Kurze Einführung in Baumsprachen

Die folgende Einführung in Baumsprachen ist ein minimal angepasster Ausschnitt aus der Bachelor-Arbeit von Peter Bücker (peter.buecker@uni-duesseldorf.de), geschrieben bei Jun.-Prof. Wiebke Petersen im Sommersemester 2011 an der Heinrich-Heine-Universität Düsseldorf.

1 Formale Baumsprachen

Formale Wortsprachen sind, wie im vorherigen Kapitel erläutert, Mengen von Wörtern und Wörter sind Folgen von Zeichen (Buchstaben, Ziffern, …). Analog dazu sind formale Baumsprachen Mengen von Bäumen, aber was genau ist ein Baum? Ein Baum ist eine hierachische Struktur, aufgebaut aus Knoten und einer Dominanzrelation. An der Spitze eines Baums steht ein einziger Knoten, der sogenannte Wurzelknoten. Für sich genommen ist er bereits ein vollständiger Baum, in aller Regel gehören zum Wurzelknoten aber weitere, untergeordnete Knoten hinzu. Diese untergeordneten Knoten bezeichnet man als Kinder des Elternknotens, sie werden vom Elternknoten dominiert. So ergibt sich eine hierarchische Ordnung, ausgehend vom Wurzelknoten bis hin zu den sogenannten Blättern, also den Knoten, die über keine weiteren Kinder verfügen. Ein Baum ist sowohl in seiner Breite als auch in seiner Tiefe unbeschränkt. Die einzige strukturelle Einschränkung liegt darin, dass 1. jeder Knoten einen Elternknoten benötigt (abgesehen vom Wurzelknoten), und 2. dass jeder Knoten nur genau einen Elternknoten besitzt. Mit anderen Worten: Es handelt sich um einen gerichteten, azyklischen Graphen. Abbildung 1 zeigt als Beispiel für einen Baum den Ableitungsbaum des Wortes *abab* aus der Grammatik in Abbildung 2 (Wörter gerader Länge, bestehend aus a und b).

Abbildung 1: Ableitungsbaum des Wortes *abab* aus der Grammatik in Abbildung 2

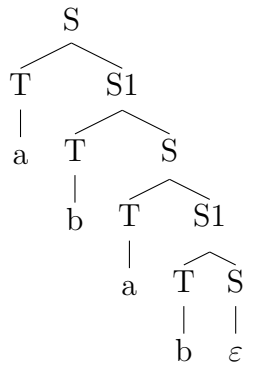


Abbildung 2: Beispielgrammatik G_1

$$
\begin{aligned}
S &\rightarrow \varepsilon \\
S &\rightarrow T\ S1 \\
S1 &\rightarrow T\ S \\
T &\rightarrow a \\
T &\rightarrow b
\end{aligned}
$$

Bäume lassen sich abgesehen von der bildlichen Darstellung auch als Term repräsentieren. Der Term zum Ableitungsbaum in Abbildung 1 sieht wie folgt aus:

$$S(T(a), S1(T(b), S(T(a), S1(T(b), S())))).$$

Jeder Knoten, der kein Blatt ist, wird dabei durch einen n-stelligen Term repräsentiert, wobei n die Anzahl von Kindern des Knotens ist. Die Argumente eines Terms werden auf die gleiche Art und Weise gebildet. Blattknoten werden als Atom repräsentiert, d. h. sie tragen keine Klammern. Da beide Repräsentationsformen äquivalent sind, werden wir die Begriffe Baum und Term im weiteren Verlauf der Arbeit als austauschbar behandeln. Im Folgenden werden Terme und Bäume formal definiert, gefolgt von grundlegenden Konzepten wie Ersetzungen und Kontexten (Comon u. a. 2008, S. 15–17).

1.1 Terme und Bäume

Das obige Beispiel eines Terms zeigt die zwei unterschiedlichen Arten von Ausdrücken, die es in einem Term gibt. Einerseits gibt es komplexe Ausdrücke der Form $f(t_1, \ldots, t_n)$, und andererseits atomare Symbole wie a oder b, auch Variablen genannt. Terme werden rekursiv definiert, d. h. zunächst ist jede Variable ein Term. Weiterhin ist ein Ausdruck wie $f(t_1, \ldots, t_n)$ ein Term, falls f ein n-stelliger Funktor ist und $t_1, \ldots, t_n$ Terme sind.

Definition 1 (Comon u. a. 2008, S. 15)**.** *Sei N die Menge von positiven ganzen Zahlen. Die Menge von endlichen Ketten über N nennen wir N^*. Ein **Rangalphabet** ist ein Paar $(\mathcal{F}, Arity)$, wobei $\mathcal{F}$ eine endliche Menge und Arity eine Abbildung $\mathcal{F} \rightarrow N$ ist. Arity weist jedem $f \in \mathcal{F}$ eine Stelligkeit $Arity(f)$ zu. Die Menge aller n-stelligen Funktoren schreiben wir $\mathcal{F}_n$. 0-stellige Funktoren heißen **Konstanten**. Wir verwenden eine Kurzschreibweise aus Funktoren und leeren Argumentstellen um*

anzuzeigen, welche Stelligkeit ein Funktor hat: $f(,)$ ist beispielsweise ein 2-stelliger Funktor.

*Sei $\mathcal{X}$ eine Menge von Konstanten, genannt **Variablen**. Es gelte $\mathcal{X} \cap \mathcal{F}_0 = \emptyset$. Die Menge $T(\mathcal{F}, \mathcal{X})$ von **Termen** über dem Rangalphabet $\mathcal{F}$ und den Variablen $\mathcal{X}$ ist die kleinste Menge, so dass gilt:*

- $\mathcal{F}_0 \subseteq T(\mathcal{F}, \mathcal{X})$
- $\mathcal{X} \subseteq T(\mathcal{F}, \mathcal{X})$
- *Wenn $n \geq 1$, $f \in \mathcal{F}_n$ und $t_1, \ldots, t_n \in T(\mathcal{F}, \mathcal{X})$, dann ist $f(t_1, \ldots, t_n) \in T(\mathcal{F}, \mathcal{X})$.*

*Wenn $\mathcal{X} = \emptyset$, dann schreiben wir einfach $T(\mathcal{F})$ anstatt von $T(\mathcal{F}, \mathcal{X})$. Terme in $T(\mathcal{F})$ nennen wir **Basisterme**. Ein Term $t \in T(\mathcal{F}, \mathcal{X})$ heißt **linear**, falls jede Variable höchstens ein Mal vorkommt.*

