

Vorlesung Baumautomaten (Mitschrift)

Benedikt Elßmann (3720358)
be57xocu@studserv.uni-leipzig.de

Universität Leipzig

22. Juni 2019

Inhaltsverzeichnis

0	Einleitung	3
1	Bäume und Baumautomaten	3
1.1	Definition Rangalphabet	4
1.2	Definition Term, Tree	4
1.3	Definition Höhe	5
1.4	Definition Position	5
1.5	Definition der Label an den Positionen	5
1.6	Definition Sub-Baum	5
1.7	Definition Baumautomat	6
1.8	Definition Lauf/Run	6
1.9	Lemma	7
1.10	Definition Determinismus	9
1.11	Satz	9
1.12	Definition vollständig und reduziert	11
1.13	Satz	11
1.14	Definition Kontext	11
1.15	Pumping-Lemma	12
1.16	Korollar	13
1.17	Abschlusseigenschaften	13
1.18	Definition Kongruenz	14
1.19	Definition	14
1.20	Lemma	14
1.21	Theorem (Myhill-Nerode)	15
1.22	Korollar	16
1.23	Einschub - Homomorphismen von Baumsprachen	16
1.23.1	Allgemeine Homomorphismen	16
1.23.2	Worthomomorphismen	16
1.23.3	Baumhomomorphismen	17
1.23.4	lineare Terme	18
1.23.5	linearer Homomorphismus	18
1.23.6	Satz	19
1.23.7	Satz	19
1.24	Top-Down Baumautomaten	20
1.25	Satz	21
2	Grammatiken	21
2.1	Definition - Grammatik	21
2.2	Definition	22
2.3	Definition - reduziert	22
2.4	Satz	23
2.5	Definition - Normalisierung	23
2.6	Satz	23
2.7	Theorem	23
2.8	Satz	25

2.9 Satz	25
2.10 Definition	26
2.11 Theorem	26
2.12 Satz	28
2.13 Satz	29

0 Einleitung

Automaten lesen Wörter $w = a_1 \dots a_n$ und geben "accept" aus oder nicht. Dafür gibt es Erweiterungen, wie etwa:

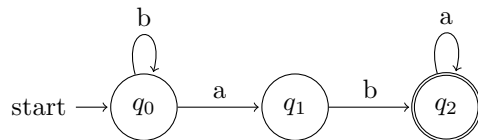
- gewichtete Automaten, das heißt der Output ist ein Semiringelement
- Automaten mit Gedächtnis (Stack)
- Automaten über anderen Strukturen
 - ω -Wörter $w = a_1 \dots a_n$
 - Graphen
 - Bäume
 - Kombinationen dieser

Typische Fragestellungen:

- Ausdrucksstärke
- Darstellung als rationale Ausdrücke (Kleene)
- Darstellung als Grammatik
- Darstellung als Logik

1 Bäume und Baumautomaten

Wir betrachten über $A = \{a, b\}$ den Automaten \mathcal{A} :



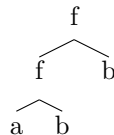
mit $L(\mathcal{A}) = b^*aba^*$.

Betrachtung des Wortes $w = baba \in L(\mathcal{A})$:

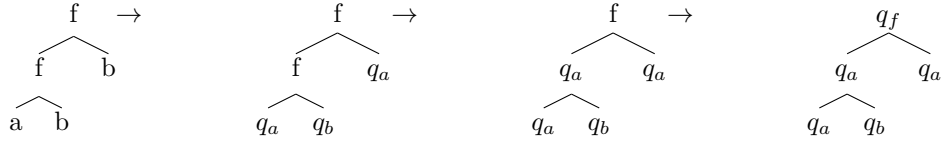
Der eindeutige erfolgreiche Lauf für w lässt sich darstellen als:

$q_0baba \rightarrow bq_0aba \rightarrow baq_1ba \rightarrow babq_2a \rightarrow babaq_2 \in F$ (Finalzustand)

Baumautomaten funktionieren analog. Unser erstes Beispiel wird



Akzeptiert mit dem Lauf:



mit $q_f \in F$

1.1 Definition Rangalphabet

Ein paar (Σ, rk) , wobei Σ eine endliche Menge von Symbolen und $rk : \Sigma \rightarrow \mathbb{N}$ eine Abbildung ist, heißt Rangalphabet.

Für $f \in \Sigma$ heißt $rk(f)$ der Rang (oder die Stelligkeit) von f .

Intuitiv: $rk(f)$ ist die Anzahl der Kinder von f in einem Baum.

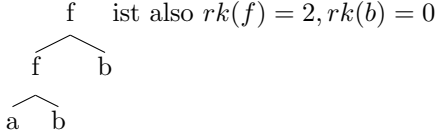
Insbesondere ist die Anzahl der Kinder für jedes Symbol fest.

Gilt $rk(f) = n$, schreiben wir auch $f^{(n)}$ statt f . wir schreiben:

- 0-stellige Symbole (Konstanten) a, b, \dots
- unär, binär, $\dots f, g, \dots$

Wir setzen $\Sigma^{(n)} = \{f \in \Sigma \mid rk(f) = n\}$

In



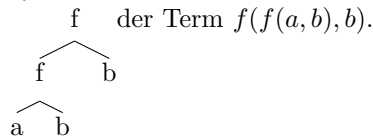
1.2 Definition Term, Tree

Sei (Σ, rk) ein Rangalphabet. Die Menge T_Σ der Bäume über Σ ist induktiv definiert durch:

- $\Sigma^0 \subseteq T_\Sigma$
- $f^{(n)} \in \Sigma$, $t_1, \dots, t_n \in T_\Sigma$, dann ist $f(t_1, \dots, t_n) \in T_\Sigma$

Intuitiv sind t_1, \dots, t_n die Kinder von f .

Z.B. ist



1.3 Definition Höhe

Sei (Σ, rk) ein Rangalphabet. Die Höhe ht ist gegeben durch:

- für $a^{(0)} \in \Sigma : ht(a) = 1.$
- für $f(t_1, \dots, t_n) \in T_\Sigma : ht(f) = 1 + \max\{ht(t_i) | i \in \{1, \dots, n\}\}$

Ziel: Zugriff auf einen Knoten innerhalb eines Baumes und deren Label.
Dafür ordnen wir den Knoten Positionen zu. Das geht induktiv wie folgt:

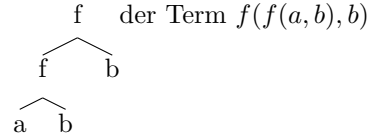
1.4 Definition Position

Sei (Σ, rk) ein Rangalphabet. Die Positionenmenge ist definiert durch:

- für $a^{(0)} \in T_\Sigma$ ist $Pos(a) = \{\varepsilon\}$
- für $f(t_1, \dots, t_n) \in T_\Sigma$ ist $Pos(f(t_1, \dots, t_n)) = \{\varepsilon\} \cup 1 \cdot Pos(t_1) \cup \dots \cup n \cdot Pos(t_n)$

Beispiel:

Betrachtung von $f(f(a, b), b)$ bzw.



$$Pos(f) = \{\varepsilon, 1, 2, 1.1, 1.2\}$$

1.5 Definition der Label an den Positionen

Für einen Term der Form $t = f(t_1, \dots, t_n)$ ist das Symbol $t(p)$ in t an p-ter Position induktiv definiert durch:

- $t(\varepsilon) = f$
- $t(ip) = t_i(p), i \in \{1, \dots, n\}$

Beispiel: Betrachtung von $f(f(a, b), b)$

Dann ist

$$t(\varepsilon) = f$$

$$t(1) = t(1 \cdot \varepsilon) = t_1(\varepsilon) = f$$

$$t(2) = t(2 \cdot \varepsilon) = t_2(\varepsilon) = b$$

$$t(1.1) = t_1(1) = a$$

$$t(1.2) = t_2(1) = b$$

1.6 Definition Sub-Baum

Für T_Σ ist ein Sub-Baum $t|_p$ an p-ter Position wie folgt definiert:

- $Pos(t|_p) = \{i | pi \in Pos(t)\}$

- $\forall q \in Pos(t|_p)$ ist $t|_p(q) = t(pq)$

Wir schreiben $t[u]_p$ für den Baum, der entsteht, wenn man in t den sub-Baum $t|_p$ durch u ersetzt.

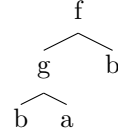
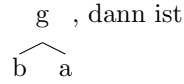
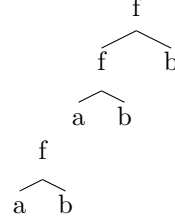
Beispiel: $f(f(a, b), b)$ bzw.

$$t|_1 = f(a, b)$$

$$t|_2 = t(1.2) = b$$

$$u = g(b, a)$$

$$t[u]_1 = f(g(b, a), b)$$



1.7 Definition Baumautomat

Ein Baumautomat \mathcal{A} ist ein 4-Tupel (Q, Σ, F, Δ) , wobei:

$Q \dots$ endliche Menge an Zuständen

$\Sigma \dots$ Rangalphabet, wobei $\Sigma \cup Q \neq \emptyset$

$F \dots \subseteq Q$ Finalzustände

$\Delta \dots$ Menge von Regeln

$$r : f(q_1 \dots q_n) \rightarrow q$$

für $q, q_1, \dots, q_n \in Q$, für $a^{(0)} \in T_\Sigma : a \rightarrow q$

Beispiel:

$$\mathcal{A} = \{\{q_a, q_b, q_f\}, \{a^{(0)}, b^{(0)}, f^{(2)}\}, \{q_f\}, \Delta\}$$

$$\text{mit } \Delta = \{a \rightarrow q_a, b \rightarrow q_b, f(q_a, q_b) \rightarrow q_a, f(q_a, q_b) \rightarrow q_f\}$$

1.8 Definition Lauf/Run

Sei $\mathcal{A} = (Q, \Sigma, F, \Delta)$ ein Baumautomat und $t \in T_\Sigma$. Ein Lauf r für t von \mathcal{A} ist ein Term mit

- $Pos(r) = Pos(t)$
- Ist $t(p) = a$ ein Blatt, dann ist $r(p) = q_a$, nur wenn $(a \rightarrow q_a) \in \Delta$
- Ist $t(p) = f^{(m)}$, dann ist $r(p) = q$, wenn $(f(q_1, \dots, q_n) \rightarrow q) \in \Delta$ und $r(p_i) = q_i, i \in \{1, \dots, n\}$

Ein Lauf ist erfolgreich, wenn $r(\varepsilon) \in F$. Der Automat \mathcal{A} akzeptiert t , falls es einen erfolgreichen Lauf für t von \mathcal{A} gibt.

