingberechenbar} \\ & & \text{in Zeit } O(n^k) \text{ für ein geeignetes } k\}. \end{array}$

Wir setzen $Pol := (O, M, Q, Z, 0)$. Pol ist die Kategorie der in Polynomialzeit berechenbaren Funktionen.

Definition: Sind $D_i \subset O_i \in O(\mathfrak{C})$ und gibt es $f : O_1 \longrightarrow O_2$ mit $L_1 = f^{-1}(L_2)$, dann schreiben wir

$$L_1 \underset{\mathfrak{C}}{\leq} L_2.$$

Sprechweise: L_1 ist mittels $\mathfrak{C}$ auf L_2 reduzierbar. Genauer:

Das „Wortproblem“: gilt für $w \in O_1$
$w \in L_1$?

ist $\mathfrak{C}$ **reduzierbar** auf $f(w) \in L_2$? durch ein geeignetes $f \in \mathfrak{C}$.

Die Sprechweise wird verständlicher durch das

Lemma: $L_1 \underset{\mathfrak{C}}{\leq} L_2 \iff$ Es gibt ein $f \in M(\mathfrak{C})$ mit (1) und (2).

(1) $L_1 \subset Q(f),\ L_2 \subset Z(f)$

(2) $w \in L_1 \iff f(w) \in L_2.$

Lemma: $\underset{\mathfrak{C}}{\leq}$ ist idempotent und transitiv.

Beweis: Sei $L_1 \underset{\mathfrak{C}}{\leq} L_2$, $L_2 \underset{\mathfrak{C}}{\leq} L_3$,
$\Longrightarrow \exists f_1$ und f_2 in $\mathfrak{C}$ mit

$$L_1 = f_1^{-1}(L_2) \text{ und } L_2 = f_2^{-1}(L_3).$$

Hieraus folgt

$$L_1 = f_1^{-1}(f_2^{-1}(L_3)) = (f_2 \circ f_1)^{-1}(L_3).$$

Wegen $f_2 \circ f_1 \in M(\mathfrak{C})$ ist $L_1 \underset{\mathfrak{C}}{\leq} L_3$.

Damit ist der zweite Teil des Lemmas bewiesen. Der erste Teil ergibt sich, da 1_{0_1} definitionsgemäß in $\mathfrak{C}$ liegt.

Definition:

$$L_1 \underset{\mathfrak{C}}{\equiv} L_2 :\Longleftrightarrow L_1 \underset{\mathfrak{C}}{\leq} L_2 \;\&\; L_2 \underset{\mathfrak{C}}{\leq} L_1.$$

Offensichtlich ist $\equiv_{\mathfrak{C}}$ eine Äquivalenzrelation.

Satz: Ist $\mathfrak{C} = (M, O)$ mit $O = \{\mathbb{N}^k \mid k = 0, 1, 2, \ldots\}$ und $M \subset \mu\mathbb{P}$, dann folgt aus $L' \underset{L}{\leq}$ und $\chi_{L'} \notin \mu\mathbb{P}$, daß $\chi_L \notin \mu\mathbb{P}$ gilt.

Beweis: Aus $\chi_L \in \mu\mathbb{P}$ und $f \in \mathfrak{C}$ folgt $\chi_{L'} = \chi_L \circ f \in \mu\mathbb{P}$. Widerspruch zur Voraussetzung.

Wir werden diesen Satz verwenden um für eine Reihe von Problemen zu zeigen, daß sie nicht entscheidbar sind.

Erläuterung des Konzeptes:

Die Reduktionstheorie hat sich aus der Geometrie entwickelt. Die Kategorie $\mathfrak{C}$ ist dort eine Gruppe von Bewegungen.

Sei $G = \mathfrak{C}$ die Gruppe der Bewegungen im euklidischen Raum und $L_1 = \{P_1, P_2, P_3\}, L_1' = \{P_1', P_2', P_3'\}$ seien Punktetripel aus dem $\mathbb{R}^3$. Es gilt dann

$$L_1 \leq L_2 \iff \text{Es gibt } f \in G \text{ mit } L_1 = f^{-1}(L_2).$$

Da G eine Gruppe ist, gilt auch $f^{-1} \in G$, so daß wegen $L_2 = (f^{-1})^{-1}(L_1) = f(L_1)$ auch $L_2 \underset{G}{\leq} L_2$.

In anderen Worten: Aus $L_2 \underset{G}{\leq} L_1$ folgt, daß die beiden Dreiecke P_1, P_2, P_3 und $P_1' P_2' P_3'$ kongruent sind.

Aus der Schule kennt man Kongruenzsätze, wie

> Zwei Dreiecke sind genau dann kongruent, wenn sie in drei Seiten übereinstimmen.

Von kongruenten Dreiecken weiß man z. B., daß sie den gleichen **Flächeninhalt** haben. In der Reduktionstheorie versucht man, Probleme hinsichtlich ihrer **Komplexität** zu klassifizieren.

Man sucht Invarianten unter Reduktionen, wie es z.B. der Flächeninhalt für Bewegungen darstellt, um die Inäquivalenz von Sprachen relativ zu Kategorien $\mathfrak{C}$ zu zeigen.

Die gröbste Klassifikation liefert die Kategorie $\mathfrak{C}\mu$ der totalen μ-rekursiven Funktionen. man kann mit den in der Vorlesung vorgestellten Konstruktionen Mengen L definieren, so daß weder L und $\overline{L}$ aufzählbar sind. Die Kategorie erlaubt eine Klassifikation dieser Mengen. Für die Informatik interessanter sind Klassifikationen, die sich durch kleinere Reduktionskategorien ergeben. Am bekanntesten ist hier die oben definierte Kategorie Pol.

Um das mit dieser Kategorie verknüpfte berühmte Problem $\underline{P} = \underline{NP}$? zu verstehen, bedarf es noch einer Ergänzung.

Sei $\mathfrak{C}$ eine Reduktionskategorie und

$$\mathfrak{T} = \{L_1, L_2, \ldots\}$$

eine Menge von Sprachen aus den Objekten von $\mathfrak{C}$.