Definition 2 (Comon u. a. 2008, S. 15–16)**.** *Ein **endlicher, geordneter Baum** über einer Menge von Beschriftungen E ist eine Abbildung von einer präfixabgeschlossenen Menge $\mathcal{P}os(t) \subseteq N^*$ nach E. Ein Term $t \in T(\mathcal{F}, \mathcal{X})$ kann als **endlicher, geordneter Rangbaum** betrachtet werden, dessen Blätter mit Variablen oder Konstanten beschriftet sind, während die übrigens Knoten mit mindestens 1-stelligen Funktoren beschriftet sind. Ein mit einem n-stelligen Funktor beschrifteter Knoten muss genau n Kinder haben. Genauer gesagt kann ein Term $t \in T(\mathcal{F}, \mathcal{X})$ als partielle Funktion $t : N^* \to \mathcal{F} \cup \mathcal{X}$ mit Definitionsbereich $\mathcal{P}os(t)$ betrachtet werden, falls t die folgenden Bedingungen erfüllt:*

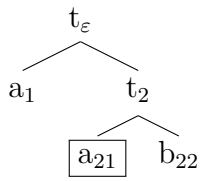
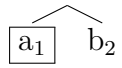
- *$\mathcal{P}os(t)$ ist nicht leer und präfix-abgeschlossen*
- *$\forall p \in \mathcal{P}os(t)$: falls $t(p) \in \mathcal{F}_n, n \geq 1$, dann ist $\{j \mid pj \in \mathcal{P}os(t)\} = \{1, \ldots, n\}$*
- *$\forall p \in \mathcal{P}os(t)$: falls $t(p) \in \mathcal{X} \cup \mathcal{F}_0$, dann ist $\{j \mid pj \in \mathcal{P}os(t)\} = \emptyset$.*

*Jedes Element aus $\mathcal{P}os(t)$ wird als **Position** bezeichnet. Falls $\forall j \in N : pj \notin \mathcal{P}os(t)$ gilt, so heißt p **Grenzposition**. Die Menge aller Grenzpositionen heißt $\mathcal{FP}os(t)$. Falls $t(p) \in \mathcal{X}$ gilt, so heißt p **Variablenposition**. Die Menge aller Variablenpositionen heißt $\mathcal{VP}os(t)$. Der Wurzelknoten wird als $Head(t)$ bezeichnet, wobei $Head(t) = t(\varepsilon)$.*

Bevor weitere Begriffe eingeführt werden, sollen die Definitionen 1 und 2 näher erläutert werden. Eine Voraussetzung für die Definition von Termen sind Rangalphabete. Ein Rangalphabet ist eine Menge, in der jedem Element eine Stelligkeit zugeordnet wird. Nehmen wir als Beispiel die folgende Konfiguration an:

$$M = \{a, b, t\}, Arity(a) = 0, Arity(b) = 1, Arity(t) = 2.$$

Abbildung 3: Baum des Terms $t(a, t(a, b))$ samt aller Teilbäume

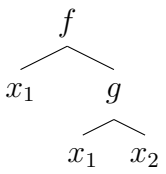
Term	$t(a, t(a, b))$	a	$t(a, b)$	a	b
Teilterm	t	$t\|_1$	$t\|_2$	$t\|_{21}$	$t\|_{22}$
Baum		a_ε		a_ε	b_ε

Im Rangalphabet $A = (M, Arity)$ wird jedem Element aus der Menge M über die Funktion $Arity$ eine Stelligkeit zugewiesen. In diesem Beispiel ist a 0-stellig, b 1-stellig und t 2-stellig. Alternativ zur Auflistung aller Zuweisungen der Funktion $Arity$ kann man auch eine abgekürzte Schreibweise mit a, $b()$ und $t(,)$ oder $a/0$, $b/1$ und $t/2$ verwenden. Terme bestehen aus zwei disjunkten Mengen, nämlich der Menge der Funktoren $\mathcal{F}$ und der Menge der Variablen $\mathcal{X}$. Funktoren können dazu genutzt werden, komplexe Ausdrücke zu erzeugen, während Variablen als Platzhalter fungieren (siehe unten). Als Besonderheit werden 0-stellige Funktoren auch als Konstanten bezeichnet. Wie vor Definition 1 bereits kurz angedeutet, werden Terme induktiv definiert, ähnlich zur Definition von regulären Wortausdrücken. Das heißt, es werden zunächst die kleinstmöglichen Terme definiert – dies sind Konstanten und Variablen: Jede Konstante ist ein Term, ebenso wie alle Variablen. Auf dieser Basis können durch Verschachtelung beliebig komplexe Ausdrücke wie z. B. $t(a, t(a, b))$ gebildet werden (vgl. Abbildung 3).

In Definition 2 werden Bäume als eine partielle Funktion aus Ketten von natürlichen, positiven Zahlen nach $\mathcal{F} \cup \mathcal{X}$ eingeführt. Der Definitionsbereich der Funktion wird als $\mathcal{P}os(t)$ bezeichnet und fungiert als hierarchisches Adressierungssystem für Knoten im Baum. Jedes Element aus $\mathcal{P}os(t)$ ist eine Kette von natürlichen, positiven Zahlen, in der jede Zahl für einen Knoten des Baums steht. Die Abfolge der Zahlen bestimmt einen Pfad durch den Baum. Beispiel: Die Adresse 2.1 bezeichnet in Abbildung 3 den ersten Knoten unterhalb des zweiten Knotens unterhalb der Wurzel.[1] Zerlegt man die Adresse in eine Folge von Zahlen $a_1, \ldots, a_n$, so referenziert a_1 den a_1-sten Knoten unterhalb der Wurzel, a_2 den a_2-ten Knoten unterhalb von a_1, und so weiter. Die Funktion $\mathcal{P}os(t)$ weist jeder solchen Adresse ein Element aus $\mathcal{F} \cup \mathcal{X}$ zu, d. h. zu jeder Adresse existiert im Baum ein entsprechender Knoten.

[1] Zur Übersicht wurden alle Knoten mit ihrer Adresse als Subskript versehen. Die einzelnen Stellen der Adressen werden mit einem Punkt voneinander getrennt.

Abbildung 4: Term $f(x_1, g(x_1, x_2))$



$$\mathcal{F} = \{f(,), g(,), a, b\},\ \mathcal{X} = \{x_1, x_2\}$$

Da jeder Term aus kleineren Termen besteht, können wir einen Term in seine Teilterme zerlegen. In Abbildung 3 sehen wir die Zerlegung des Terms $t(a, t(a, b))$ in seine vier Teilterme. Teilterme werden über ihre Position im ursprünglichen Term adressiert, z. B. $t|_2$ für den Teilterm $t(a, b)$ aus $t(a, t(a, b))$.