Wir bezeichnen mit $L(\mathcal{A}) = \{t \in T_\Sigma \mid \mathcal{A} \text{ akzeptiert } t\}$ die von \mathcal{A} erkannte Baumsprache. Eine Sprache $L \subseteq T_\Sigma$ heißt erkennbar, falls ein Baumautomat \mathcal{A} existiert mit $L = L(\mathcal{A})$.

Um einzelne Schritte von Baumautomaten zu formalisieren, betrachten wir die *move relation* $\rightarrow_{\mathcal{A}}$, definiert wie folgt:

Gegeben sei $\mathcal{A} = (Q, \Sigma, F, \Delta)$, dann ist $t \rightarrow_{\mathcal{A}} t'$ mit $t, t' \in T_{\Sigma \cup Q}$, falls

- $t(p) = f^{(n)}$
- $t(pi) = q_i$ für $i \in \{1, \dots, n\}$ und p_i sind Blätter
- $(f(q_1, \dots, q_n) \rightarrow q) \in \Delta$
- und $t' = t[q]_p$

Mit $\rightarrow_{\mathcal{A}}^*$ bezeichnen wir die transitive Hülle von $\rightarrow_{\mathcal{A}}$.

1.9 Lemma

Sei $\mathcal{A} = (Q, \Sigma, F, \Delta)$ ein Baumautomat. Dann ist $L(\mathcal{A}) = \{t \in T_\Sigma \mid t \rightarrow_{\mathcal{A}}^* q \text{ mit } q \in F\} (= Z)$

Beweis: „ $L(\mathcal{A}) \subseteq Z$ “:

Wir zeigen: Es existiert ein Run r für t von \mathcal{A} mit $r(\varepsilon) = q$, dann ist $t \rightarrow_{\mathcal{A}}^* q$

Inuktionsannahme:

$t = a^{(0)} \in T_\Sigma$. Dann gilt $a \in L(\mathcal{A})$, falls ein Lauf r existiert mit $r(a) = q_a$ und $(a \rightarrow q_a) \in \Delta$. Dann folgt $a \rightarrow_{\mathcal{A}}^* q_a$.

Sei nun $t = f(t_1, \dots, t_n)$

Induktionsvoraussetzung:

Falls für t_1, \dots, t_n Läufe r_i existieren mit $r_i(\varepsilon) = q_i$, dann gilt auch $t_i \rightarrow_{\mathcal{A}}^* q_i$ mit $i \in \{1, \dots, n\}$

Induktionsschritt:

zu zeigen: Es existiert ein Lauf r für t mit $r(\varepsilon) = q$, dann $t \rightarrow_{\mathcal{A}}^* q$.

Sei also r ein Lauf mit $r(\varepsilon) = q$. Dann ist $r(i) = q_i$, $i \in \{1, \dots, n\}$, mit $(f(q_1, \dots, q_n) \rightarrow q) \in \Delta$.

Laut Induktionsvoraussetzung gilt nun, $t_i \rightarrow_{\mathcal{A}}^* q_i$, $i \in \{1, \dots, n\}$.

Damit $t = f(t_1, \dots, t_n) \rightarrow_{\mathcal{A}}^* f(q_1, t_2, \dots, t_n) \rightarrow_{\mathcal{A}}^* \dots \rightarrow_{\mathcal{A}}^* f(q_1, \dots, q_n)$

Des weiteren haben wir die regel $f(q_1, \dots, q_n) \rightarrow q$, das heißt $f(q_1, \dots, q_n) \rightarrow_{\mathcal{A}}^* q$.

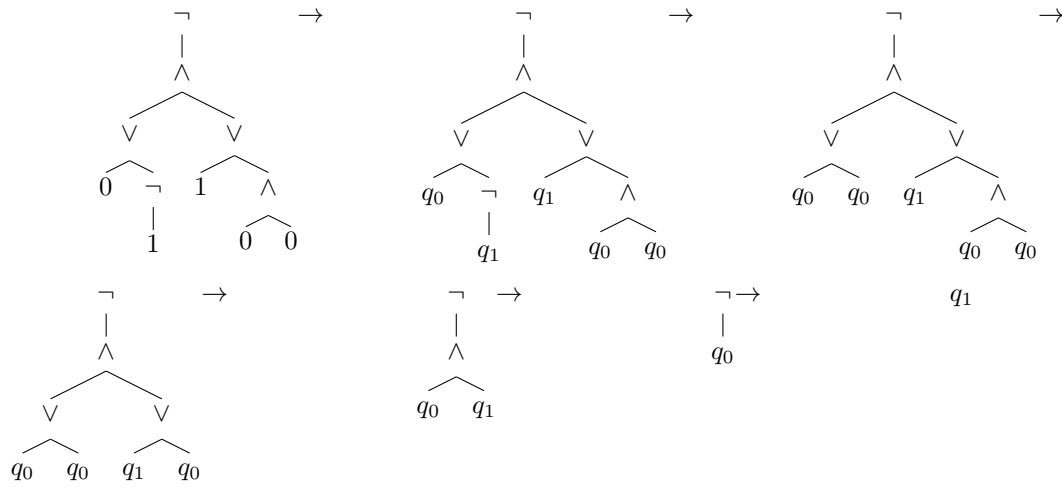
Insgesamt also $t \rightarrow_{\mathcal{A}}^* q$

Beweis: „ $L(Z \subseteq \mathcal{A})$ “: analog

Einige Beispiele für Baautomaten:

1. Sei $B = (\{q_0, q_1\}, \{0^{(0)}, 1^{(0)}, \neg^{(1)}, \wedge^{(2)}, \vee^{(2)}\} \{q_1\}, \Delta)$ mit
 $\Delta = \{0 \rightarrow q_0, 1 \rightarrow q_1,$
 $\neg(q_0) \rightarrow q_1, \neg(q_1) \rightarrow q_0,$
 $\wedge(q_0, q_0) \rightarrow q_0, \wedge(q_0, q_1) \rightarrow q_0, \wedge(q_1, q_0) \rightarrow q_0, \wedge(q_1, q_1) \rightarrow q_1$
 $\vee(q_0, q_0) \rightarrow q_0, \vee(q_0, q_1) \rightarrow q_1, \vee(q_1, q_0) \rightarrow q_1, \vee(q_1, q_1) \rightarrow q_1\}$

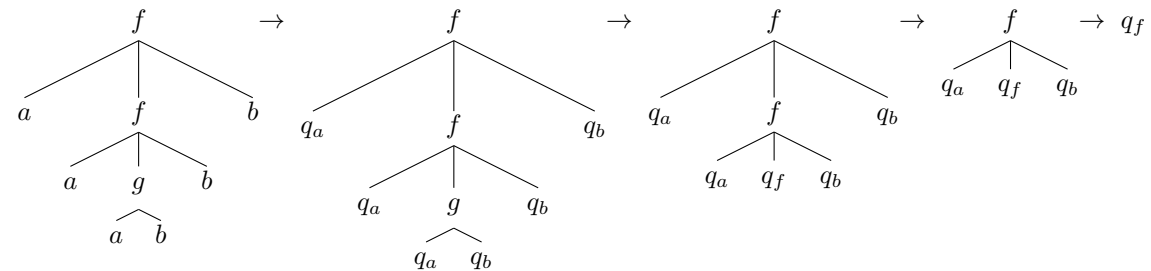
Beispiellauf:



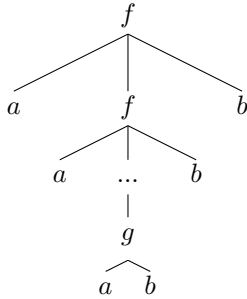
2. ($a^n b^n$ light)

Betrachten $\mathcal{A} = (\{q_a, q_b, q_f\}, \{a^{(0)}, b^{(0)}, f^{(3)}, g^{(2)}\}, \{q_f\}, \Delta)$
mit $\Delta =$
 $\{a \rightarrow q_a, b \rightarrow q_b, g(q_a, q_b) \rightarrow q_f, f(q_a, q_f, q_b) \rightarrow q_f\}$

Beispiellauf:



\mathcal{A} akzeptiert also alle Bäume der Form:



3. Simulation eines Wortautomaten: (siehe Übung)

Betrachtet man $\Sigma = \{a^{(0)}, f^{(2)}, g^{(1)}\}$. Dann ist $L = \{f(g^i(a), g^i(a)) | i \geq 0\}$ nicht erkennbar.

1.10 Definition Determinismus

Ein Automat $\mathcal{A}(Q, \Sigma, F, \Delta)$ heißt deterministisch, falls: aus $f(q_1, \dots, q_n) \rightarrow q$ und $f(q_1, \dots, q_n) \rightarrow q'$ folgt $q = q'$

1.11 Satz

Sei $\mathcal{A} = (Q, \Sigma, F, \Delta)$ ein Baumat, dann existiert ein deterministischer Baumat \mathcal{A}_d , so dass $L(\mathcal{A}) = L(\mathcal{A}_d)$.

Beweis: Setze $\mathcal{A}_d = (Q_d, \Sigma, F_d, \Delta_d)$
mit $Q_d = 2^Q$ (*)
und $f(s_1, \dots, s_n) \rightarrow s \in \Delta_d$
 $\Leftrightarrow s = \{q \in Q | \exists q_1 \in s_1 \dots q_n \in s_n : (f(q_1, \dots, q_n) \rightarrow q) \in \Delta\}$
und $F_d = \{s \in Q_d | s \cap F \neq \emptyset\}$.

Wir zeigen:

1. \mathcal{A} ist deterministisch
2. $L(\mathcal{A}) \subset L(\mathcal{A}_d)$
3. $L(\mathcal{A}_d) \subset L(\mathcal{A})$

1. ist klar, denn (*) ist mit einer Äquivalenz definiert.
2. „ $L(\mathcal{A}) \subseteq L(\mathcal{A}_d)$ “:

Wir zeigen hierzu: Ist $Z = \{q | t \rightarrow_{\mathcal{A}}^* q\}$, dann $t \rightarrow_{\mathcal{A}_d}^* z$.