Definition: Ist $L \subset O_i$ und $O_i \in O(\mathfrak{C})$, dann heißt L $\mathfrak{T}$-hart bezüglich $\mathfrak{C} :\Longleftrightarrow \mathfrak{T} \subset \{f^{-1}(L) \mid f \in M(\mathfrak{C})\}$.

Erläuterung: Die Anwendung der auf den Teilmengen der Objekte von $\mathfrak{C}$ definierten Abbildungen f^{-1} mit $f \in M(\mathfrak{C})$ erzeugt ganz $\mathfrak{T}$ und eventuell noch mehr. Wir schreiben dafür $\mathfrak{T} \subset \mathfrak{C}^{-1}(L)$.

Definition: L heißt **$\mathfrak{F}$-vollständig** (complete) bezüglich $\mathfrak{C}$, falls $L \in \mathfrak{T}$ gilt und L $\mathfrak{T}$-hart ist.

In den interessanten Fällen gilt dann auch $\mathfrak{C}^{-1}(L) = \mathfrak{T}$.

Wir wählen nun $\mathfrak{C} = Pol$ und

$\mathfrak{T} := \underline{NP}$:= $\{L \mid L$ läßt sich durch eine nichtdeterministische Turingmaschine in polynomieller Zeit entscheiden.$\}$

In diesem Fall gilt offensichtlich:

Ist $L \in \underline{NP}$, dann gilt $Pol^{-1}(L) \subset \underline{NP}$.

L ist also NP-vollständig genau dann, wenn $Pol^{-1}(L) = NP$ gilt.

Das Problem

$$\underline{P} = \underline{NP}?$$

ist die Frage, ob die Menge der in polynomieller Zeit entscheidbarer Probleme $\underline{P}$ gleich der Menge der **nicht**deterministisch in polynomieller Zeit entscheidbarer Probleme ist. Das ist ein schönes Problem.

Die Beantwortung dieser Frage hat für sich allein genommen wohl keine praktische Bedeutung. Die Frage ist aber wichtig, da sie zeigt, daß wir ein fundamentales Problem, nämlich ob richtig Raten „hilft", noch nicht verstanden haben. Die Arbeiten zur Klärung dieser Frage berühren damit auch Probleme großer praktischer Relevanz. Aus diesem Grund ist es zu erwarten, daß bei der Suche nach einer Lösung dieses Problems Entdeckungen gemacht werden, die auch von großer praktischer Bedeutung sind.

Diese und verwandte Fragestellungen werden in der Vorlesung „Komplexitätstheorie“ ausführlich behandelt. Hier müssen wir uns mit dieser Skizze begnügen. Es sei nur noch hinzugefügt, daß die Frage, ob ein vorgegebener Schaltkreis der Größe n auf irgendeine mögliche Eingabe mit einer Ausgabe $= 1$ reagiert, ein NP-vollständiges Problem ist, wenn n als Maß für die Problemgröße dient. Das Testproblem für Schaltkreise ist also ein NP-vollständiges Problem.

Semi-Thue-Systeme

Sei A ein Alphabet und $P \subset A^* \times A^*$ eine endliche Menge. Wir betrachten nun Ableitungen von Wörtern, indem wir Substitutionen von Teilwörtern $u_1 \longrightarrow v_1$ zulassen, falls $(u_1, v_1) \in P$ ist. Wir definieren im folgenden die Relation $(u \xrightarrow{P})$ für $(u, v) \in A^* \times A^*$.

$$
\begin{aligned}
u &\xrightarrow{P} u \text{ für } u \in A^*,\\
w_1 u w_2 &\xrightarrow{P} w_1 v w_2 \text{ für } (u,v) \in P,\\
w_1 \xrightarrow{P} w_2,\ w_2 &\xrightarrow{P} w_3 \Rightarrow w_1 \xrightarrow{P} w_3.
\end{aligned}
$$

Definition: Das Tupel $\mathfrak{S} := (A^*, P, \rightarrow)$ heißt **Semi-Thue-System**.

Thue war ein norwegischer Mathematiker, der sich mit dem Wortproblem von Gruppen auseinandergesetzt hat. Ein $\mathfrak{S}$ heißt **Thue-System** $:\Longleftrightarrow$ Aus $(u, v) \in P$ folgt $(v, u) \in P$.

Definition: Das Problem : „Gilt für $u, v \in A^*$ die Relation $u \xrightarrow{P} v$?“ heißt das Wortproblem für Semi-Thue-Systeme.

Gesucht ist also ein Algorithmus, der für gegebene P, u, v entscheidet, ob $u \xrightarrow{P} v$ gilt.

Satz: Das Wortproblem für Semi-Thue-Systeme ist nicht generell entscheidbar.

Beweis: Zum Beweis reduzieren wir das Halteproblem von Turingmaschinen auf das Wortproblem für Semi-Thue-Systeme. Hieraus folgt, wie wir gezeigt haben, die Nichtentscheidbarkeit des Wortproblems für Semi-Thue-Systeme.

Als Alphabet A wählen wir die Vereinigung des Bandalphabetes der TM mit dem Zustandsalphabet. P ist durch die Turingtafel gegeben.

Die Turingtafel hat die Form

$$\begin{array}{l} aqb \longrightarrow \left\{ \begin{array}{llll} a & q_1 & b' & \\ c & q_1 & a & b' \\ a & b' & q_1 & c \end{array} \right. \\ \vdots \end{array}$$

Hierin tritt bei deterministischen TM höchstens eine der drei möglichen Formen auf.

Wir wählen

$w = q_a u$ als Anfangskonfiguration

und ergänzen die Turingtafel durch Terminalproduktionen

$$q_e a \longrightarrow q_e.$$

Das Halteproblem der TM reduziert sich auf das Problem

$$q_a u \longrightarrow q_e.$$

Man kann den Satz noch verschärfen zu dem

Satz: Es gibt ein spezielles Semi-Thue-System für das das Wortproblem nicht entscheidbar ist.

Beweis: Wie vorher: Man geht dabei von einer universellen TM aus, für die, wie wir gezeigt haben, das Halteproblem nicht entscheidbar ist.