Definition 3 (Comon u. a. 2008, S. 16)**.** *Ein* ***Teilterm*** $t|_p$ *eines Terms* $t \in T(\mathcal{F}, \mathcal{X})$ *an Position* p *ist wie folgt definiert:*

- $\mathcal{P}os(t|_p) = \{j \mid pj \in \mathcal{P}os(t)\}$
- $\forall q \in \mathcal{P}os(t|_p),\ t|_p(q) = t(pq)$

Mit $t[u]_p$ *bezeichnen wir den Term, der entsteht, wenn wir* u *anstelle von* $t|_p$ *in* t *einsetzen. Wir bezeichnen* $\trianglerighteq$ *als die* ***Teilterm-Ordnung****, d. h. wir schreiben* $t \trianglerighteq t'$*, falls* t' *ein Teilterm von* t *ist. Wir schreiben* $t \triangleright t'$*, falls* $t \trianglerighteq t'$ *und* $t \neq t'$*. Eine Menge von Termen* F *wird geschlossen genannt, falls sie bezüglich der Teilterm-Ordnung geschlossen ist, d. h. falls gilt:* $\forall t \in F : (t \trianglerighteq t' \Rightarrow t' \in F)$*.*

Die folgende Definition zeigt, wie Ersetzungen in Termen vorgenommen werden können. Dazu wird eine Abbildung σ definiert, die Variablen auf Terme abbildet, d. h. Variablen werden durch Terme ersetzt. Variablen können auch durch andere Variablen ersetzt werden, aber nur endlich viele Variablen werden nicht auf sich selbst abgebildet. Anders gesagt, darf eine Ersetzung nicht unendlich viele Ersetzungen vornehmen. Nehmen wir als Beispiel für eine Ersetzung den Term aus Abbildung 4 und $\sigma = \{x_1 \leftarrow a, x_2 \leftarrow g(a, b)\}$. Die Ersetzung $\sigma(f(x_1, g(x_1, x_2)))$ ergibt $f(a, g(a, g(a, b)))$, denn alle Vorkommen der Variable x_1 werden durch a ersetzt, und alle Vorkommen von x_2 durch $g(a, b)$.

Definition 4 (Comon u. a. 2008, S. 17)**.** *Eine* ***Ersetzung*** σ *ist eine Abbildung von* $\mathcal{X}$ *nach* $T(\mathcal{F}, \mathcal{X})$*, in der es nur endlich viele Variablen gibt, die nicht auf sich selbst abgebildet werden. Die Domäne einer Ersetzung* σ *ist die Teilmenge von Variablen* $x \in \mathcal{X}$*, so dass* $\sigma(x) \neq x$*. Die Ersetzung* $\{x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n\}$ *ist* $\mathcal{X} \setminus \{x_1, \ldots, x_n\}$ *nach Abbildung von* $x_i \in \mathcal{X}$ *auf* $t_i \in T(\mathcal{F}, \mathcal{X})$ *ab, für alle* $1 \leq i \leq n$*.*

Ersetzungen lassen sich auf $T(\mathcal{F}, \mathcal{X})$ erweitern:

$$\forall f \in \mathcal{F}_n, \forall t_1, \ldots, t_n \in T(\mathcal{F}, \mathcal{X}) : \sigma(f(t_1, \ldots, t_n)) = f(\sigma(t_1), \ldots, \sigma(t_n)).$$

Wir wollen nun noch den Begriff des Kontexts einführen. Ein Kontext ist ein linearer Term mit mindestens einer Variablen und dient als Marker in einem Baum, d. h. über einen Kontext kann eine Stelle im Baum identifiziert werden, an der später beispielsweise eine Ersetzung durchgeführt werden soll. Betrachten wir dazu den Term $t(a, t(a, b))$ und den Kontext $C = t(a, x_1)$: Über den Ausdruck $C[b]$ können wir nun eine Ersetzung von des Teilbaums $t(a, b)$ durch b vornehmen und erhalten den Term $t(a, b)$.

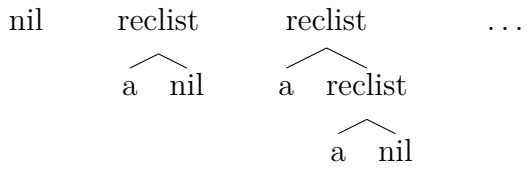
Definition 5 (Comon u. a. 2008, S. 17)**.** *Sei $\mathcal{X}_n$ eine Menge von n Variablen. Ein linearer Term $C \in T(\mathcal{F}, \mathcal{X}_n)$ heißt **Kontext** und der Ausdruck $C[t_1, \ldots, t_n]$ für $t_1, \ldots, t_n \in T(\mathcal{F})$ denotiert den Term in $T(\mathcal{F})$, der durch Ersetzung von x_i durch t_i in C für $1 \leq i \leq n$ entsteht: $C[t_1, \ldots, t_n] = C\{x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n\}$. Die Menge aller Kontexte über $(x_1, \ldots, x_n)$ nennen wir $\mathcal{C}^n(\mathcal{F})$.*

Die wichtigsten Begriffe rund um Bäume und Terme sind damit eingeführt. Wir wollen uns nun mit Baumgrammatiken, dem Äquivalent zu Grammatiken für formale Wortsprachen, befassen.

1.2 Baumgrammatiken

Eine formale Baumgrammatik besteht, ebenso wie eine formale Wortgrammatik, aus vier Teilen: einem Startsymbol, Terminalsymbolen, Nicht-Terminalsymbolen und Produktionsregeln. Im Wortfall sind alle Terminale 0-stellig, d. h. sie haben keine innere Struktur. Deshalb lassen sich nur flache Zeichenketten aus einer Wortgrammatik ableiten, obwohl während der Ableitung durchaus eine Baumstruktur, nämlich der Ableitungsbaum, entsteht. Möchte man aus einer Grammatik hingegen Bäume ableiten, müssen mehrstellige Terminale erlaubt sein. Als Beispiel soll eine Baumgrammatik dienen, die Listen in der Form von Listenkopf und Restliste modelliert, wie es beispielsweise in funktionalen Programmiersprachen oder Merkmalstrukturen

Abbildung 5: Beispielableitungen aus der Grammatik G



üblich ist:[2]

$$\begin{aligned} List &\to nil \\ List &\to reclist(Head, List) \\ Head &\to a \\ Head &\to b \end{aligned}$$

Die Grammatik hat zwei Nichtterminale ($List$, $Head$), vier Terminale ($nil/0$, $a/0$, $b/0$ und $reclist/2$), das Startsymbol $List$ und die abgebildeten Produktionsregeln. Die Regeln der Form $Head \to \beta$ dienen zur Generierung der in der Liste enthaltenen Objekte und sind für dieses Beispiel bewusst einfach gehalten. Das Terminal nil steht für eine leere Liste. Einige Beispielableitungen finden sich in Abbildung 5. Der Beispielbaum $reclist(a, reclist(a, nil))$ leitet sich aus der Grammatik wie folgt ab:

$$\begin{aligned} List &\Rightarrow reclist(Head, List) \Rightarrow reclist(a, List) \Rightarrow reclist(a, reclist(Head, List)) \\ &\Rightarrow reclist(a, reclist(a, List)) \Rightarrow reclist(a, reclist(a, nil)) \end{aligned}$$

Im ersten Schritt wird aus dem Startsymbol eine 1-elementige Liste $reclist(Head, List)$ abgeleitet. Die Regel $Head \to a$ ersetzt in diesem Baum $Head$ durch a, gefolgt von der Regel $List \to reclist(Head, List)$, die $List$ im Inneren des Terms durch eine weitere 1-elementige Liste ersetzt. Dieser Prozess wird dann wiederholt und mit einer Anwendung der Regel $List \to nil$ terminiert.