Induktionsannahme:

Angenommen $a \rightarrow_{\mathcal{A}} q_a$, dann ist $q_a \in \{q \in Q | q \rightarrow_{\mathcal{A}}^* q\}$, das heißt

$a \rightarrow_{\mathcal{A}}^* q_a \Leftrightarrow q_a \in \{q \in Q \mid a \rightarrow_{\mathcal{A}}^* q\}$
 $\Leftrightarrow q_a \in \{q \in Q \mid (a \rightarrow q) \in \Delta\}$
 also $a \rightarrow_{\mathcal{A}}^* q_a \Leftrightarrow q_a \in \{q \in Q \mid (a \rightarrow q) \in \Delta\}$, das heißt
 $z := \{q_a \in Q \mid a \rightarrow_{\mathcal{A}}^* q_a\} = \{q \in Q \mid (a \rightarrow q) \in \Delta\} =: s$

Nun ist $(a \rightarrow s) \in \Delta_d$ per Definition, also auch $(a \rightarrow z) \in \Delta_d$, damit: $a \rightarrow_{\mathcal{A}_d}^* z$.

Betrachten wir nun $t = \sigma(t_1, \dots, t_n)$

Induktionsvoraussetzung:

$t_i \rightarrow_{\mathcal{A}_d}^* z_i$ mit $Z_i = \{q \in Q \mid t_i \rightarrow_{\mathcal{A}}^* q\}$

Das heißt, es existieren Läufe r_i für t_i von \mathcal{A}_d mit $r_i(\varepsilon) = z_i$

Induktionsschritt:

zu zeigen: $t \rightarrow_{\mathcal{A}}^* z$ mit $Z = \{q \in Q \mid t \rightarrow_{\mathcal{A}}^* q\}$

Das heißt, es existiert ein Lauf r für t von \mathcal{A}_d mit $r(\varepsilon) = z$

Das heißt, $\exists r$:

- $r(\varepsilon) = z$
- $r(i) = z_i$
- $\sigma(z_1, \dots, z_n) \rightarrow z \in \Delta_d$

Setze nun $r|_i = r_i$, damit ist insbesondere $r(i) = r_i(\varepsilon) = z := \{q \mid t_i \rightarrow_{\mathcal{A}_d}^* q\}$

Es bleibt also zu zeigen: \exists Regel $\sigma(z_i, \dots, z_m) \rightarrow z \in \Delta_d$.

Es ist nun $z \in Z \Leftrightarrow t \rightarrow_{\mathcal{A}}^* z$

$\Leftrightarrow \exists q_i \in Q : t_i \rightarrow_{\mathcal{A}}^* q_i, \sigma(q_1, \dots, q_m) \rightarrow z \in \Delta$

$\Leftrightarrow \exists z_i \in Z_i$ und $\sigma/z_1, \dots, z_m) \rightarrow z \in \Delta$

Also $Z = \{z \in Q \mid \exists z_i \in Z_i : (\sigma/z_1, \dots, z_m) \rightarrow z) \in \Delta\}$ also per Definition $\sigma/z_1, \dots, z_m) \rightarrow z \in \Delta_d$

2. „ $L(\mathcal{A}_d) \subseteq L(\mathcal{A})$ “:

Sei $t \in T_{\Sigma}$ mit $t \notin L(\mathcal{A})$, dann ist $Z \cap F = \{q \in Q \mid t \rightarrow_{\mathcal{A}}^* q\} \cap F = \emptyset$

Laut 2. ist $t \rightarrow_{\mathcal{A}_d}^* z$ (und \mathcal{A} ist deterministisch) Wegen $Z \cap F = \emptyset$ ist $Z \notin F_d$, also $t \notin L(\mathcal{A}_d)$

Wir vereinbaren die Abkürzungen: NBA/NTA für nichtdeterministischer Baumautomat und DBA/DTA für deterministischer Baumautomat.

Wie im Wortfall ist die Konstruktion exponentiell, das heißt wir benötigen exponentiell viele Zustände ($Q_d = 2^{|Q|}$). Und wie im Wortfall lässt sich das im Allgemeinen nicht vermeiden.

Beispiel: Betrachtet man $\Sigma = \{f^{(1)}, g^{(1)}, a^{(0)}\}$ und sei $L_n = \{f \in T_{\Sigma} \mid t(\underbrace{1 \dots 1}_{n\text{-mal}}) = f\}$

Ein NTA benötigt $n + 2$ Zustände:

$\mathcal{A} = (Q, \Sigma, F, \Delta)$ mit $Q = \{q, q_1, \dots, q_{n+1}\}$, $F = \{q_{n+1}\}$
mit Übergängen $\Delta = \{a \rightarrow q, f(q) \rightarrow q, g(q) \rightarrow q, f(q) \rightarrow q_1,$
 $f(q_i) \rightarrow q_{i+1}, g(q_i) \rightarrow q_{i+1}\}$ für $i \in \{1, \dots, n\}$

Man kann zeigen: Ein DTA \mathcal{A}' mit $L(\mathcal{A}') = L_n$ hat mindestens 2^{n+1} Zustände.

1.12 Definition vollständig und reduziert

Ein Automat $(\mathcal{A} = Q, \Sigma, F, \Delta)$ heißt:

- vollständig, falls für jedes $f^{(n)} \in \Sigma$ und alle $q_1, \dots, q_n \in Q$ eine Regel $f(q_1, \dots, q_n) \rightarrow q \in \Delta$ existiert.
- reduziert, falls für jeden Zustand $q \in Q$ ein Term $t \in T_\Sigma$ existiert mit $f \rightarrow_{\mathcal{A}}^* q$

1.13 Satz

Sei \mathcal{A} ein Baumat. Dann existiert ein vollständiger, reduzierter Baumat \mathcal{A}' mit $L(\mathcal{A}) = L(\mathcal{A}')$.

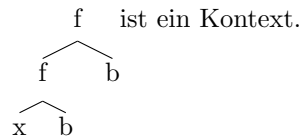
Für Wortautomaten gibt es das Pumping-Lemma, das die Gedächtnislosigkeit der Automaten formalisiert. Formal besagt es: Ist L eine reguläre Wortsprache, dann existiert ein $n \in \mathbb{N}$, so dass sich $w \in L$ mit $|w| > n$ zerlegen lässt in $w = xyz$, $y \neq \varepsilon$ und $\forall i \geq 0$ ist $xy^iz \in L$.

Baumat haben auch kein Gedächtnis, also erwarten wir ein analoges Resultat. Dazu müssen wir formalisieren, was „aufgepumpt“ werden soll.

1.14 Definition Kontext

Es sei Σ ein Rangalphabet und $x^{(0)} \notin \Sigma$. Es sei $C \in T_{\Sigma \cup \{x\}}$. Falls es genau eine Position $p \in \text{Pos}(C)$ gibt mit $C(p) = x$, dann heißt C ein Kontext.

Beispiel:

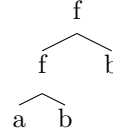


Wir schreiben $T_\Sigma(x)$ für die Menge aller solcher Kontexte.

Ist $C \in T_\Sigma(x)$ mit $C(p) = x$, dann schreiben wir $C[u]$ statt $C[u]_p$ für den Baum, der entsteht, wenn wir x durch u ersetzen.

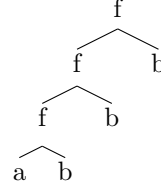
Wir schreiben $C^0 = x$, $C^1 = C$, $C^n = C^{n-1}[C]$

Beispiel: Betrachtet $t =$



Setze $u = f(a, b)$ und $C = f(x, b)$.

Dann ist $t = C[u]$ und $C^2[u] =$



1.15 Pumping-Lemma

Sei $L \subseteq t_\Sigma$ erkennbar, dann existiert ein $k \in \mathbb{N}$, so dass:

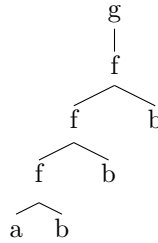
Für alle $T \in L$ mit $ht(t) > k$ gibt es einen Kontext $C \in T_\Sigma(x)$, einen nicht-trivialen Kontext $C' \in T_\Sigma(x)$ und einen Term $u \in T_\Sigma$ mit $t = C[C'[u]]$ und $C[(C')^n[u]] \in L$ für alle $n \geq 0$.

Beweis: Sei L erkennbar, das heißt \exists Baumentautomat $\mathcal{A} = (Q, \Sigma, F, \Delta)$ mit $L = L(\mathcal{A})$. Setze $|Q| = k$ und betrachte $t \in L$ mit $ht(t) > k$. Betrachte nun einen Lauf r und einen Pfad in t , der länger als k ist. Nun gibt es $p_1, p_2 \in Pos(r)$ mit $r(p_1) = r(p_2) = q \in Q$. Sei nun $u = t|_{p_2}$ der Sub-Baum von t bei p_2 und $u' = t|_{p_1}$. Dann existiert C' mit $C'[u] = u'$ und es existiert C mit $t = C[C'[u]]$. Es ist wegen $t \in L$

$C[C'[u]] \rightarrow_{\mathcal{A}}^* C[C'[q]] \rightarrow_{\mathcal{A}}^* C[q] \rightarrow_{\mathcal{A}}^* q_f \in F$, also auch

$C[(C')^n[u]] \rightarrow_{\mathcal{A}}^* C[(C')^n[q]] \rightarrow_{\mathcal{A}}^* C C[(C')^{(n-1)}[q]] \rightarrow_{\mathcal{A}}^* \dots \rightarrow_{\mathcal{A}}^* C[q] \rightarrow_{\mathcal{A}}^* q_f \in F$. q.e.d.

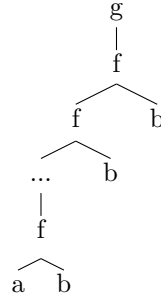
Beispiel: Betrachte den Baumentautomaten $\mathcal{A} = (\{q_a, q_b, q_g, q_f\}, \{a^{(0)}, b^{(0)}, g^{(1)}, f^{(2)}\}, \{q_f\}, \Delta)$ mit $\Delta = \{a \rightarrow q_a, b \rightarrow q_b, f(q_a, q_b) \rightarrow q_g, g(q_g) \rightarrow q_f\}$



$u = f(a, b)$, $u' = C'[u] = f(f(a, b), b)$

$C = g(f(x, b))$, $C' = f(x, b)$

$$C[(C')^n[u]] =$$



Die Sprache $L = \{f(g^i(a), g^i(a)) \mid i \geq 0\}$ kann nicht erkennbar sein, denn für große i würde man ein k finden, so dass ein gegebener Baumatomat auch $f(g^{i+lk}(a), g^i(a))$ für alle $l \geq 0$ akzeptiert.