Satz: Das Wortproblem für Thue-Systeme ist nicht generell entscheidbar.

Beweis: Sei $\mathfrak{S}$ ein deterministisches Semi-Thue-System, wie wir es aus einer Turingtafel erhalten, und $\mathfrak{T}$ gehe aus $\mathfrak{S}$ durch Symmetrischmachen hervor.

Wir zeigen

$$u \xrightarrow{\mathfrak{T}} v \iff u \xrightarrow{\mathfrak{S}} v \text{ oder } v \xrightarrow{\mathfrak{S}} u.$$

Sei $w_0 = q_0 w \in K_a$ und $w_P = q_e$ Endzustand der TM von der wir ausgehen. Wir betrachten eine reguläre oder kanonische Berechnung

$$w_0 \xrightarrow{\mathfrak{S}} w_1 \xrightarrow{\mathfrak{S}^{-1}} w_2 \xrightarrow{\mathfrak{S}} \ldots \xrightarrow{\mathfrak{S}} w_P.$$

Die Ableitung beginnt mit $\mathfrak{S}$, wenn wir die TM so normieren, daß ein Anfangszustand nicht mehr angenommen wird. Sie endet mit $\mathfrak{S}$, da der Endzustand in $\mathfrak{S}^{-1}$ nicht auf der rechten Seite vorkommt.

Nun betrachten wir die Situation

$$w' \xrightarrow{\mathfrak{S}^{-1}} w'' \xrightarrow{\mathfrak{S}} w'''.$$

Da die Turingmaschine, von der wir $\mathfrak{S}$ abgeleitet haben, deterministisch ist, sind durch die Umkehrungen der Maschine keine Verzweigungen möglich.

Hieraus ergibt sich, daß wir in der Ableitung den Abschnitt

$$w_{p-2} \xrightarrow{\mathfrak{S}^{-1}} w_{p-1} \xrightarrow{\mathfrak{S}} w_p$$

durch

$$w_{p-2} \xrightarrow{\mathfrak{S}} w_p$$

ersetzen dürfen (w_p kommt in keiner Regel von $\mathfrak{S}^{-1}$ auf der rechten Seite vor). Durch Iteration des Schlusses erhalten wir

$$w_0 \xrightarrow{\mathfrak{S}} w_p.$$

Damit haben wir die Ableitungen über $\mathfrak{S}$ durch Ableitungen über $\mathfrak{T}$ simuliert. Die Simulation ist treu, da die Anfangs- und Endkonfigurationen identisch sind. Also haben wir das Wortprobelm von $\mathfrak{S}$ auf das Wortproblem von $\mathfrak{T}$ reduziert.

Sei A Alphabet und $P = f_1, \ldots, f_k \subset A^* \times A^*$. Wir betrachten das von P in $A^* \times A^*$ erzeugte Untermonoid $\langle P \rangle$ und bilden

$$\mathfrak{K} = \langle P \rangle \cap \{(w, w) \mid w \in A^*\}.$$

Der zweite Teil des Ausdrucks enthält das Untermonoid von $A^* \times A^*$, dessen Elemente auf der „Diagonale" liegen. $\mathfrak{K}$ ist offensichtlich auch ein Untermonoid von $A^* \times A^*$.

Problem: Enthält $\mathfrak{K}$ mehr als nur die Einheit ε, in anderen Worten: Gibt es Indizes $i_1, \ldots, i_j$, so daß

$$p_{i_1} \cdot p_{i_2} \cdot \ldots \cdot p_{i_j} = q_{i_1} \cdot q_{i_2} \cdot \ldots \cdot q_{i_j}, \; j > 0.$$

gilt, wenn $f_i = (p_i, q_i)$ ist.

Definition: P heißt **kombinierbar** $\iff \mathfrak{K} \neq \varepsilon$.

Das Problem „P kombinierbar?" ist das **Post'sche Korrespondenzproblem**.

Satz: Das Korrespondenzproblem ist nicht entscheidbar.

Beweis: Zum Beweis gehen wir von einem Semi-Thue-System $\mathfrak{S} = (\mathcal{A}^*, P, \rightarrow)$ und reduzieren das Wortproblem $u \xrightarrow{\mathfrak{S}} v$ auf das Korrespondenzproblem. Hieraus ergibt sich dann die Nichtentscheidbarkeit.

Zunächst ordnen wir P ein Produktionensystem P' zu. Wir bedienen uns hierbei der folgenden Schreibweise

$$\underline{u} = (u, u) \quad , \quad \underline{u}^\beta = (u^\beta, {}^\beta u)$$
$$(u_1, \ldots, u_n)^\beta \;=\; u_1\beta u_2\beta u_3 \ldots \beta u_n\beta,$$
$${}^\beta(u_1, \ldots, u_n) \;=\; \beta u_1\beta u_2 \ldots \beta u_n$$

Wir führen zwei neue Hilfssymbole $\beta, \gamma \notin \mathcal{A}$ ein und setzen

$${}^\beta g_u = (\gamma\gamma\beta, \gamma\gamma^\beta(u\gamma)),\; g_v^\beta = ((\gamma v)^\beta\gamma, \beta\gamma).$$

Nun setzen wir

$$P' = \{\underline{a}^\beta \mid a \in \mathcal{A}\} \cup \{\underline{\gamma}^\beta\} \cup \{f^\beta \mid f \in P\} \cup \{{}^\beta g_u, g_v^\beta\}$$

Das Element γ dient als Trennzeichen zwischen den Wörtern, die in Ableitungen erzeugt werden. Das Zeichen β dient zum Sperren trivialer Kombinationen (a, a).