Definition 6 (Comon u. a. 2008, S. 51–52)**.** *Eine* ***Baumgrammatik*** G *ist ein* 4*-Tupel* $(N, \mathcal{F}, P, S)$*.* N *ist das Rangalphabet der Nichtterminalsymbole.* $\mathcal{F}$ *ist das Rangalphabet der Terminalsymbole. Es gelte* $N \cap \mathcal{F} = \emptyset$*.* P *ist die endliche Menge der Produktionsregeln* $\alpha \to \beta$*, wobei* $P \subseteq T(\mathcal{F} \cup N \cup \mathcal{X}) \times T(\mathcal{F} \cup N \cup \mathcal{X})$*.* $\mathcal{X}$ *ist eine Menge von Dummy-Variablen.* $S \in N$ *ist das Startsymbol mit* $\mathcal{A}rity(S) = 0$*. Eine* ***reguläre Baumgrammatik*** $G = (N, \mathcal{F}, P, S)$ *ist eine Baumgrammatik mit*

[2]Das Beispiel entstammt ursprünglich aus Comon u. a. (2008, S. 23).

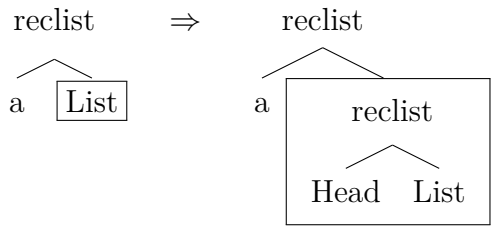
den folgenden Einschränkungen:

- $\forall n \in N : \mathcal{A}rity(n) = 0$
- $\forall(\alpha \to \beta) \in P : \alpha \in N,\ \beta \in T(\mathcal{F} \cup N)$

Die Expansionsrelation für die Grammatik G nennen wir $\Rightarrow_G$. Falls G aus dem Kontext bekannt ist, schreiben wir einfach $\Rightarrow$. Es gilt $s \Rightarrow_G t$, falls eine Regel $A \to \alpha$ und ein Kontext C existieren, so dass gilt: $s = C[A]$ und $t = C[\alpha]$.

Die von G definierte/erzeugte/generierte Sprache ist $L(G) = \{t \in T(\mathcal{F}) \mid S \Rightarrow_G^+ t\}$. $\Rightarrow_G^+$ ist die transitive Hülle von $\Rightarrow_G$. Eine Folge von Bäumen $(t_0, t_1, \ldots, t_n)$ mit $t_0 = S$, $t_n \in T(\mathcal{F})$ und $t_i \Rightarrow_G t_{i+1}$ heißt Ableitung von t_n.

Die grundlegenden Bestandteile einer Baumgrammatik wie Nichtterminale, Terminale und Produktionsregeln wurden bereits erläutert. Im Folgenden soll noch gezeigt werden, auf welche Art und Weise eine Baumgrammatik eine Baumsprache generiert bzw. beschreibt. Kurz gesagt, sind all die Bäume Teil der generierten Sprache, die sich aus der Grammatik ableiten lassen. Das heißt, alle Bäume $t \in T(\mathcal{F})$, die sich durch schrittweise Anwendung der Produktionsregeln aus dem Startsymbol S erzeugen lassen, sind in der von G generierten Sprache $L(G)$ enthalten. Formal betrachtet nutzt man dazu die Expansionsrelation $\Rightarrow_G^+$, die wir schon von den formalen Wortgrammatiken kennen. Zwei Bäume stehen in der Expansionsrelation $s \Rightarrow t$, falls es einen Kontext und eine Produktionsregel gibt, so dass die linke Regelseite eingesetzt in den Kontext s ergibt, während die rechte Regelseite, eingesetzt in den selben Kontext, t ergibt. Mit anderen Worten: Der Kontext markiert einen Knoten im Baum, auf den eine Regel angewandt wird, und die Regel kann nur angewandt werden, wenn der betroffene Knoten vorher der linken Regelseite entsprach. Betrachten wir dazu ein Beispiel:



Die Expansion im Beispiel ist möglich, da ein Kontext und eine Regel existieren, nämlich $C = reclist(a, x_1)$ und $List \to reclist(Head, List)$, so dass durch Einsetzen der linken Regelseite in C der Baum $reclist(a, List)$ entsteht, und durch Einsetzen der rechten Regelseite $reclist(a, reclist(Head, List))$.

1.3 Reguläre Baumausdrücke

Während Grammatiken Sprachen generieren, dienen reguläre Ausdrücke der Beschreibung von Sprachen. Reguläre Ausdrücke bringen einer Sprache zugrundeliegende Muster wie etwa Wiederholungen zum Ausdruck. Wir haben im vorherigen Kapitel bereits gesehen, dass reguläre Ausdrücke für Wörter die Metasymbole $*$ oder $+$ verwenden, um Wiederholungen oder Vereinigung von Sprachen auszudrücken. Reguläre Baumausdrücke funktionieren ganz ähnlich, obgleich ihre Syntax etwas komplizierter ist, wie wir im Folgenden sehen werden.

Nehmen wir als Beispiel zunächst wieder die Sprache, die alle rekursiven Listen enthält, wie in Kapitel 1.2 beschrieben.[3] Die Sprache sieht wie folgt aus:

$$List = \{nil, reclist(a, nil), reclist(a, reclist(a, nil)), \dots\}.$$

Anstelle von a ließen sich beliebige andere Terme einsetzen – der genaue Inhalt ist hier aber nicht weiter interessant. Die Bäume der Sprache L zeigen ein deutliches Wiederholungsmuster: Ein Baum ist entweder die leere Liste nil, oder er besteht aus dem zweistelligen Funktor $reclist$ mit einem Inhaltsargument a, und der Restliste, die wiederum ein Baum ist. Die Sprache $List$ kann mit Hilfe eines regulären Baumausdrucks wie folgt modelliert werden:

$$List = L(nil + reclist(a, \square_1)^{*,\square_1}._{\square_1} nil).$$

Der Ausdruck enthält ein neues, bisher nicht verwendetes Symbol $\square_1$, um zu markieren, an welcher Stelle des Baums Operationen wie Iteration oder Konkatenation angewandt werden sollen. Bei Wortsprachen brauchen wir einen solchen Platzhalter nicht, da die Operation direkt am Ort des Auftretens ausgeführt werden kann, wie z. B. im regulären Ausdruck `ab*c`. Hier ist klar, an welcher Stelle beliebig viele `b`s stehen können, nämlich zwischen `a` und `c`. Derartige Operationen auf Bäumen werfen die grundsätzliche Frage auf, wo sie geschehen sollen, da ein Baum sich nur durch Verschachtelung erweitern lässt. Deshalb wäre bei nicht-unären Bäumen ohne entsprechende Platzhalter unklar, an welcher Stelle eine Operation ausgeführt werden soll. Beispiel: Der Baum $f(a, b)$ kann nicht einfach um den Baum $g(c)$ erweitert werden, ohne zu wissen, an welcher Stelle des ersten Baums dies geschehen soll.