1.16 Korollar

Für $\mathcal{A} = (Q, \Sigma, F, \Delta)$ ist $L(\mathcal{A}) \neq \emptyset \Leftrightarrow \exists t \in L$ mit $ht(t) \leq |Q|$:

- $|L(\mathcal{A})|$ nicht endlich $\Leftrightarrow \exists t \in L$ mit $|Q| < ht(t) \leq 2|Q|$

1.17 Abschlusseigenschaften

Erkennbare Sprachen sind abgeschlossen unter Vereinigung, Schnitt und Komplement. Das heißt, sind L_1 und L_2 erkennbar, dann auch $L_1 \cup L_2$, $L_1 \cap L_2$ und L_1^c (in T_Σ).

Beweis:

Seien \mathcal{A}_1 und \mathcal{A}_2 vollständige DTA. Betrachte für die Vereinigung

$\mathcal{A}_\cup = (Q_1 \times Q_2, \Sigma, F_1 \times Q_2 \cup Q_1 \times F_2, \Delta_1 \times \Delta_2)$ mit

$\Delta_1 \times \Delta_2 = \{f((q_1, q'_1), \dots, (q_n, q'_n)) \rightarrow (q, q') \mid f(q_1, \dots, q_n) \rightarrow q \in \Delta_1, f(q'_1, \dots, q'_n) \rightarrow q' \in \Delta_2\}$

Dann akzeptiert \mathcal{A}_\cup die Sprache $L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$.

Für $L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$ betrachte den Automaten

$\mathcal{A}_\cap = (Q_1 \times Q_2, \Sigma, F_1 \times F_2, \Delta_1 \times \Delta_2)$

Für T_Σ $L(\mathcal{A}_1) = L(\mathcal{A}_1)^c$ betrachte

$\mathcal{A}_C = (Q_1, \Sigma, Q_1, F_1, \Delta_1)$.

Der Automat \mathcal{A}_C akzeptiert $L(\mathcal{A}_1)^c$.

Beispiel:

Betrachte $\Sigma = \{a^{(0)}, g^{(1)}, f^{(2)}\}$ und $L = \{f(g^i(a), g^j(a)) \mid i \leq j\}$

Dann ist L nicht erkennbar, denn: Wäre L erkennbar, dann auch L' mit $L' = \{f(g^i(a), g^j(a)) \mid i \geq j\}$, also auch $L \cap L' \not\subseteq$.

Bemerkung: Wenn \mathcal{A} deterministisch und vollständig ist, dann können wir eine Übergangsfunktion $\delta : T_\Sigma \rightarrow Q$ definieren mit $\delta(t) = q$, falls $t \rightarrow_{\mathcal{A}}^* q$.

Wiederholung - Äquivalenzrelation:

Eine Äquivalenzrelation \sim auf einer Menge M ist eine Relation mit

- $\forall m \in M : m \sim m$
- $\forall m, n \in M : m \sim n \Rightarrow n \sim m$
- $\forall l, m, n \in M : l \sim m, m \sim n \Rightarrow l \sim n$

Insbesondere: Ist \sim eine Äquivalenzrelation auf M , so induziert \sim eine Partition auf und umgekehrt, das heißt Mengen $(M_i)_{i \in I}$ mit $M_i \cup M_j = \emptyset$ für $i \neq j$ und $M = \bigcup_{i \in I} M_i$

1.18 Definition Kongruenz

Eine Äquivalenzrelation \equiv auf T_Σ heißt Kongruenz, falls für alle $f^{(n)} \in \Sigma$:

$$v_1 \equiv u_1, \dots, v_n \equiv u_n \Rightarrow f(v_1, \dots, v_n) \equiv f(u_1, \dots, u_n).$$

Beispiel:

Die Relation $t \equiv t'$, falls t und t' die gleiche Anzahl Blätter modulo 2 haben.

- Außerdem: $t \equiv t' \Leftrightarrow ht(t) = ht(t')$
- Nicht: gleiche Höhe modulo 2

1.19 Definition

Eine Kongruenz \equiv hat endlichen Index, falls \equiv endlich viele Äquivalenzklassen indiziert.

1.20 Lemma

Sei Σ ein Rangalphabet. Dann ist \equiv genau dann eine Kongruenz auf T_Σ , wenn \equiv eine Äquivalenzrelation ist mit $u \equiv v \Rightarrow C[u] \equiv C[v]$ für alle Kontexte.

Beweis:

„ \Rightarrow “ Induktion:

Induktionsannahme: $C = x$, dann ist $u \equiv v \Rightarrow C[u] \equiv C[v]$ klar.

Sei nun $C = f(C_1, \dots, C_n)$. Sei $x = C[ip] = C_i[p]$.

Dann ist $C[u]_{ip} = f(C_1, \dots, C_{i-1}, C_i[p], C_{i+1}, \dots, C_n) = C[u]_{ip} = f(C_1, \dots, C_{i-1}, C_i[v], C_{i+1}, \dots, C_n)$

„ \Leftarrow “: Angenommen $u \equiv v$ und $C[u] \equiv C[v]$ für alle Kontexte.

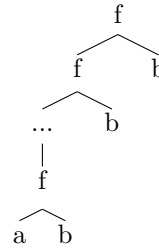
Sei $f^{(n)} \in \Sigma$. Dann ist:

$$\begin{aligned} & f(u_1, \dots, u_n) \\ &= C^1[u_1] \equiv C^1[v_1] = f(v_1, u_2, \dots, u_n) \\ &= \dots \\ &= C^1[u_n] \equiv C^1[v_n] = f(v_1, \dots, v_n) \end{aligned}$$

Betrachte nun eine Sprache $L \subseteq T_\Sigma$ von Bäumen. Wir definieren \equiv_L als:

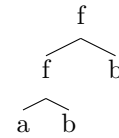
$$u \equiv_L v \Leftrightarrow \forall C \in T_\Sigma(x) : C[u] \in L \Leftrightarrow C[v] \in L.$$

Beispiel: Betrachte alle Bäume der Form

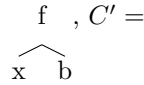


$$L = \{f(f(\dots f(a, b), b) \dots, b)\}$$

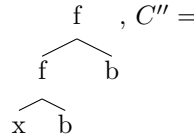
Dann gilt:



$C =$



, $C' =$



, $C'' =$



1.21 Theorem (Myhill-Nerode)

Die folgenden Aussagen sind äquivalent:

- a) L ist erkennbar
- b) L ist die Vereinigung von Äquivalenzklassen einer Kongruenz mit endlichem Index
- c) \equiv_L hat endlichen Index

Beweis:

„a \Rightarrow b“: Sei \mathcal{A} vollständiger DTA mit $L(\mathcal{A}) = L$. Sei $\mathcal{A} = (Q, \Sigma, F, \Delta)$.

Definiere $u \equiv_{\mathcal{A}} v \Leftrightarrow \delta(u) = \delta(v)$.

Offensichtlich hat $\equiv_{\mathcal{A}}$ höchstens $|Q|$ -viele Äquivalenzklassen. Außerdem ist $\equiv_{\mathcal{A}}$ eine Kongruenz.

Nun ist L Vereinigung aller Klassen $[u]_{\equiv_{\mathcal{A}}}$ mit $\delta(u) \in F$.

„b \Rightarrow c“: Sei \sim eine Kongruenz mit endlichem Index. Sei $u \sim v$. Wegen Lemma 1.20 gilt

$C[u] \sim C[v] \forall C \in T_{\Sigma}(x)$. Nun ist L die Vereinigung von Äquivalenzklassen von \sim , das heißt

$C[u] \in L \Leftrightarrow C[v] \in L$. Insbesondere ist also $u \equiv_L v$

Wir haben gezeigt: $v \in [u]_{\sim} \Rightarrow v \in [u]_{\equiv_L}$, also $[u]_{\sim} \leq [u]_{\equiv_L}$

(Also ist \sim eine Verfeinerung von \equiv_L)

Insbesondere hat \equiv_L kleineren Index als \sim , also endlichen.

„c \Rightarrow a“: Die Zustände Q_{\min} sind die Äquivalenzklassen bezüglich \equiv_L . (Damit ist Q_{\min} endlich).

Wir definieren Regeln

$f([u_1], \dots, [u_n]) \rightarrow [f(u_1, \dots, u_n)]$.

Das ist wohldefiniert, weil \equiv_L eine Kongruenz ist. Finalzustände F_{\min} sind $\{[u]_{\equiv_L} \mid u \in L\}$.

Dann akzeptiert $\mathcal{A}_{\min} = (Q_{\min}, \Sigma, F_{\min}, \Delta_{\min})$ die Sprache L .

Beispiel:

Betrachte $\Sigma = \{a^{(0)}, g^{(1)}, f^{(2)}\}$ und

$L = \{f(g^i(a), g^i(a)) \mid i \geq 0\}$

Betrachte $g^i(a)$ und $g^j(a)$ mit $i \neq j$. Dann ist $C^i = f(x, g^i(a))$ ein Kontext mit $C^i[g^i(a)] \in L$, aber $C^i[g^j(a)] \notin L$. Da es unendlich viele $g^i(a)$ gibt, hat die Kongruenz bezüglich L unendlichen Index, also ist L nicht erkennbar.

1.22 Korollar

Ist L erkennbar, gibt es einen bis auf Umbenennung der Zustände eindeutigen, vollständigen DBA \mathcal{A} mit $L = L(\mathcal{A})$. Dieser ist \mathcal{A}_{\min} aus obigem Beweis.