Sei nun also

$$F = (\underline{u_1} \times f_1 \times \underline{v_1}) \circ (\underline{u_2} \times f_2 \times v_2) \\ \circ \ldots \circ (\underline{u_m} \times f_m \times \underline{v_m})$$

eine Ableitung

$$u \xrightarrow{F} v,\ f_i \in P \text{ für } i = 1, \ldots, m.$$

Wir ordnen F das Produkt

$$F' = {}^\beta g_u \cdot (\underline{u_1}^\beta \cdot f_1^\beta \cdot \underline{v_1}^\beta) \cdot \underline{\gamma}^\beta \cdot (\underline{u_2}^\beta \cdot f_2^\beta \cdot \underline{v_3}^\beta) \\ \cdot \ldots \cdot \underline{\gamma}^\beta \cdot (\underline{u_m}^\beta \cdot f_m^\beta \cdot v_m^\beta) \cdot g_v^\beta$$

zu. Das Produkt liegt in dem Monoid $(\mathcal{A} \cup \{\beta, \gamma\})^* \times (\mathcal{A} \cup \{\beta, \gamma\})^*$.

Wir zeigen: F' liegt genau dann auf der Diagonale, wenn $u \xrightarrow{F} v$ gilt.

Zum Beweis bezeichnen wir mit F_i das i-te Präfix von F. Wir setzen also

$$\begin{aligned}
F'_0 &= {}^{\beta}g_u,\ F_1 = F'_0 \cdot (\underline{u_1}^{\beta} \cdot f_1^{\beta} \cdot \underline{v_1}^{\beta}) \cdot \underline{\gamma}^{\beta} \\
F'_{i+1} &= F'_i \cdot (\underline{u_{i+1}^{\beta}} \cdot f_{i+1}^{\beta} \cdot \underline{v_{i+1}^{\beta}}) \cdot \underline{\gamma}^{\beta} \\
& \qquad \text{für } i = 1, \ldots, m-2, \\
F'_m &= F'_{m-1} \cdot (\underline{u_m^{\beta}} \cdot f_m^{\beta} \cdot \underline{v_m^{\beta}}) \cdot g_v^{\beta}
\end{aligned}$$

Mit $Q(F'_i)$ bezeichnen wir die erste, mit $Z(F'_i)$ die zweite Komponente. Wir haben dann mit $f = (p, q)$:

$$Z(F'_0) = Q(F'_0) \cdot u^{\beta} \cdot \gamma,$$
$$Z(F'_i) = Q(F'_i) \cdot (u_i, q_i, v_i)^{\beta} \cdot \gamma.$$

Wir zeigen letzteres durch Induktion. Wir haben für $i+1 \leq m-1$

$$\begin{aligned}
Z(F'_{i+1}) &= Z(F'_i) \cdot Z(\underline{u^\beta_{i+1}} \cdot f^\beta_{i+1} \cdot \underline{v^\beta_{i+1}}) \cdot Z(\underline{\gamma}^\beta) \\
&= Z(F'_i) \cdot^\beta u_{i+1} \cdot^\beta q_{i+1} \cdot^\beta v_{i+1} \cdot^\beta \gamma \\
&= Q(F'_i) \cdot (u_i q_i v_i)^\beta \cdot \gamma \cdot^\beta (u_{i+1} q_{i+1} v_{i+1} \gamma) \\
&= Q(F'_i) \cdot (u_i q_i v_i \gamma)^\beta \cdot (u_{i+1} q_{i+1} v_{i+1})^\beta \cdot \gamma \\
&= Q(F'_{i+1}) \cdot (u_{i+1} q_{i+1} v_{i+1})^\beta \cdot \gamma
\end{aligned}$$

In den letzten beiden Zeilen wird $u_i q_i v_i = u_{i+1} p_{i+1} v_{i+1}$ verwendet.

Es bleibt der Fall $i = m$:

$$\begin{aligned}
Z(F'_m) &= Z(F'_{m-1}) \cdot Z(\underline{u^\beta_m} f^\beta_m \underline{v^\beta_m}) \cdot Z(g^\beta_v) \\
&= Z(f'_{m-1}) \cdot^\beta (u_m q_m v_m) \cdot \beta\gamma \\
&= Z(F'_{m-1}) \cdot^\beta (u_m q_m v_m \gamma) \\
&= Q(F'_{m-1}) \cdot (u_m q_m v_m)^\beta \cdot \gamma \cdot^\beta (u_m q_m v_m \gamma) \\
&= Q(F'_{m-1}) \cdot (u_m q_m v_m \gamma)^\beta \cdot (u_m q_m v_m)^\beta \gamma \\
&= Q(F'_{m-1}) \cdot Q(\underline{u^\beta_m} f^\beta_m \underline{v^\beta_m}) \cdot Q(\underline{\gamma}^\beta) \cdot v^\beta \cdot \gamma \\
&= Q(F'_{m-1} \cdot (\underline{u^\beta_m} f^\beta_m \underline{v^\beta_m}) \cdot (\underline{\gamma v})^\beta \cdot \underline{\gamma}) \\
&= Q(F'_{m-1} \cdot (\underline{u^\beta_m} f^\beta_m \underline{v^\beta_m}) \cdot g^\beta_v) \\
&= Q(F'_m).
\end{aligned}$$

Damit haben wir gezeigt, daß $F' = F'_m$ in der Tat eine Kombination von P' ist.

Wir zeigen nun die Umkehrung. Sei also P' kombinierbar. Wir zeigen: Daraus folgt $u \xrightarrow{P} v$.

Sei $H = h_0 h_1 \dots h_n h_{n+1}$ eine Kombination von P', d. h. $Q(H) = Z(H)$.

Hieraus folgt $Q(h_0)$ und $Z(h_0)$ haben ein gemeinsames Präfix.

Ebenso folgt $Q(h_{n+1})$, $Z(h_{n+1})$ haben ein gemeinsames Suffix.

hierdurch ergibt sich eindeutig

$$h_0 =^{\beta} g_u, \; h_{n+1} = g_v^{\beta}.$$

Wir betrachten das minimale Präfix von $h_0 \dots h_{n+1}$, das mit g_v^{β} endet. Dann ist bereits dieses Teilprodukt eine Kombination. Wir dürfen also annehmen, daß $h_i \neq^{\beta} g_u, g_v^{\beta}$ für $1 \leq i \leq n$. O.B.d.A. dürfen wir weiter annehmen, daß keine zwei Faktoren $\underline{\gamma}^{\beta}$ aufeinanderfolgen.