Neben diesem neuen Symbol enthält der Ausdruck die zwei neuen Konstruktionen $t^{*,\square}$ und $t._{\square} s$. Die erste Operation entspricht dem Kleeneschen Stern für reguläre

[3] Wir verwenden in diesem Abschnitt anstatt von $Head$ nur das einfache Symbol a, da die Regeln ansonsten unnötig komplex würden.

Wortausdrücke und wird Hülle genannt. Sie beschreibt die Sprache, die durch beliebig häufige Einsetzung von t in t an der Stelle von $\Box$ entsteht. Im oben gezeigten Teilausdruck $reclist(a, \Box_1)^{*,\Box_1}$ entsteht dadurch die Sprache

$$\{\Box_1, reclist(a, \Box_1), reclist(a, reclist(a, \Box_1)), \dots\}.$$

Falls wir $t = reclist(a, \Box_1)$ kein Mal in $\Box_1$ einsetzen, erhalten wir einfach nur $\{\Box_1\}$. Dies entspricht dem leeren Wort, das wir bei Sternbildung im Fall von regulären Wortausdrücken erhalten. Setzen wir t ein Mal ein, so erhalten wir einfach t. Setzen wir t mehrfach ein, so landen wir im ersten Schritt wieder bei $\{\Box\}$, danach bei t, und in allen weiteren Schritten wird t erneut in den Ausdruck des vorherigen Schritts eingesetzt. Die Hüllenoperation $t^{*,\Box}$ ist die Vereinigung der Sprachen, die durch das n-malige Einsetzen von t in t anstelle von $\Box$ entsteht, für $n \geq 0$.

Der Ausdruck $t._{\Box}s$ repräsentiert gewissermaßen die Konkatenation zweier Bäume, obwohl wie bereits erwähnt ein Nacheinanderschreiben zweier Bäume nicht möglich ist. Stattdessen arbeitet der Ausdruck als Substitution $t\{\Box \leftarrow s\}$, d. h. s wird anstelle von $\Box$ in t eingesetzt. Der Ausdruck $reclist(a, \Box_1).{}_{\Box_1} nil$ ergibt also die Sprache $\{reclist(a, nil)\}$. Bevor wir allerdings die Definition regulärer Baumausdrücke vorstellen können, müssen wir unsere bisherige Definition von Substitution (vgl. Seite 6) auf Sprachen verallgemeinern und die Begriffe der Konkatenation und der Hülle formal einführen. Zum Schluss der Definitionen werden alle neuen Begriffe in einem Gesamtbeispiel demonstriert.

Definition 7 (Comon u. a. 2008, S. 55–56)**.** *Sei $\mathcal{K}$ eine endliche, von $\mathcal{F}$ disjunkte Menge von 0-stelligen Platzhaltern. Gegeben einen Baum $t \in T(\mathcal{F} \cup \mathcal{K})$, Platzhalter $\Box_1, \dots, \Box_n \in \mathcal{K}$ und Sprachen $L_1, \dots, L_n \subseteq T(\mathcal{F} \cup \mathcal{K})$. Eine* ***Baumersetzung*** *von $\Box_1, \dots, \Box_n$ durch $L_1, \dots, L_n$ in t, geschrieben $t\{\Box_1 \leftarrow L_1, \dots, \Box_n \leftarrow L_n\}$, ist die durch die folgenden Gleichungen definierte Sprache:*

- $\Box_i\{\Box_1 \leftarrow L_1, \dots, \Box_n \leftarrow L_n\} = L_i$ *für* $1 \leq i \leq n$
- $a\{\Box_1 \leftarrow L_1, \dots, \Box_n \leftarrow L_n\} = \{a\}$ *für alle 0-stelligen* $a \in \mathcal{F} \cup \mathcal{K}$ *für die gilt:* $a \neq \Box_1, \dots, a \neq \Box_n$
- $f(s_1, \dots, s_n)\{\Box_1 \leftarrow L_1, \dots, \Box_n \leftarrow L_n\} =$
  $\{f(t_1, \dots, t_n) \mid t_i \in s_i\{\Box_1 \leftarrow L_1, \dots, \Box_n \leftarrow L_n\}\}$

Wir verallgemeinern diese Ersetzung nun auf Baumsprachen: Seien $L, L_1, \dots, L_n \subseteq T(\mathcal{F} \cup \mathcal{K})$ Sprachen, dann definieren wir $L\{\Box_1 \leftarrow L_1, \dots, \Box_n \leftarrow L_n\}$ als die Menge $\bigcup_{t \in L} t\{\Box_1 \leftarrow L_1, \dots, \Box_n \leftarrow L_n\}$.

Zur Definition der Baumersetzung werden zunächst die zu verarbeitenden Terme aus $T(\mathcal{F} \cup \mathcal{K})$ in drei unterschiedliche Kategorien unterteilt: alleinstehende Platz-

halter, alleinstehende 0-stellige Symbole, die nicht zu den verwendeten Platzhaltern gehören, und mehrstellige Terme. Für jede dieser Kategorien wird bestimmt, was das Ergebnis der Baumersetzung auf ihnen ist. In jedem Fall ist wichtig, dass eine Baumersetzung immer eine Sprache zum Ergebnis hat, und nie einen Term. Nun zu den einzelnen Kategorien: Da in einem Platzhalter $\Box_i$ eben nur $\Box_i$ enthalten ist, kann nur L_i eingesetzt werden – die resultierende Sprache ist also L_i (siehe 1. Gleichung in obiger Definition). Ein sonstiger 0-stelliger Term, der kein verwendeter Platzhalter ist, eignet sich erst gar nicht für eine Einsetzung. In diesem Fall definieren wir die resultierende Sprache derart, dass sie nur den besagten 0-stelligen Term enthält (siehe 2. Gleichung). Handelt es sich bei unserem Term um einen mehrstelligen Term, so behalten wir den Funktor und verarbeiten die Argumente rekursiv weiter (siehe 3. Gleichung).

Den letzten Schritt der Definition stellt die Erweiterung auf Sprachen statt. Wie schon für mehrstellige Terme, gehen wir hier rekursiv vor: Jeder Term aus der Sprache, auf der die Ersetzung durchgeführt werden soll, wird für sich genommen der Baumersetzung unterzogen. Das Ergebnis der Ersetzung auf der Sprache ist die Vereinigung all der einzelnen Baumersetzungen.