Beweis:

Sei $L = L(\mathcal{A})$. Vorher gesehen:

$\equiv_{\mathcal{A}}$ ist Verfeinerung von \equiv_L

Also ist $|Q| \geq |Q_{\min}|$. Wir nehmen OBDA an: beide reduziert. Sei nun $q \in Q$. Betrachte ein $u \in T_{\Sigma}$ mit $\delta(u) = q$. Betrachte die Funktion $\rho : Q \rightarrow Q_{\min}$ mit $\delta(u) = q \mapsto \delta_{\min}(u)$

Die Abbildung ρ ist wohldefiniert, denn falls $\delta(u) = \delta(v)$, dann $u \equiv_{\mathcal{A}} v \Rightarrow u \equiv_L v \Leftrightarrow \delta_{\min}(u) = \delta_{\min}(v)$. Außerdem ist ρ surjektiv, denn $\delta_{\min}(u)$ hat das Urbild $\delta(u)$.

Also: $|Q| = |Q_{\min}| \Rightarrow \rho$ ist Bijektion. \square

1.23 Einschub - Homomorphismen von Baumsprachen

1.23.1 Allgemeine Homomorphismen

$(M, \cdot), (N, *); h : M \rightarrow N$ heißt Homomorphismus, falls

$\forall m, \hat{m} : h(m \cdot \hat{m}) = h(m) * h(\hat{m})$

1.23.2 Worthomomorphismen

$(A^*, \cdot), (B^*, \cdot); h : A^* \rightarrow B^*$ ist Homomorphismus, falls $h(w \cdot \hat{w}) = h(w) \cdot h(\hat{w})$. (zusätzlich $h(\varepsilon) = \varepsilon$)

Nutze für die Definition des Homomorphismus die induktive Definition von Wörtern aus A^* .

0.) $\varepsilon \in A^*$

1.) $a \in A^* \forall a \in A$

2.) $a \cdot w \in A^* \forall a \in A, w \in A^*$

Ein Wort-Homomorphismus entsteht deshalb aus einer Abbildung $\bar{h} : A \rightarrow B$ wie folgt:

0.) $h(\varepsilon) = \varepsilon$

1.) $h(a) = \bar{h}(a) \forall a \in A$

2.) $h(a \cdot w) = h(a) \cdot h(w) = \bar{h}(a) \cdot h(w) \forall a \in A, w \in A^*$

1.23.3 Baumhomomorphismen

T_Σ :

- 1.) $a \in T_\Sigma \ \forall a^{(0)} \in \Sigma$
- 2.) $f(t_1, \dots, t_n) \in T_\Sigma \ \forall f^{(n)} \in T_\Sigma$

Zunächst: Schreibe $\Sigma = \bigcup_{n=0}^r \Sigma^{(n)}$, wobei $\Sigma^{(n)} = rk^{-1}(n)$.

Sei $X = \{x_1, \dots, x_n\}$ und $X_n = \{x_1, \dots, x_n\}, X_0 = \emptyset$

Dann ist für ein Rangalphabet Γ auch $\Gamma \cup X_n$ ein Rangalphabet mit $rk(x_i) = 0$

Nachtrag - Substitution:

$t, s \in T_{\Sigma \cup X_n}$ (für ein $n \in \mathbb{N}$)

Sei $P \subseteq pos(t), P = \{p \in pos(t) | t(p) = x_i\}$ für ein $i \in \{1, \dots, n\}$

Etwa $P = \{p_1, \dots, p_m\}$

Dann ist $t_{[x_i \leftarrow s]} = t[s]_{p_1} \cdot \dots \cdot t[s]_{p_m}$.

Definition von Homomorphismen für jeden Rang $n = 0, \dots, r$

Wähle eine Funktion $\bar{h}_n : \Sigma^{(n)} \rightarrow T_{\Gamma \cup X_n}$

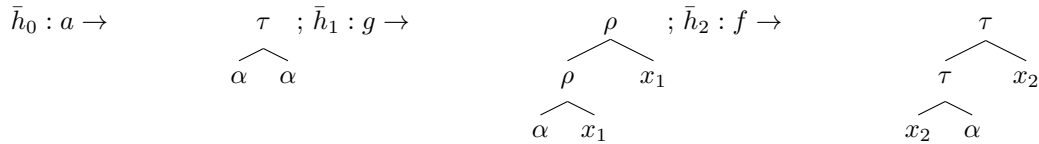
Der von $\bar{h}_0, \dots, \bar{h}_r$ erzeugte Homomorphismus $h : T_\Sigma \rightarrow T_\Gamma$

- 1.) $h(a) = \bar{h}_0(a) \ \forall a \in \Sigma$
- 2.) $h(f(t_1, \dots, t_n)) = \bar{h}_n(f)[x_1 \leftarrow h(t_1)] \dots [x_n \leftarrow h(t_n)]$

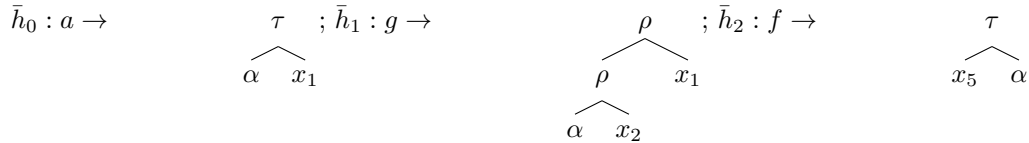
Beispiel:

$\Sigma = \{a^{(0)}, g^{(1)}, f^{(2)}\}$

$\Gamma = \{\alpha^{(0)}, \delta^{(2)}, \tau^{(2)}\}$

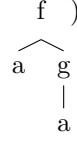


Gegenbeispiel:



Erzeugter Homomorphismus $h : T_\Sigma \rightarrow T_\Gamma$ wie oben

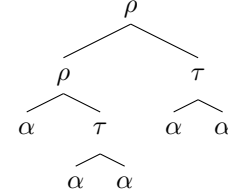
$h($



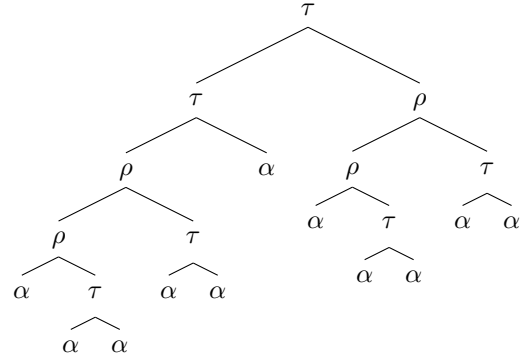
$h(a) = \bar{h}_0(a) =$



$h(g(a)) = \bar{h}_1(g)[x_1 \leftarrow h(a)] =$



$h(f(a, g(a))) = \bar{h}_2(f)[x_1 \leftarrow h(a)][x_2 \leftarrow h(g(a))]$



1.23.4 lineare Terme

Ein Term $t \in T_{\Sigma \cup X_n}$ heißt linear, falls jede Variable höchstens einmal vorkommt, d.h. falls $\forall i \in \{1, \dots, n\} : |\{p \in \text{pos}(t) \mid t(p) = x_i\}| \leq 1$

1.23.5 linearer Homomorphismus

Ein Homomorphismen $h : T_{\Sigma} \rightarrow T_{\Gamma}$ erzeugt von $\bar{h}_0, \dots, \bar{h}_r$ heißt linear, falls $\forall f^{(n)} \in T_{\Sigma}$ gilt: $\bar{h}_n(f) \in T_{\Gamma \cup X_n}$ ist linear.

Beispiel: im Allgemeinen erhalten Homomorphismen die Erkennbarkeit nicht.

Betrachte

$$\Sigma = \{f^{(1)}, g^{(1)}, a^{(0)}\}, \Gamma = \{\hat{f}^{(2)}, g^{(1)}, a^{(0)}\}$$

$$\bar{h}_0 : a \rightarrow a, \bar{h}_1 : g \rightarrow g(x_1), \bar{h}_1 : f \rightarrow \hat{f}(x_1, x_1)$$

Für $L \subseteq T_{\Sigma}, L = \{f(g^i(a)) \mid i \in \mathbb{N}\}$ ist $h(L) = \{\hat{f}(g^i(a), g^i(a)) \mid i \in \mathbb{N}\}$ nicht erkennbar, obwohl L erkennbar ist.

1.23.6 Satz

Sei $L \subseteq T_\Sigma$ erkennbar, $h : T_\Sigma \rightarrow T_\Gamma$ ein linearer Homomorphismen, dann ist $h(L) \subseteq T_\Gamma$ erkennbar.

Beweisskizze:

Sei $A = (Q, \Sigma, F, \Delta)$ ein reduzierter DFTA mit $L(A) = L$

- Seien $(\bar{h}_n)_{n=0}^r$ die erzeugenden Funktionen von h
- Definiere NFTA $A' = (Q', \Gamma, F', \Delta)$ wie folgt:
für jede Regel $\rho : f(q_1, \dots, q_n) \rightarrow q \in \Delta$:
Setze $Q^\rho = \{q_p^\rho \mid q \in \text{pos}(\bar{h}_n(f))\}$ und
 Δ^ρ für jedes $p \in \text{pos}(\bar{h}_n(f))$
falls $(\bar{h}_n f)(p) = g^{(k)} \in \Gamma^{(k)}$ für ein k ,
 $g(q_{p-1}^\rho, \dots, q_{p-k}^\rho) \rightarrow q_p \in \Delta^\rho$
falls $(\bar{h}_n f)(g) = x_i$ für ein i
 $q_i \rightarrow q_p^\rho \in \Delta^\rho$
 $q_\varepsilon^\rho \rightarrow q \in \Delta^\rho$
OBdA Q^ρ paarweise disjunkt auch mit Q
- $Q' = \bigcup_{\rho \in \Delta} Q^\rho \cup Q$
- $\Delta = \bigcup_{\rho \in \Delta} \Delta^\rho \cup Q$
- $F' = F$

1.23.7 Satz

Sei $h : T_\Sigma \rightarrow T_\Gamma$ beliebiger Homomorphismus und $L \subseteq T_\Gamma$ erkennbar. Dann ist auch $h^{-1}(L) \subseteq T_\Sigma$ erkennbar.

Beweis:

Sei $A' = (Q', \Gamma, F', \Delta')$ ein vollständiger DFTA mit $L(A') = L$.