Nun definieren wir

$$
\begin{aligned}
H^1\underline{\gamma}^\beta &= {}^\beta g_u \cdot h_1 \dots h_{i_1} \cdot \underline{\gamma}^\beta \\
&\quad \text{mit } h_1, \dots, h_{i_1} \neq \underline{\gamma}^\beta. \\
H^{l+1\beta}\gamma &= H^l\underline{\gamma}^\beta h_{i_l+1} \dots h_{i_{l+1}} \cdot \underline{\gamma}^\beta \\
&\quad \text{mit } h_{i_l+1}, \dots, h_{i_{l+1}} \neq \underline{\gamma}^\beta \\
H^m &= H^{m-1\beta}\underline{\gamma} h_{i_{m-1}+1} \dots h_n \cdot g_v^\beta, \\
&\quad h_{i_{m-1}+1}, \dots, h_n \neq \underline{\gamma}^\beta.
\end{aligned}
$$

Nun ist

$$
\begin{aligned}
Q(H^1) &= Q^(\beta g_u) \cdot Q(h_1 \dots h_{i_1}) \\
&= \gamma\gamma\beta \cdot Q(h_1 \dots h_{i_1}) \\
Z(H^1) &= Z({}^\beta g_u) \cdot Z(h_1 \dots h_{i_1}) \\
&= \gamma\gamma^\beta (u\gamma) \cdot Z(h_1 \dots h_{i_1}) \\
&= \gamma\gamma\beta\ u^\beta \cdot \gamma \cdot Z(h_1 \dots h_{i_1})
\end{aligned}
$$

Durch Vergleich erhält man $Q(h_1 \dots h_{i_1}) = u^\beta$
und

$$
Z(H^1) = Q(H^1) \cdot \gamma \cdot Z(h_1 \dots h_{i_1}).
$$

Wir machen den allgemeinen Ansatz

$$Z(H^l) = Q(H^l) \cdot \gamma \cdot Z(h_{i_{l-1}+1} \ldots h_{i_l})$$

und setzen dies induktiv auf $l+1$ fort.

$$\begin{array}{rcl}
Z(H^{l+1}) & = & Z(H^l \cdot \underline{\gamma}^\beta \cdot h_{i_l+1} \ldots h_{i_{l+1}}) \\
 & = & Z(H^l) \cdot \beta \cdot \gamma \cdot Z(h_{i_l+1} \ldots h_{i_{l+1}}) \\
 & = & Q(H^l) \cdot \gamma \cdot Z(h_{i_{l-1}+1} \ldots h_{i_l}) \\
 & & \cdot \beta \cdot \gamma \cdot Z(h_{i_{l+1}} \ldots h_{i_{l+1}}) \\
Q(H^{l+1} \underline{\gamma}^\beta) & = & Q(H^l) \cdot \gamma \cdot \beta \cdot Q(h_{i_l+1} \ldots h_{i_{l+1}}) \\
 & & \cdot \gamma \cdot \beta
\end{array}$$

Durch Vergleich:

$$Z(h_{i_{l-1}+1} \ldots h_{i_l})\beta = \beta \cdot Q(h_{i_l+1} \ldots h_{i_{l+1}})$$

Also:

$$\begin{array}{rcl}
Z(H^{l+1}) & = & Q(H^l) \cdot \gamma \cdot \beta \cdot Q(h_{i_l+1} \ldots h_{i_{l+1}}) \\
 & & \cdot \gamma \cdot Z(h_{i_l+1} \ldots h_{i_{l+1}}) \\
 & = & Q(H^{l+1}) \cdot \gamma \cdot Z(h_{i_l+1} \ldots h_{i_{l+1}})
\end{array}$$

Weiter folgt wegen

$$Q(h_{i_l+1}\dots h_{i_{l+1}})_\beta \longrightarrow Z(H_{i_l+1}\dots h_{i_{l+1}})_\beta$$

Index β : Vergiß β!

Also

$$u \xrightarrow{P} Z(h_{i_l+1}\dots h_{i_{l+1}})_\beta.$$

Es bleibt der Fall $l = m$.

$$\begin{aligned}
Q(H^m) &= Q(H^{m-1})Q(\underline{\gamma}^\beta)\cdot Q(h_{i_{m-1}+i}\dots h_{i_m})\\
&\quad \cdot Q(g\beta_v)\\
&= Q(H^{m-1})\cdot\gamma\cdot\beta\cdot Q(h_{i_{m-1}+1}\dots h_{i_m})\\
&\quad \cdot(\gamma v)^\beta\cdot\gamma\\
Z(H^m) &= Z(H^{m-1})\cdot\beta\gamma\cdot Z(h_{i_{m-1}+1}\dots h_{i_m})\cdot\beta\gamma\\
&= Q(H^{m-1})\cdot\gamma\cdot Z(h_{i_{m-2}+1}\cdots h_{i_{m-1}})\\
&\quad \cdot\beta\cdot\gamma\cdot Z(h_{i_{m-1}}\dots h_{i_m})\cdot\beta\cdot\gamma
\end{aligned}$$

Wegen $Z(H^m) = Q(H^m)$ folgt nun

$$\beta Q(h_{i_{m-1}} \dots h_{i_m}) = Z(h_{i_{m-2}+1} \dots h_{i_{m-1}})^\beta$$

und

$$^\beta v = Z(h_{i_{m-1}} \dots h_{i_m}).$$

Also gilt wegen

$$u \longrightarrow Z(h_{i_{m-2}} \dots h_{i_{m-1}})_\beta$$

und

$$Q(h_{i_{m-1}} \dots h_{i_m})_\beta \longrightarrow Z(h_{i_{m-1}} \dots h_{i_m})_\beta$$

$$u \xrightarrow{P} v.$$

Satz: Sei $Y \subset A^* \times A^*$ und $\langle Y \rangle$ das durch Y erzeugte Untermonoid von $A^* \times A^*$. Weiter sei $\#A \geq 2$. Das Problem „Ist Y freies Erzeugendensystem?“ ist nicht generell entscheidbar.