Die zweite Voraussetzung zur formalen Definition von regulären Baumausdrücken ist die Operation der Konkatenation, die wir nun vorstellen.

Definition 8 (Comon u. a. 2008, S. 56)**.** *Seien L und M Sprachen über $T(\mathcal{F} \cup \mathcal{K})$ und $\Box \in \mathcal{K}$, dann ist die **Konkatenation** von L und M durch $\Box$, geschrieben $L._{\Box}M$, die Baumsprache, die man durch Ersetzung aller Vorkommen von $\Box$ in L durch M erhält: $L._{\Box}M = \bigcup_{t \in L} t\{\Box \leftarrow M\}$.*

Die Konkatenation zweier Sprachen L und M ist also nichts weiter als die Einsetzung von M in L, vorausgesetzt es gibt einen passenden Platzhalter in L, an dessen Stelle M treten kann. Der Platzhalter kann in L auch mehrfach auftreten. Es fehlt nun nur noch die Definition der Hülle einer Baumsprache.

Definition 9 (Comon u. a. 2008, S. 56)**.** *Sei $L \subseteq T(\mathcal{F} \cup \mathcal{K})$ eine Sprache und $\Box \in \mathcal{K}$, dann definieren wir die Folge $L^{n,\Box}$ wie folgt:*

- $L^{0,\Box} = \{\Box\}$
- $L^{n+1,\Box} = L^{n,\Box} \cup L._{\Box}L^{n,\Box}$

*Die **Hülle** $L^{*,\Box}$ von L ist die Vereinigung aller $L^{n,\Box}$ für $n \geq 0$: $L^{*,\Box} = \bigcup_{n \geq 0} L^{n,\Box}$.*

Die Definition funktioniert rekursiv, d. h. es wird zunächst der Basisfall $L^{0,\Box}$ bestimmt: Setzt man L kein Mal in sich selbst ein, so bleibt nur der verwendete Platzhalter $\Box$ über. Danach geht es schrittweise weiter, wir setzen L also ein Mal, zwei

Mal, und so weiter in sich selbst ein. Jeder Schritt basiert direkt auf dem vorherigen Schritt. Wenn wir also L ein Mal in sich selbst einsetzen wollen, so müssen wir es erst kein Mal einsetzen. Dieses Ergebnis setzen wir dann wieder in L ein und vereinigen beides. In jedem weiteren Schritt wird L ein weiteres Mal in sich selbst eingesetzt.

Beispiel 1.1. Sei $\mathcal{F} = \{nil, reclist(,)\}$, $\mathcal{K} = \{\Box\}$ und $L = \{nil, reclist(a, \Box)\} \subset L(\mathcal{F} \cup \mathcal{K})$. Berechnen wir nun $L^{0,\Box}$, $L^{1,\Box}$ und $L^{2,\Box}$.

$$
\begin{aligned}
L^{0,\Box} &= \{\Box\} \\
L^{1,\Box} &= L^{0,\Box} \cup L._{\Box} L^{0,\Box} = \{\Box, nil, reclist(a, \Box)\} \\
L^{2,\Box} &= L^{1,\Box} \cup L._{\Box} L^{1,\Box} \\
&= \{\Box, nil, reclist(a, \Box), reclist(a, nil), reclist(a, reclist(a, \Box))\}
\end{aligned}
$$

$\Diamond$

Mit diesen Werkzeugen ausgestattet, können wir nun die Menge der regulären Baumausdrücke definieren:

Definition 10 (Comon u. a. 2008, S. 57–58)**.** *Die Menge $Regexp(\mathcal{F}, \mathcal{K})$ von **regulären Baumausdrücken** über $\mathcal{F}$ und $\mathcal{K}$ ist die kleinste Menge, so dass gilt:*

- $\emptyset \subseteq Regexp(\mathcal{F}, \mathcal{K})$
- *Falls $a \in \mathcal{F}_0 \cup \mathcal{K}$ eine Konstante ist, gilt $a \in Regexp(\mathcal{F}, \mathcal{K})$*
- *Falls $f \in \mathcal{F}_n$, $n \geq 1$ und $E_1, \ldots, E_n$ reguläre Baumausdrücke aus $Regexp(\mathcal{F}, \mathcal{K})$ sind, ist auch $f(E_1, \ldots, E_n)$ ein regulärer Baumausdruck aus $Regexp(\mathcal{F}, \mathcal{K})$*
- *Falls E_1, E_2 reguläre Baumausdrücke aus $Regexp(\mathcal{F}, \mathcal{K})$ sind, ist auch $(E_1 + E_2)$ ein regulärer Baumausdruck aus $Regexp(\mathcal{F}, \mathcal{K})$*
- *Falls E_1, E_2 reguläre Baumausdrücke aus $Regexp(\mathcal{F}, \mathcal{K})$ sind und $\Box$ ein Element von $\mathcal{K}$ ist, ist auch $E_1._{\Box} E_2$ ein regulärer Baumausdruck aus $Regexp(\mathcal{F}, \mathcal{K})$*
- *Falls E ein regulärer Baumausdruck aus $Regexp(\mathcal{F}, \mathcal{K})$ und $\Box$ ein Element von $\mathcal{K}$ ist, dann ist auch $E^{*,\Box}$ eine regulärer Baumausdruck aus $Regexp(\mathcal{F}, \mathcal{K})$*

Jeder reguläre Baumausdruck denotiert eine reguläre Baumsprache gemäß den folgenden Regeln:[4]

- $L(\emptyset) = \emptyset$

[4]Wir weichen an dieser Stelle bewusst von der in Comon u. a. (2008) verwendeten Notation mit $[\![\ldots]\!]$ ab, da die Notation $L(\ldots)$ im Bereich der formalen Wortsprachen bereits etabliert ist.

- $L(a) = \{a\}$
- $L(f(E_1, \ldots, E_n)) = \{f(s_1, \ldots, s_n) \mid s_1 \in L(E_1), \ldots, s_n \in L(E_n)\}$
- $L(E_1 + E_2) = L(E_1) \cup L(E_2)$
- $L(E_1 ._\Box E_2) = L(E_1)\{\Box \leftarrow E_2\}$
- $L(E^{*,\Box}) = L(E)^{*,\Box}$

1.4 Endliche Baumautomaten

Es gibt einige Ähnlichkeiten zwischen endlichen Wortautomaten und endlichen Baumautomaten: beide verfügen über eine Menge von Zuständen, von denen einige als Endzustände gekennzeichnet werden. Zusätzlich gibt es eine Menge von möglichen Übergängen zwischen Zuständen in den Automaten. Im Falle von Wortautomaten sind die Übergänge einfach ein Tripel aus aktuellem Zustand, Eingabesymbol und neuem Zustand. Betrachten wir jedoch Baumautomaten, werden die Übergänge etwas komplexer. Der „aktuelle Zustand“ bzw. der „neue Zustand“ sind nun Terme $t \in T(\mathcal{F} \cup Q)$, d. h. Basisterme, die zusätzlich zu Funktoren auch Zustände beinhalten können. Zustände werden als einstellige Funktoren behandelt und tragen als einziges Argument einen Term $t \in T(\mathcal{F})$. Ein zulässiger Übergang ist beispielsweise $f(q(x), q(y)) \to q(f(x, y))$, falls $f \in \mathcal{F}_2$, $x, y \in \mathcal{X}$ und $q \in Q$. Ein direktes Äquivalent zu einem Eingabesymbol gibt es bei Baumautomaten nicht.