Definiere $A = (Q, \Sigma, F, \Delta)$ wie folgt:

$$Q = Q', F = F'$$

$$f(q_1, \dots, q_n) \rightarrow q \in \Delta$$

$$\Leftrightarrow \bar{h}_n(f)[x_1 \leftarrow q_1] \cdots [x_n \leftarrow q_n] \rightarrow_{\mathcal{A}'}^* q$$

Beweis der Korrektheit (strukturelle Induktion):

Zeige die stärkere Aussage: für $q \in Q = Q'$ gilt

$$t \rightarrow_{\mathcal{A}}^* q \Leftrightarrow h(t) \rightarrow_{\mathcal{A}'}^* q$$

$$(h^{-1}(s) \rightarrow_{\mathcal{A}}^* q \Leftrightarrow s \rightarrow_{\mathcal{A}'}^* q)$$

Induktionsannahme:

Sei $t = a \in \Sigma^{(0)}$. Dann gilt $h(a) \rightarrow_{\mathcal{A}'}^* q \Leftrightarrow \bar{h}_0(a) \rightarrow_{\mathcal{A}'}^* q \Leftrightarrow a \rightarrow q \in \Delta \Leftrightarrow a \rightarrow_{\mathcal{A}}^* q$

Induktionsvoraussetzung:

Die Aussage gelte für Terme mit Höhe $\leq k$

Induktionsschritt: Dann gilt sie auf für t mit Höhe $k + 1$.

Sei $t = f(t_1, \dots, t_n)$ mit $ht(t_i) \leq k \ \forall i \in \{1, \dots, n\}$

$$\begin{aligned}
t \rightarrow_{\mathcal{A}}^* q &\Leftrightarrow \exists q_1, \dots, q_n : t_i \rightarrow_{\mathcal{A}}^* q_i \text{ und } f(q_1, \dots, q_n) \rightarrow q \in \Delta \\
&\Leftrightarrow \exists q_1, \dots, q_n : h(t_i) \rightarrow_{\mathcal{A}'}^* q_i \text{ und } f(q_1, \dots, q_n) \rightarrow q \in \Delta \\
&\Leftrightarrow \exists q_1, \dots, q_n : h(t_i) \rightarrow_{\mathcal{A}'}^* q_i \text{ und } \bar{h}_n(f)[x \leftarrow q_i] \cdot \dots \cdot [x_n \leftarrow q_n] \rightarrow_{\mathcal{A}'}^* q \\
&\Leftrightarrow \bar{h}_n(f)[x_1 \leftarrow h(t_1)] \cdot \dots \cdot [x_n \leftarrow h(t_n)] \rightarrow_{\mathcal{A}'}^* q \\
&\Leftrightarrow h(f(t_1, \dots, t_n)) (= h(t)) \rightarrow_{\mathcal{A}'}^* q
\end{aligned}$$

1.24 Top-Down Baautomaten

Bisher: Bottom-Up TA - laufen Bäume von den Blättern zu der Wurzel nach oben.

Nun: Top-Down TA - umgekehrt

Definition:

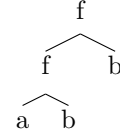
Ein nicht-deterministischer Top-Down Baautomat ist ein Tupel $\mathcal{A} = (Q, \Sigma, I, \Delta)$, wobei:

- Q eine endliche Menge Zustände
- Σ ein Rangalphabet
- $I \subseteq Q$ Initialzustände
- Δ eine endliche Menge Regeln der Form
 $q(f(x_1, \dots, x_n)) \rightarrow f(q_1(x_1), \dots, q_n(x_n))$
bzw. für $n = 0$: $q(a) \rightarrow a$

ist.

Beispiel:

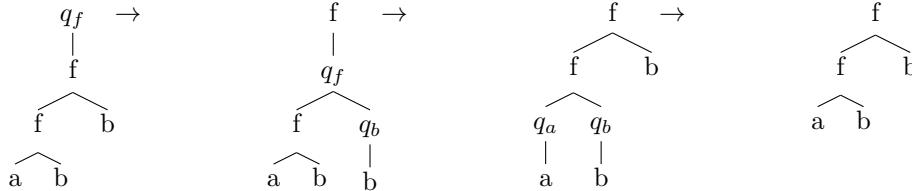
Wir betrachten wieder $\Sigma = \{a^{(0)}, b^{(0)}, f^{(2)}\}$ und $t =$



Setze

$$Q = \{q_f(f(x_1, x_2)) \rightarrow f(q_f(x_1), q_b(x_2)), q_f(f(x_1, x_2)) \rightarrow f(q_a(x_1), q_b(x_2)), q_a(a) \rightarrow a, q_b(b) \rightarrow b\}$$

Run für t (intuitiv):



Betrachte folgende Übergangsrelation:

$C[q(f(t_1, \dots, t_n))] \rightarrow C[f(q_1(t_1), \dots, q_n(t_n))], C \in T_{\Sigma \cup Q}(x),$
 falls $q(f(x_1, \dots, x_n)) \rightarrow f(q_1(x_1), \dots, q_n(x_n)) \in \Delta$

\rightarrow^* transitive Hülle

$L(\mathcal{A}) = \{t \in T_\Sigma | q(t) \rightarrow^*, q \in I\}$

1.25 Satz

Eine Sprache $L \subseteq T_\Sigma$ ist genau dann erkennbar, wenn es einen nichtdeterministischen Top-Down TA \mathcal{A} gibt mit $L = L(\mathcal{A})$.

Bemerkung:

Top-Down Automaten sind nicht determinisierbar. Deterministische Top-Down TA erkennen „path closed“ Sprachen.

2 Grammatiken

2.1 Definition - Grammatik

Eine Grammatik ist ein Tupel $G = (S, N, \Sigma, R)$. Dabei ist:

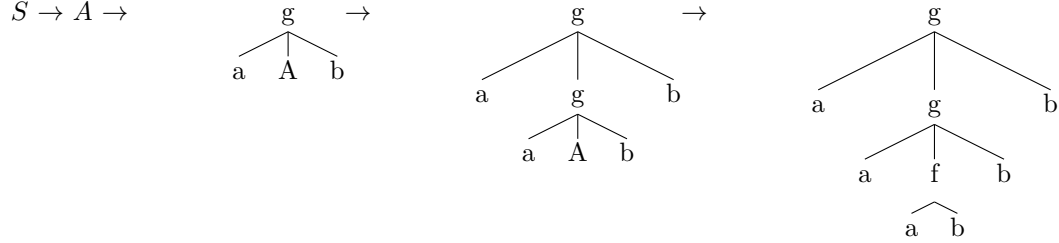
- S Startsymbol ($S = S^{(0)}, S \in N$)
- N (Rangalphabet) nichtterminale Symbole
- Σ (Rangalphabet) terminale Symbole ($\Sigma \cap N = \emptyset$)
- R Regeln der Form $\alpha \rightarrow \beta$ mit:
 $\alpha, \beta \in T_{\Sigma \cup N \cup X}$ ($X \cap (\Sigma \cup N) = \emptyset$ Variablen),
 α enthält mindestens ein Nichtterminal-Symbol

Eine reguläre Grammatik enthält nur Regeln der Form $A \rightarrow B$, wobei A den Rang 0 hat und $B \in T_{\Sigma \cup N}$. Insbesondere enthält N nur Symbole mit Rang 0.

Beispiel: Betrachte $G = (S, \{S, A\}, \{a^{(0)}, b^{(0)}, f^{(2)}\}, R)$ mit
 $R = \{S \rightarrow f(A, b), A \rightarrow a, A \rightarrow f(A, b)\}$
 zum Beispiel haben wir:



Beispiel: Betrachte $G = (S, \{S, A\}, \{a^{(0)}, g^{(3)}, f^{(2)}\}, R)$ mit
 $R = \{S \rightarrow A, A \rightarrow f(a, b), A \rightarrow g(a, A, b)\}$
zum Beispiel:



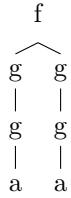
Bemerkung - kontextfreie Baumgrammatik:

$F(x_1, \dots, x_n) \rightarrow t, t \in T_{\Sigma \cup N \cup \{x_1, \dots, x_n\}}$

z.B. $S \rightarrow F(a, a), F(x, x) \rightarrow F(G(x), G(x)), F(x, x) \rightarrow f(x, x), G(x) \rightarrow g(x)$

Damit können wir erzeugen:

$S \rightarrow F(a, a) \rightarrow F(G(a), G(a)) \rightarrow F(G(G(a)), G(G(a))) \rightarrow \dots \rightarrow f(g(g(a)), g(g(a)))$



Betrachte nun folgende Ableitungsrelation für reguläre Grammatiken:

Wir schreiben $s \rightarrow_G$ genau dann wenn ein Kontext $C \in T_{\Sigma \cup N}(x)$ existiert, sodass

$s = C[A], t = C[\alpha], A \rightarrow \alpha \in R$ für $G = (S, N, \Sigma, R)$.

Mit \rightarrow_G^* bezeichnen wir die transitive Hülle von \rightarrow_G .

2.2 Definition

Ist G eine reguläre Grammatik, dann heißt $L(G) = \{t \in T_\Sigma \mid S \rightarrow_G^* t\}$ die von G akzeptierte Sprache.

Eine Sprache $L \subseteq T_\Sigma$ heißt regulär, falls $L = L(G)$ für eine reguläre Grammatik.

Betrachte reguläre Grammatik $G = (S, N, \Sigma, R)$ Wir bezeichnen mit $L_G(A)$, $A \in N$, die von G erzeugte Sprache mit A als Startsymbol.

2.3 Definition - reduziert

Sei $G = (S, N, \Sigma, R)$ eine reguläre Grammatik und $A \in N$. Dann heißt A

- erreichbar, falls ein Kontext $C \in T_\Sigma(x)$ existiert, so dass $S \rightarrow_G^* C[A]$
- produktiv, falls $L_G(A) \neq \emptyset$

G heißt reduziert, falls alle $A \in N$ erreichbar und produktiv sind.

2.4 Satz

Ist G eine reguläre Grammatik mit $L(G) = L$, dann existiert eine reduzierte reguläre Grammatik G' mit $L(G') = L$.

Nun: zu G eine „Normalform“ konstruieren.

2.5 Definition - Normalisierung

Eine reguläre Grammatik G heißt normalisiert, falls alle Regeln aus R die Form

- $A \rightarrow f(A_1, \dots, A_n), A, A_1, \dots, A_n \in N, v \in \Sigma^{(n)}$
- $A \rightarrow a, A \in N, a \in \Sigma$

2.6 Satz

Ist G eine reguläre Grammatik mit $L(G) = L$, dann existiert eine normalisierte reguläre Grammatik G' mit $L(G') = L$.