Beweis: Wir führen den Beweis, indem wir das Post'sche Korrespondenzproblem auf das „Freiheitsproblem“ reduzieren. Sei $P \subset A^* \times A^*$ ein vorgegebenes Post'sches System. Da $\#A \geq 2$ ist, gibt es einen präfixfreien Code $C \subset A^*$, der ebensoviele Elemente wie P besitzt; es gibt also auch eine bijektive Abbildung

$$i : P \longrightarrow C.$$

Nun bilden wir

$$\begin{array}{rcl} Y & := & \{(P^{\beta}, i(f)) \mid f \in P, f = (p,q)\} \\ & \cup & \{({}^{\beta}q, i(f)) \mid f \in P, f = (p,q)\} \\ & \cup & \{(\beta, 1)\}. \end{array}$$

(1) P kombinierbar $\Longrightarrow$ Y nicht frei.

Sei also

$$f = f_1 \cdot f_2 \cdot \ldots \cdot f_m \in \langle P \rangle.$$

f liegt genau dann auf der Diagonalen, wenn

$$p_1 \cdot p_2 \cdot \ldots \cdot p_m = q_1 \cdot q_2 \cdot \ldots \cdot q_m$$

gilt.

Hieraus folgt in diesem Fall

$$\beta \cdot (p_1 \cdot \ldots \cdot p_m)^\beta =^\beta (q_1 \cdot \ldots \cdot q_m) \cdot \beta.$$

Nun setzen wir

$$\left.\begin{array}{lcl} y_l & := & (p_l^\beta, i(f_l)) \\ z_l & := & ({}^\beta q_l, i(f_l)) \end{array}\right\} \text{ für } f_l = (p_l, q_l) \in P.$$

Wir erhalten

$$(\beta, 1) \cdot y_1 \cdot \ldots \cdot y_m = z_1 \cdot \ldots \cdot z_m \cdot (\beta, 1).$$

Also ist Y nicht frei.

(2) Sei nun Y nicht frei. Wir zeigen, daß dann P kombinierbar ist.

Sei also $g_1, \ldots, g_m$, $h_1, \ldots, h_n \in Y$ und

$$g_1 \cdot \ldots \cdot g_m = h_1 \cdot \ldots \cdot h_n$$

eine nicht-triviale Gleichheit. Wir dürfen dann aufgrund der Gültigkeit der Kürzungsregel in A^* annehmen, daß

$$g_1 \neq h_1 \text{ und } g_m \neq h_n$$

gilt.

Aufgrund der angenommenen Gleichheit sind die Komponenten von h_1 und g_1 Präfixe voneinander. Da C präfixfrei ist, kann das aufgrund der β-Einschiebungen nur sein, wenn

$$g_1 = (\beta, 1) \text{ und } h_1 \text{ ein „}q\text{-Faktor“}.$$

oder

$$g_1 \text{ ein „}p\text{-Faktor“ und } h_1 = (\beta, \varepsilon)$$

ist.

Nun zerlegen wir das Produkt in Teilprodukte, indem wir β^2 als Trennzeichen verwenden.

Diese Zerlegung liefert uns Teilprodukte, so daß entsprechende Faktoren jeweils gleich sind.

Wir dürfen annehmen, daß

$$g_1 = (\beta, 1) \text{ und } h_n = (\beta, 1)$$

gilt, und daß alle anderen Faktoren von $(\beta, 1)$ verschieden sind.

Da C ein präfixfreier Code ist, folgt nun

$$g_l = (p_l^\beta, i(f_l)) \text{ und } h_l = ({}^\beta q_l, i(f_l))$$

für $l = 2, \ldots, m-1$ und $n = m$.

Also ist

$$\beta(p_1 \cdot \ldots \cdot p_{m-1})^\beta =^\beta (q_1 \cdot \ldots \cdot q_{m-1}) \cdot \beta,$$

d. h. aus Y nicht frei folgt P ist kombinierbar.

Korollar: Die Nichtentscheidbarkeit des Freiheitsproblemes gilt bereits für $\#A = 2$.

Beweis: Kodiere A in ein freies Erzeugendensystem $E \subset \{0, 1\}^*$.

$G = (Z, T, P, S)$ heißt (Chomsky-)Grammatik $:\Longleftrightarrow$ Z und T sind endliche Alphabete, es ist

$$Z \cap T = \emptyset,\ S \in Z,\ P \subset Z^* \times (Z \cup T)^*,$$

und

$$\mathfrak{S} = (Z \cup T, P, \longrightarrow)$$

ist ein Semi-Thue-System.

$$L(G) := \{w \in T^* \mid S \xrightarrow{\mathfrak{S}} w\}$$

heißt die zu G gehörige (durch G definierte) Sprache.

$A := Z \cup T$ heißt das **Alphabet** von G, S das **Axiom**, T die Menge der **Terminalen** und Z das **Variablenalphabet**.

Häufig schreibt man auch $A = Z \cup T$ und

$$G = (A, T, P, S).$$

Chomsky‘s Hoffnung: Erfassung der Syntax natürlicher Sprachen durch formale Sprachen $L(G)$.

Aus den Grundvorlesungen ist bekannt, daß man die Syntax von Programmiersprachen durch solche Sprachen erfassen kann.

Vereinbarung: Für die Elemente von T wählen wir kleine, für die Elemente von Z große Buchstaben.

Beispiele:

(1)
$$\begin{array}{lcl} A & := & \{S, X, a, b\},\ T := \{a, b\} \\ P & := & \{(S, aS), (S, aX), (X, bX), (X, b)\} \end{array}$$

Ableitungen sind

$$S \longrightarrow aS \longrightarrow aaS \longrightarrow aaaS \longrightarrow aaaX \longrightarrow aaab$$

oder

$$S \longrightarrow aX \longrightarrow abX \longrightarrow abbX \longrightarrow abbbX \longrightarrow abbbb.$$

$$L(G) = \{a^m b^n \mid m, n \in \mathbb{N} - \{0\}\}.$$

(2)
$$\begin{array}{rcl} A & = & \{S, a, b\},\ T = \{a, b\} \\ P & = & \{(S, aSb), (S, 1)\} \\ L(G) & = & \{a^n b^n \mid n \in \mathbb{N}\}. \end{array}$$

Die Theorie behandelt folgende Fragen:

1) **Wortproblem und Äquivalenzproblem**

   Wortproblem: $w \in L(G)$? für $w \in T^*$.