Wie also arbeiten Baumautomaten? Ein Baumautomat verarbeitet Basisterme. Betrachten wir diese Terme als Bäume, so beginnt der Baumautomat an den Blättern und arbeitet sich in einem Bottom-Up-Prozess Stück für Stück in Richtung des Wurzelknotens. In jedem Schritt werden Teiltermen des Eingabebaums Zustände zugewiesen. Ein Baumautomat akzeptiert eine Eingabe, falls zum Schluss dem Wurzelknoten ein Endzustand zugewiesen wurde. Den gesamten Prozess bezeichnen wir als **Reduktion**. Im Unterschied zu Wortautomaten kennen Baumautomaten keinen Startzustand, sondern beginnen mit der Anwendung von Regeln der Form $a \to q(a)$, die Konstanten Zustände zuweisen. Andere Regelanwendungen sind zu Beginn gar nicht möglich, da das Regelformat für mehrstellige Terme $f(q_1, \ldots, q_n)$ verlangt, dass deren Argumente $q_1, \ldots, q_n$ von der Form $q(x)$ sind, also als Wurzel einen Zustand tragen. Da der Eingabebaum aber keine Zustände enthalten kann, weil es sich um einen Basisterm über $T(\mathcal{F})$ handelt, müssen zunächst den Blättern Zustände zugewiesen werden, damit anschließend mehrstellige Terme verarbeitet werden können.

1.4.1 Nichtdeterministische Baumautomaten

Wir wollen nun die nichtdeterministischen Baumautomaten formal definieren, bevor wir uns einem vollständigen Beispiel widmen.

Definition 11 (Comon u. a. 2008, S. 20)**.** *Ein* ***nichtdeterministischer endlicher Baumautomat*** *(NFTA, nondeterministic finite tree automaton) über $\mathcal{F}$ ist ein 4-Tupel $\mathcal{A} = (Q, \mathcal{F}, Q_f, \Delta)$. Q ist eine Menge von (einstelligen) Zuständen, $Q_f \subseteq Q$ ist eine Menge von Endzuständen, und Δ ist eine Menge von Übergangsregeln der Form*

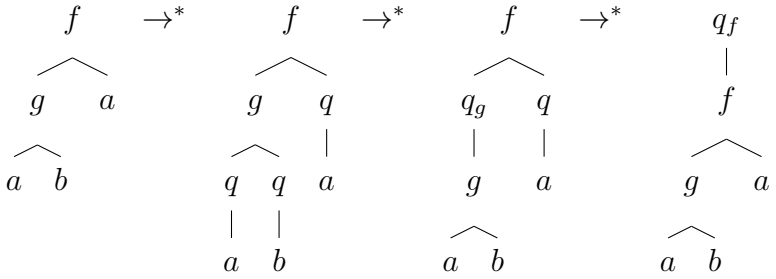
$$f(q_1(x_1), \ldots, q_n(x_n)) \to q(f(x_1, \ldots, x_n))$$

mit $n \geq 0$, $f \in \mathcal{F}_n$, $q, q1, \ldots, q_n \in Q$ und $x_1, \ldots, x_n \in \mathcal{X}$.

Beispiel 1.2. Sei $\mathcal{A} = (Q, \mathcal{F}, Q_f, \Delta)$ ein nichtdeterministischer endlicher Baumautomat. $Q = \{q, q_f, q_g\}$, $\mathcal{F} = \{f(,), g(,), a, b\}$ und $Q_f = \{q_f\}$. Sei Δ die folgende Menge von Übergängen:

$$\begin{aligned} a &\to q(a) \\ b &\to q(b) \\ f(q_g(x), q(y)) &\to q_f(f(x, y)) \\ g(q(x), q(y)) &\to q_g(g(x, y)) \end{aligned}$$

Eine Reduktion des Terms $t = f(g(a, b), a)$ spielt sich dann wie folgt ab:



Der erste Übergang[5] weist den Blättern des Baums den Zustand q zu. Der zweite Übergang weist dem Teilbaum $t|_1$ anhand der Regel $g(q(x), q(y)) \to q_g(g(x, y))$ den Zustand q_g zu. Der letzte Übergang weist dem Wurzelknoten mittels der Regel

[5] Eigentlich sind es drei separate Übergänge, nämlich für jedes Blatt einen, aber zur Einfachheit der Darstellung sind diese zusammengefasst.

$f(q_g(x), q(y)) \to q_f(f(x, y))$ den Endzustand q_f zu, d. h. der Baumautomat akzeptiert die Eingabe. $\Diamond$

Bisher wurde nur an einem Beispiel gezeigt, wie die Übergänge in einem Baumautomaten funktionieren. Im Folgenden wird gezeigt, wie sich die Übergangsrelation formal definieren lässt.

Definition 12 (Comon u. a. 2008, S. 20)**.** *Sei $\mathcal{A} = (Q, \mathcal{F}, Q_f, \Delta)$ ein nicht-deterministischer endlicher Baumautomat über $\mathcal{F}$. Zwei Terme $t, t' \in T(\mathcal{F} \cup Q)$ stehen in der Übergangsrelation $\to_{\mathcal{A}}$, falls die folgenden Bedingungen erfüllt sind:*

- $\exists C \in \mathcal{C}(\mathcal{F} \cup Q), \exists u_1, \ldots, u_n \in T(\mathcal{F})$,
- $\exists f(q_1(x_1), \ldots, q_n(x_n)) \to q(f(x_1, \ldots, x_n)) \in \Delta$,
- $t = C[f(q_1(u_1), \ldots, q_n(u_n))]$,
- $t' = C[q(f(u_1, \ldots, u_n))]$.

Die reflexive, transitive Hülle von $\to_{\mathcal{A}}$ heißt $\to^*_{\mathcal{A}}$. Ein Basisterm $t \in T(\mathcal{F})$ wird von $\mathcal{A}$ akzeptiert, falls $t \to^*_{\mathcal{A}} q(t)$ und $q \in Q_f$. Falls $\mathcal{A}$ aus dem Kontext bekannt ist, schreiben wir einfach $\to$.