Beweis: Wir ersetzen Regeln der Form $A \rightarrow f(s_1, \dots, s_n)$ durch $A \rightarrow f(A_1, \dots, A_n)$ wobei: Ist $s_i \in N$, dann ist $A_i = s_i$, ansonsten ist A_i ein neues Symbol und wir fügen $A_i \rightarrow s_i$ hinzu.
...Iterieren...

Es bleiben übrig:

$A \rightarrow f(A_1, \dots, A_n), A \rightarrow a \in \Sigma, A_i \rightarrow A_j$ (letztere überbrücken)

Wir erhalten nun:

2.7 Theorem

L ist erkennbar $\Leftrightarrow L$ ist regulär.

Beweis:

„ \Rightarrow “ Ist L erkennbar, so existiert ein nicht-deterministischer Top-Down-TA $\mathcal{A} = (Q, \Sigma, I, \Delta)$ mit $L = L(\mathcal{A})$. Betrachte eine Grammatik $G = (S, N, \Sigma, R)$ mit

- S ist ein neues Symbol
- $N = \{A_q | q \in Q\}$
- $R = \{A_q \rightarrow f(A_{q_1}, \dots, A_{q_n}) | q(f(x_1, \dots, x_n)) \rightarrow f(q_1(x_1), \dots, q_n(x_n)) \in \Delta\} \cup \{S \rightarrow A_{q_i} | q_i \in I\}$

Offensichtlich ist $L(G) = L(\mathcal{A})$

„ \Leftarrow “ analog (gehe von normalisierter Grammatik aus) \square

Ziel: Definieren Konkatination und Kleene-Stern

Problem hierbei ist: Wir müssen erklären, wie wir Bäume zusammensetzen.

Dazu definieren wir: Substitution von Sprachen.

Betrachte $t \in T_{\Sigma \cup k}$, wobei $k = \{\square_1, \dots, \square_n\}$, \square_i sind Konstanten.

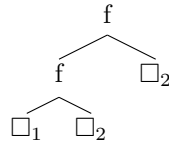
Es seien $L_i \subseteq T_{\Sigma \cup k}$ Sprachen. Dann ist die Substitution $t\{\square_1 \leftarrow L_1, \dots, \square_n \leftarrow L_n\}$ induktiv wie folgt definiert:

- $\square_i\{\square_1 \leftarrow L_1, \dots, \square_n \leftarrow L_n\} = L_i$
- $a\{\square_1 \leftarrow L_1, \dots, \square_n \leftarrow L_n\} = \{a\}$, $a \neq \square_i$, $a \in \Sigma^0$
- $f(s_1, \dots, s_n)\{\square_1 \leftarrow L_1, \dots, \square_n \leftarrow L_n\} = \{f(t_1, \dots, t_n) \mid t_i \in s_i\{\square_1 \leftarrow L_1, \dots, \square_n \leftarrow L_n\}\}$

Außerdem setzen $L\{\square_1 \leftarrow L_1, \dots, \square_n \leftarrow L_n\} = \bigcup_{t \in L} t\{\square_1 \leftarrow L_1, \dots, \square_n \leftarrow L_n\}$

Beispiel: Betrachte $k = \{\square_1, \square_2\}$

$t =$



$L_2 = \{a, b\}$

Dann ist $t\{\square_2 \leftarrow L_2\} = \{$

$\}$

Nun setzen wir für zwei Sprachen L, M :

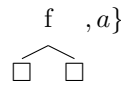
$$L \cdot_{\square} M = \bigcup_{t \in L} t\{\square \leftarrow M\}$$

Des weiteren ergibt sich der Kleene-Stern:

- $L^{0.\square} = \{\square\}$
- $L^{n+1.\square} = L^{n.\square} \cup L \cdot_{\square} L^{n.\square}$

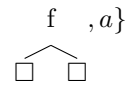
$$\Rightarrow L^{*.\square} = \bigcup_{n \geq L^{n.\square}}$$

Beispiel: Betrachte $L = \{$



$$L^{0.\square} = \{\square\}$$

$$L^{1.\square} = \{\square\} \cup L \cdot_{\square} \{\square\} = \{\square\} \cup \{$$



$$L^{2.\square} = \{\square,$$



$$\cup L \cdot_{\square} \{\square,$$



$$f, a\} = \dots$$

Abschlusseigenschaften:

2.8 Satz

Es sei $L \subseteq T_{\Sigma \cup k}$ regulär, sowie $\square_1, \dots, \square_n \in k$. Dann ist $L\{\square_1 \leftarrow L_1, \dots, \square_n \leftarrow L_n\}$ regulär.

Beweis:

Betrachte normalisierte Grammatiken G, G_1, \dots, G_n mit $L(G) = L, L(G_1) = L_1, \dots, L(G_n) = L_n$, $G = (S, N, \Sigma \cup k, R), G_i = (S_i, N_i, \Sigma \cup k, R_i)$. (alle Nichtterminale paarweise disjunkt)

Konstruiere $G' = (S, N', \Sigma \cup k, R')$ mit:

- $N' = N \cup N_1 \cup \dots \cup N_n$
- R' enthält alle Regeln in R, R_1, \dots, R_n , wobei $A \rightarrow \square_i$ ersetzt werden durch $A \rightarrow S_i$

Direkt zeigen: $L\{\square_1 \leftarrow L_1, \dots, \square_n \leftarrow L_n\} \subseteq L(G')$

Wir zeigen „ \supseteq “

Induktion: über die Anzahl der Ableitungsschritte zeigen wir $A \rightarrow_{G'}^* s'$ mit $S \in T_{\Sigma \cup k}$ — s' enthält keine Nichtterminale

$\exists s$ mit $A_{G'}^*$ und $s' \in s\{\square_1 \leftarrow L_1, \dots, \square_n \leftarrow L_n\}$ — das heißt $s' \in L_G(A)\{\square_1 \leftarrow L_1, \dots, \square_n \leftarrow L_n\}$

Induktionsannahme: Angenommen $A \rightarrow_G^*$ in einem Schritt, das heißt $s' \in L$, die Regel kann nicht $A \rightarrow \square_i$ sein (existiert nicht in G') und auch nicht $A \rightarrow s_i$ (kein $s' \in T_{\Sigma \cup k}$). Damit: $S' \in L$. Seze $s = s'$, damit enthält s kein \square_i , also $\{s'\} = \{s\} = s\{\square_1 \leftarrow L_1, \dots, \square_n \leftarrow L_n\}$, also insbesondere $s' \in s\{\square_1 \leftarrow L_1, \dots, \square_n \leftarrow L_n\} \subseteq L(A)\{\square_1 \leftarrow L_1, \dots, \square_n \leftarrow L_n\}$

Induktionsschritt: $A \rightarrow_{G'}^* s'$: zerlege $A \rightarrow_{G'}^* s_1 \rightarrow_{G'}^* s'$ Fälle für s_1 (bzw. $A \rightarrow s_1$)

- $a \rightarrow f(A_1, \dots, A_m) \Rightarrow s' = f(t_1, \dots, t_n)$. Laut Induktionsvoraussetzung: $t_i \in L(A_i)\{\square_1 \leftarrow L_1, \dots, \square_n \leftarrow L_n\} \Rightarrow s' \in L(A)$
- $A \rightarrow s_i \in R' \Rightarrow A \rightarrow \square_i \in R$ (laut Konstruktion) $\Rightarrow \square_i \in L(A) \Rightarrow s' \in L_i \Rightarrow s' \in L(A)\{\square_1 \leftarrow L_1, \dots, \square_n \leftarrow L_n\}$

Aussage gilt für alle Nichtterminale A_i , also auch für Startsymbol S . \square

2.9 Satz

Ist $L \subseteq T_{\Sigma \cup K}$ regulär und $\square \in K$, dann ist $L^*.\square$ regulär.

Beweis: Betrachte normalisierte Grammatik $G = (S, N, \Sigma \cup K, R)$ mit $L(G) = L$.

Konstruiere $G' = (S', N \cup \{S'\}, \Sigma \cup K, R')$ (mit $S' \notin N$) wie folgt:

R' enthalte alle Regeln aus R , wobei:

- $A \rightarrow \square$ wird ersetzt durch $A \rightarrow S'$
- $S' \rightarrow S$
- $S' \rightarrow \square$ (damit $\square \in L(G')$)

Induktion liefert: $L(G') = L^*.\square$

2.10 Definition

Wir formalisieren nun die rationalen Ausdrücke:

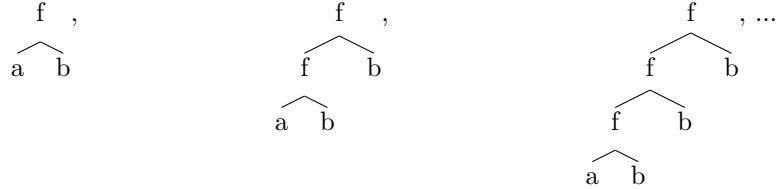
Sei Σ ein Rangalphabet, K eine Menge Konstanten mit $\square \in K$: Dann ist $Rat(\Sigma, K)$ die kleinste Menge, sodass:

- $\emptyset \in Rat(\Sigma, K)$
- $a \in \Sigma^0 \cup K' \Rightarrow a \in Rat(\Sigma, K)$
- $f \in \sigma^n, E_1, \dots, E_n \in Rat(\Sigma, K) \Rightarrow f(E_1, \dots, E_n) \in Rat(\Sigma, K)$
- $E_1, E_2 \in Rat(\Sigma, K) \Rightarrow E_1 \cup E_2 \in Rat(\Sigma, K)$
- $E_1, E_2 \in Rat(\Sigma, K), \square \in K \Rightarrow E_1 \cdot \square E_2 \in Rat(\Sigma, K)$
- $E_1 \in Rat(\Sigma, K), \square \in K \Rightarrow E_1^* \cdot \square \in Rat(\Sigma, K)$

Die Ausdrücke $Rat(\Sigma, K)$ heißen rational über Σ und K . Ist E ein rationaler Ausdruck in $Rat(\Sigma, K)$, dann repräsentiert E eine Menge von Termen aus $T_{\Sigma \cup K}$, bezeichnet mit $\|E\|$.