   $L(G) \;=\; L(G')$? für Grammatiken G, G'.
   $L(G) \;=\; \emptyset$?, $L(G) = T^*$?

2) **Eindeutigkeit und Mehrdeutigkeit**
   Das ist die Frage, ob sich für $w \in L(G)$ „verschiedene" Herleitungen angeben lassen. Die Definition der Eindeutigkeit formaler Sprachen ist nicht trivial.
   Im Falle spezieller Sprachklassen läßt sie sich durch die Gleichheit von „Ableitungsbäumen" erfassen.

3) **Hierarchiefragen und Abschlußeigenschaften**
Hierarchiefragen werden durch die verschiedenen Grammatikklassen aufgeworfen werden. Abschlußeigenschaften haben wir bereits ausführlich in dem Abschnitt über endliche Automaten untersucht.

4) **Korrespondenzen zwischen Maschinenklassen und Sprachklassen**
Gegeben sei eine Sprachklasse $\mathfrak{K}$ und eine Maschinenklasse $\mathfrak{M}$: Kann das Wortproblem für die Sprachen aus L durch eine Maschine aus $\mathfrak{M}$ entschieden werden?
Liegen die Wortmengen, die Maschinen aus $\mathfrak{M}$ akzeptieren, in der Sprachklasse $\mathfrak{K}$?

Die Chomsky-Hierarchie

Typ 3: G ist **einseitig linear** $:\Longleftrightarrow$
G ist links linear oder rechtslinear. (Siehe: Endl. Automaten)

Typ 2: G ist **kontextfrei** $:\Longleftrightarrow$

$$P \subset Z \times (Z \cup T)^*.$$

Typ 1: G ist **kontext-sensitiv** $:\Longleftrightarrow$
Es gibt eine kontextfreie Grammatik $\tilde{P}$ mit $(z,u) \in \tilde{P} \Longrightarrow u \neq 1$ und zu jeder Produktion $f \in \tilde{P}$ eine Menge $k_f \subset A^* \times A^*$, so daß

$$P = \{ufv \mid (u,v) \in K_f, f \in \tilde{P}\}$$

Sprechweise: f darf nur im Kontext (u,v) angewendet werden.

Typ 0: P unterliegt keinen Einschränkungen.

Satz: Typ 3 $\subsetneq$ Typ 2 $\subsetneq$ Typ 1 $\subsetneq$ Typ 0.

Beweis: **(1)** Wie wir gesehen haben, ist $L = \{a^n b^n \mid u \in \mathbb{N}\}$ nicht regulär. Also $L \notin$ Typ 3. L ist aber offensichtlich in Typ 2.

(2) Typ 1 $\exists L = \{a^n b^n c^n \mid n \in \mathbb{N}\} \notin Typ 2$.

Beweisskizze: Die Erzeugung der a- und c-Komponente kann man nicht beide an die Erzeugung der b-Komponente ankoppeln, da die Kopplung durch jeweils eine Variable auf der Nahtstelle zwischen der a- und b- bzw. der b- und c-Komponente erfolgen muß. Diese Variablen werden aber mit wachsendem n über jeden endlichen Abstand hinaus auseinandergetrieben. Also können diese Variablen nicht durch die rechte Seite einer Produktion verbunden bleiben.

Da Z aber auch endlich ist, weiß nach einiger Zeit die Rechte nicht mehr, was die Linke tut. Der formale Beweis beruht auf dem „Pumping-Lemma“für k.-f. Sprachen. Also ist Typ 2 $\neq$Typ 1. Zum Beweis für Typ 2 $\subset$ Typ 1 müssen wir den Fall $z \longrightarrow 1 \in P$ noch diskutieren. Den Beweis für diese Eigenschaften verschieben wir auf spätere Vorlesungen.

(3) Das Wortproblem für Typ 1-Sprachen ist entscheidbar, da aus $u \xrightarrow{P} v$ folgt $|u| \leq |v|$. Indem wir zeigen, daß das Wortproblem für Typ 0-Sprachen nicht generell entscheidbar ist, folgt auch

$$\text{Typ 1} \subsetneq \text{Typ 0.}$$

Satz: Das Wortproblem $w \in L(G)$ ist nicht generell entscheidbar.

Beweis: Wir gehen von einem beliebigen S.T.S. aus

$$\mathfrak{S} = (A, P, \longrightarrow).$$

Nun wählen wir $S \in A$ und $a \in A$ und bilden

$$\begin{aligned} A' &:= A \cup \{S, a\}, \\ P' &:= P \cup \{(S, u), (v, a)\} \end{aligned}$$

mit beliebigen $u, v \in A^*$. Wir setzen nun $T = \{a\}$

$$G = (A', T, P', S).$$

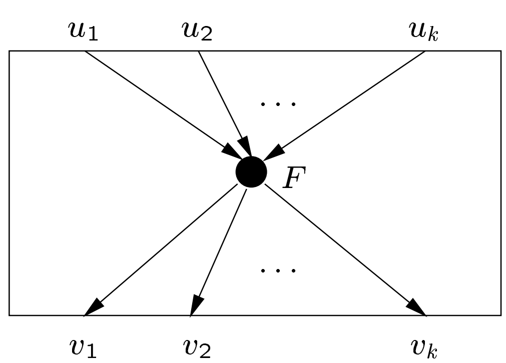
Die universelle Lösbarkeit des Wortproblems für Chomsky-Sprachen würde uns für jedes $u, v \in A^*$ das Problem

$$S \longrightarrow u \longrightarrow v \longrightarrow a$$

entscheiden und damit auch $u \xrightarrow{\mathfrak{S}} v$. Widerspruch.

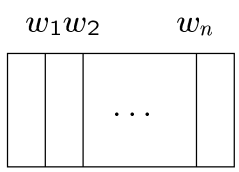
Zur Definition der Eindeutigkeit von Grammatiken ordnen wir Ableitungen von Semi-Thue-Systemen einen Graph zu.