- $\|\emptyset\| = \emptyset$
- $\|a\| = \{a\}$
- $\|f(E_1, \dots, E_n)\| = \{f(t_1, \dots, t_n) \mid t_i \in \|E_i\|\}$
- $\|E_1 \cup E_2\| = \|E_1\| \cup \|E_2\|$
- $\|E_1 \cdot E_2\| = \|E_1\| \cdot \{\square\} \cdot \|E_2\|$
- $\|E^* \cdot \square\| = \|E\|^* \cdot \square$

Beispiel: Der Ausdruck $(f(\square, b))^* \cdot \square a$ ist rational und repräsentiert alle Terme der Form



Wir erhalten die „Baumautomaten“-Variante von Kleene:

2.11 Theorem

Rationale Ausdrücke haben die selbe Ausdrucksstärke wie bottom-up Baumautomaten.

Beweis: Ist E ein rationaler Ausdruck, so existiert laut Sätzen 2.8 und 2.9 und den Abschlusseigenschaften von Baumautomaten ein Baumautomat \mathcal{A} mit $L(\mathcal{A}) = \|E\|$. Umgekehrt: Sei $\mathcal{A} = (Q, \Sigma, F, \Delta)$ ein Baumautomat. Wir zeigen: Es existiert ein rationaler Ausdruck E aus $Rat(\Sigma, K)$ mit $\|E\| = L(\mathcal{A})$.

Betrachte hierzu für $1 \leq i, j \leq |Q|$ und $Z \subseteq Q$ Terme $T(i, j, Z)$ definiert wie folgt: $t \in T(i, j, Z)$, falls $t \in T_{\Sigma \cup K}$ mit einem Lauf R in t , so dass:

- $v(\varepsilon) = q_i$
- ist $p \neq \varepsilon$ und $t(p)/inZ$, dann ist $r(p) \in \{q_1, \dots, q_j\}$

Damit: $L(\mathcal{A}) = \bigcup_{q_i \in F} T(i, |Q|, \emptyset)$

Induktion über j : $T(i, j, Z) \in Rat(\Sigma, K)$ $j = 0$: $t \in T(i, 0, Z)$ bedeutet, es existiert ein Lauf r

- $r(\varepsilon) = q_i$
- kein Symbol, das nicht nullstellig ist, darf gelabelt sein $\Rightarrow t = a$ oder $t = f(a_1, \dots, a_n), a_1, \dots, a_n \in \Sigma^0 \Rightarrow$ endlich viele, also existiert ein rationaler Ausdruck

Induktionsschritt: Angenommen für $j' < j$ ist $T(i, j', Z) \in Rat(\Sigma, Q)$.

Schreibe $T(i, j, Z) = T(i, j-1, Z) \cup T(i, j-1, Z \cup \{q_j\}) \cdot_{q_j} (T(j, j-1, Z \cup \{q_j\}))^{* \cdot q_j} \cdot_{q_j} T(j, j-1, Z)$

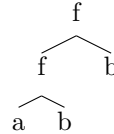
Ziel: Zusammenhang zwischen Baum- und Wortsprachen

Neben der Pfadsprache (siehe Übung) betrachten wir den sogenannten Yield-Operator, definiert wie folgt:

- $Yield(a^{(0)}) = a$ für eine Konstante $a^{(0)} \in \Sigma$
- $Yield(f(t_1, \dots, t_n)) = Yield(t_1) \cdot \dots \cdot Yield(t_n)$ für $f^{(n)} \in \Sigma, t_i \in T_\Sigma$

Für Sprache L gilt: $Yield(L) = \bigcup_{t \in \Sigma} Yield(t)$

Beispiel: $Yield(f(f(a, b), b)) = abb$



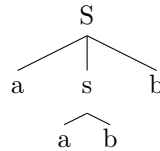
$$Yield(||g(a, \square, b)^{*\square} \cdot \square f * (a, b)||) = \{a^n b^n | n \in \mathbb{N}, n \geq 1\}$$

Wir wollen reguläre Baumsprachen mit Ableitungsbäumen von kontextfreien Wortgrammatiken „vergleichen“. Betrachte $G = (S, N, T, R)$ (kontextfreie Wortgrammatik, d.h. Regeln in R haben die Form $A \rightarrow \alpha$, wobei $A \in N, \alpha \in (N \cup T)^+$).

Betrachte zum Beispiel $G = (S, N, T, R)$ mit Regeln:

$S \rightarrow aSb, S \rightarrow ab$ mit $L(G) = \{a^n b^n | n \in \mathbb{N}, n \geq 1\}$.

Der Syntaxbaum für $aabb$ hat die Form



, d.h. S hat keinen festen Rang.

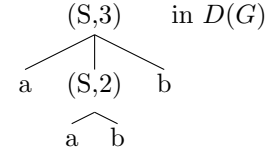
Betrachte daher für gegebene Grammatik G Tupel (A, m) für jedes $A \in N$, sodass $A \rightarrow \alpha \in R$ mit $|\alpha| = m$.

Zu einem Symbol $a \in T \cup N$ definieren wir die Menge der von a ausgehenden Ableitungsbäume in G , $D(G, a)$, wie folgt:

- $D(G, a) = \{a\}$ für $a \in T$
- $(a, 0)(\varepsilon) \in D(G, a)$, falls $a \rightarrow \varepsilon \in R$

- $(a, m)i(t_1, \dots, t_m) \in D(G, a)$, falls $t_i \in (G, a_i)$, $a_i \in T \cup N$, $a \rightarrow a_1, \dots, a_m \in R$
- Ableitungsbäume für G sind $D(G) = \bigcup_{a \in T \cup N} D(G, a)$

Beispiel: $G = (S, N, T, R)$ mit $S \rightarrow aSb, S \rightarrow ab$. Dann ist



Bemerkung: Ist G eine kontextfreie Grammatik, so ist natürlich $Yield(D(G)) = L(G)$.

2.12 Satz

- 1.) Ist $G = (S, N, T, R)$ eine kontextfreie Wortgrammatik, dann ist $D(G)$ reguläre Baumsprache.
- 2.) Ist L reguläre Baumsprache, dann ist $Yield(L)$ kontextfreie Wortsprache.
- 3.) Es existieren reguläre Baumsprachen, die nicht Ableitungsbäumen von kontextfreien Sprachen entsprechen.

Beweis:

- 1.) Erzeuge Grammatik $G' = (S, N, \Sigma, R')$ mit:

- $\Sigma = T \cup \{\varepsilon\} \cup \{(A, m) | A \in N, A \rightarrow \alpha \in R \text{ mit } |\alpha| = m\}$
- $A \rightarrow (A, 0)(\varepsilon) \in R'$, falls $A \rightarrow \varepsilon \in R$
- $A \rightarrow (A, m)(A_1, \dots, A_m) \in R'$, falls $A \rightarrow A_1 \dots A_m \in R$

damit ist offensichtlich $Yield(L(G')) = L(G)$.

- 2.) Betrachte normalisierte Grammatik $G = (S, N, \Sigma, R)$ mit $L(G) = L$.

Erzeuge Wortgrammatik $G' = (S, N, \Sigma^{(0)}, R')$ mit folgenden Regeln:

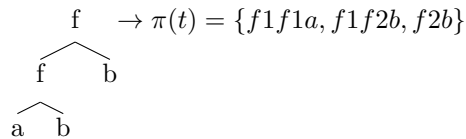
- $A \rightarrow A_1, \dots, A_m$ ($A \rightarrow a$) $\in R'$, falls $A \rightarrow f(A_1, \dots, A_n)$ ($A \rightarrow a$) $\in R$ für $f^{(m)} \in \Sigma$

- 3.) Übung

Pfadsprache $\pi(n)$:

- $a \in \Sigma^0$ dann ist $\pi(a) = \{a\}$
- $t = f(t_1, \dots, t_n)$, dann ist $\pi(t) = \bigcup_{i=1}^n \{fiw | w \in \pi(t_i)\}$

Beispiel: $t =$



$$\pi(L) = \bigcup_{t \in L} \pi(t)$$

2.13 Satz

Ist $L \subseteq T_\Sigma$ eine reguläre Baumsprache, dann ist $\pi(L)$ eine reguläre Wortsprache.

Komplexität

Um den Input zu formalisieren, setze: (für $t \in T_\Sigma$)

- $||t|| = 1$, falls t Konstante
- $||t|| = 1 + \sum_{i=1}^n ||t_i||$, falls $t = f(t_1, \dots, t_n)$
 $\cong |Pos(t)|$
- für einen Automaten \mathcal{A} : $||\mathcal{A}|| = |Q| + \sum_{r \in \mathcal{A}} ||r||$, wobei $||r|| = n + 2$ für $r = f(q_1, \dots, q_n) \rightarrow q$

Betrachte nun:

MEMBERSHIP

- Input: $t \in T_\Sigma$
- Output: „Yes“, dggw. $t \in L(\mathcal{A})$ für gegebenen Automaten \mathcal{A}

→ linear

UNIFORM MEMBERSHIP

- Input: Input: $t \in T_\Sigma, \mathcal{A}$ bottom-up Baumautomat
- Output: „Yes“, dggw. $t \in L(\mathcal{A})$

→ deterministischer TA: *rightarrow* linear

→ nichtdeterministischer TA: *rightarrow* $O(||t|| \cdot ||\mathcal{A}||)$ (polynomiell)

(„on the fly“ alle erreichbaren Zustände bestimmen)

EMPTYNESS

- Input: \mathcal{A} bottom-up Baumautomat
- Output: „Yes“ dggw. $L(\mathcal{A}) = \emptyset$

→ übersetze in Horn-Formeln

→ linear

UNIVERSALITY

- Input: \mathcal{A}
- Output: „Yes“ dggw. $L(\mathcal{A}) = T_\Sigma$

→ Exptime-vollständig (Determinisierten, bevor \mathcal{A}^c gebildet werden kann)

FINITENESS

- Input: \mathcal{A}
- Output: „Yes“ gdw. $L(\mathcal{A})$ endlich

→ Pumping: Exptime

→ nach Loops suchen für Automat

q mit $C[q] \rightarrow^* q$ für $C \in T) \Sigma(x)$

und $C'[q] \rightarrow^* q_f \in F$ für $C' \in T) \Sigma(x)$

→ polynomiell