Der Produktion $f = (u, v) \in P$ ordnen wir den durch das folgende Diagramm definierten Graph F zu. Sei $u = u_1 \cdot \ldots \cdot u_k$, $v = (v_1, \ldots, v_l)$; $u_i, v_j \in A$.



Der Graph ist in ein Rechteck eingespannt. Er enthält einen inneren Knoten; k Kanten münden in F ein. l Kanten gehen von dem Knoten aus. Der „Stern“ der Kanten wird durch (u, v) eindeutig festgelegt.

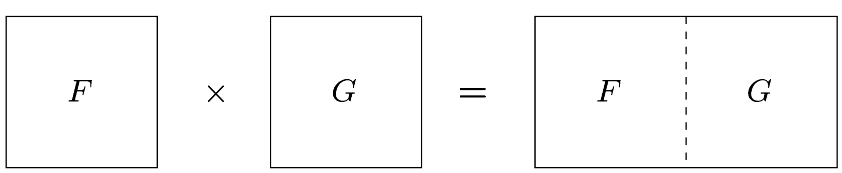
Der trivialen Relation $w \longrightarrow w$ ordnen wir

$1_w :=$ 

zu. Nun ordnen wir rekursiv Ableitungen Netze zu.

Ist F ein Netz, dann bezeichnet $Q(F)$ die Belegung der Eingangskanten, $Z(F)$ die Belegung der Ausgangskanten mit Elementen aus A.

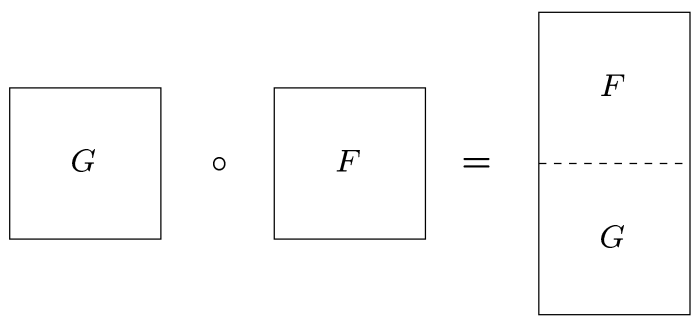
Zwei Netze F und G dürfen wir nebeneinandersetzen zu dem Netz $F \times G$.



Wir haben

$$\begin{aligned} Q(F \times G) &= Q(F) \cdot Q(G) \\ Z(F \times G) &= Z(F) \cdot Z(G). \end{aligned}$$

Netze F, G dürfen wir hintereinanderhängen, falls $Z(F) = Q(G)$ ist:



Es gilt $Q(G \circ F) = Q(F)$, $Z(G \circ F) = Z(G)$.

Beginnen wir mit den Netzen F, die wir $f \in P$ zugeordnet haben. Dann erzeugen die Operationen $\circ$ und $\times$ aus diesen Netzen und den Einheiten von 1_w eine Menge $\mathfrak{A}(P)$ von Netzen. Zwei Elemente von $\mathfrak{A}(P)$ heißen gleich, wenn sie als markierte Graphen isomorph sind.

Man sieht leicht:

$$F \in \mathfrak{A}(P) \Longrightarrow Q(F) \xrightarrow{P} Z(F).$$

und zu jeder Ableitungsfolge

$$w = w_1 \longrightarrow w_2 \longrightarrow w_3 \longrightarrow \ldots \longrightarrow w_{n+1} = \tilde{w}$$

mit

$$w_i = u_i p_i v_i \longrightarrow u_i q_i v_i$$

und $f_i = (p_i, q_i) \in P$ gibt es ein Netz

$$F_n \circ F_{n-1} \circ \ldots \circ F_1 \text{ mit } F_i = 1_{u_i} \times f_i \times 1_{v_i}$$

und $f_i = (p_i, q_i) \in P$.

Definition: F heißt Ableitung von $Q(F)$ in $Z(F)$ über P. Multiplizität $(u \xrightarrow{P} v) := \#\{F \in \mathfrak{A}(P) \mid Q(F) = u, Z(F) = v\}$.

Definition: $G = (Z, T, P, S)$ heißt **eindeutig** $:\Longleftrightarrow$ Für alle $w \in T^*$ gilt Multiplizität $(S \longrightarrow w) \leq 1$.

Ist G kontextfrei, dann sind die Netze $F \in \mathfrak{A}(P)$ Bäume, falls $\#Q(F) = 1$ ist. Also alle Ableitungsnetze kontextfreier Sprachen sind Bäume.

Definition: $G = (Z, T, P, S)$ heißt **linear** : $\iff \; P \subset Z \times T^* \cdot Z \cdot T^*$.
G heißt **minimal** linear : $\iff \; G$ ist linear $\& \#Z = 1$.

Die Ableitungsbäume zu linearen Grammatiken sind sehr einfach. Sie werden jeweils durch die Folge $(f_n, f_{n-1}, \ldots, f_1)$ der sukzessive angewendeten Produktionen eindeutig bestimmt.

Satz: Es ist nicht entscheidbar, ob minimal lineare Grammatiken eindeutig sind, falls $\#T \geq 2$ ist.

Beweis: Sei $Y \subset T^* \times T^*$ endlich.

$$\begin{aligned} Y &= \{(u_1, v_1, \ldots, (u_k, v_k))\},\ d \notin T. \\ T' &:= T \cup \{d\}. \\ G &:= (\{S\}, T', P, S) \end{aligned}$$

mit

$$P := \{(S, d), (S, uSv^{sp}) \mid (u, v) \in Y\}.$$

Offensichtlich gilt

$$L(G) = \{udv^{sp} \mid (u, v) \in \langle Y \rangle\}.$$

Es gibt genau dann mindestens zwei verschiedene Weisen (u, v) zu erzeugen, wenn Y nicht freies Erzeugendensystem von $\langle Y \rangle$ ist. Letzteres ist aber, wie wir gezeigt haben, nicht entscheidbar.