Ein Übergang von einem Term $t \in T(\mathcal{F} \cup Q)$ zu einem Term $t' \in T(\mathcal{F} \cup Q)$ kann als Ersetzung verstanden werden, bei der ein Teil von t so ersetzt wird, dass t' entsteht. Diese Ersetzung erfolgt mit Hilfe eines Kontexts, der markiert, an welcher Stelle die Ersetzung durchgeführt werden soll. Der Kontext trägt dazu genau eine Variable, an deren Stelle dann die linke bzw. rechte Seite einer Ersetzungsregel $\delta \in \Delta$ eingesetzt werden kann. Falls t dem Kontext nach Einsetzung der linken Regelseite entspricht, und t' dem Kontext nach Einsetzung der rechten Regelseite entspricht, gilt $t \to_{\mathcal{A}} t'$. Natürlich muss die Stelligkeit des Eingabeterms mit der Stelligkeit der linken Regelseite übereinstimmen, sprich falls der Eingabeterm $f(q(a))$ lautet, kann keine Regel der Form $f(q(x), q(y)) \to \ldots$ angewendet werden. Die Variablen in der Regel werden durch die tatsächlichen Werte an den jeweiligen Positionen ersetzt, d. h. anstelle von x_n wird u_n eingesetzt, für $1 \leq n \leq \mathcal{A}rity(f)$. Die reflexive, transitive Hülle $\to^*_{\mathcal{A}}$ der Übergangsrelation kann zur Vereinfachung benutzt werden, da sie konsekutive Übergänge zusammenfasst (Transitivität). So lässt sich beispielsweise formulieren, dass ein Baumautomat eine Eingabe akzeptiert, falls $t \to^*_{\mathcal{A}} q(t)$ mit $q \in Q_f$ gilt, also wenn man Übergänge findet, so dass man schrittweise von t nach $q(t)$ kommt und q gleichzeitig ein Endzustand ist.

Die Baumsprache $L(\mathcal{A})$, die von $\mathcal{A}$ **erkannt** wird, ist die Menge aller Basisterme, die $\mathcal{A}$ erkennt. Eine Menge L von Basistermen heißt **erkennbar**, falls $L = L(\mathcal{A})$

für einen NFTA $\mathcal{A}$. Zwei NFTA heißen äquivalent, falls sie die selbe Baumsprache akzeptieren.

Die bisherige Definition der Übergangsregeln lässt sich ohne Beschränkung der Allgemeinheit vereinfachen zu Regeln der Form $f(q_1, \ldots, q_n) \to q$, da die Subterme von $q_1, \ldots, q_n$ ohnehin direkt übernommen werden. Beispiel: Aus $f(q(a), q(b))$ wird unter Anwendung der Regel $f(q(x), q(y)) \to q_f(f(x, y))$ der Term $q_f(f(a, b))$, d. h. die Regel bezieht sich lediglich auf den Funktor und die Zustände der Subterme, aber nicht auf den Inhalt der Subterme selbst. Demnach können wir den Inhalt der Subterme ignorieren und lediglich die Zustände betrachten. Die eben genannte Regel lässt sich dann schreiben als $f(q, q) \to q_f$. Unter Annahme dieser Vereinfachung akzeptiert ein Baumautomat eine Eingabe, falls $t \to_{\mathcal{A}}^* q$ mit $q \in Q_f$. Im weiteren Verlauf der Arbeit wird nur noch diese verkürzte Schreibweise verwendet.

Beispiel 1.3. Grundlage sei der Automat $\mathcal{A}$ aus Beispiel 3.2. Dessen Übergangsregeln Δ lassen sich wie folgt vereinfachen:

$a \to q(a)$	wird zu	$a \to q$
$b \to q(b)$	wird zu	$b \to q$
$f(q_g(x), q(y)) \to q_f(f(x, y))$	wird zu	$f(q_g, q) \to q_f$
$g(q(x), q(y)) \to q_g(g(x, y))$	wird zu	$g(q, q) \to q_g$

$\Diamond$

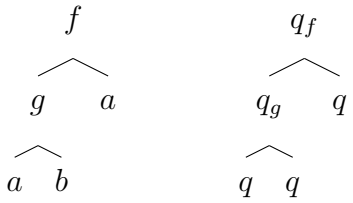
Um nachvollziehen zu können, welche Zustände während einer Reduktion bestimmten Subtermen zugeordnet wurden, ist der Begriff des Laufs nützlich. Ein Lauf weist jeder Position des Eingabeterms den Zustand zu, den er im Zuge einer Reduktion bekäme. So entsteht ein zweiter, vom Eingabeterm unabhängiger Baum, der anstatt Funktoren nur noch Zustände enthält. Von der Struktur betrachtet, unterscheiden sich die beiden Bäume nicht, da jedem Knoten aus dem Eingabebaum genau ein Zustand zugewiesen wird.[6] An diesem Zustandsbaum lässt sich aber gut ablesen, welche Einzelschritte ein Eingabeterm durchläuft, um akzeptiert zu werden.

Definition 13 (Comon u. a. 2008, S. 22)**.** *Sei $t \in T(\mathcal{F})$ ein Basisterm und $\mathcal{A} = (Q, \mathcal{F}, Q_f, \Delta)$ ein NFTA. Ein **Lauf** r von $\mathcal{A}$ ist eine mit Δ kompatible Abbildung $r : \mathcal{P}os(t) \to Q$, d. h. für jede Position $p \in \mathcal{P}os(t)$ gilt: falls $t(p) = f \in \mathcal{F}_n$, $r(p) = q$,*

[6] Die Zustände besitzen dann die gleiche Stelligkeit wie die Funktoren, die im Eingabeterm an ihrer Position stehen.

$r(pi) = q_i$ für alle $i \in \{1, \ldots, n\}$, dann $f(q_1, \ldots, q_n) \to q \in \Delta$. Ein Lauf r von $\mathcal{A}$ auf t ist erfolgreich, falls $r(\varepsilon) \in Q_f$. Ein Basisterm t wird von $\mathcal{A}$ akzeptiert, falls es einen erfolgreichen Lauf r von $\mathcal{A}$ auf t gibt.

Beispiel 1.4. Nehmen wir als Grundlage den Automaten und die gezeigte Reduktion aus Beispiel 3.2. Der Lauf zu der Reduktion findet sich auf der rechten Seite der Abbildung – man sieht dort die Zustände, die jedem Knoten aus dem Eingabebaum im Zuge des Laufs zugeordnet wurden.



$\Diamond$

1.4.2 Nichtdeterministische Baumautomaten mit ε-Übergängen

Gemäß der bisherigen Definition übersetzt jeder Übergang einen Term der Form $f(q_1, \ldots, q_n)$ in einen Zustand q – die Subterme ignorieren wir dabei wie beschrieben. Oft lassen sich Automaten jedoch vereinfachen, falls wir Übergänge zulassen, die keinen Funktor verarbeiten, sondern lediglich einen Zustand gegen einen anderen Zustand austauschen und danach wie gehabt fortfahren. Von Wortautomaten kennen wir diese sogenannten ε-Übergänge bereits und wissen, dass sie zu Automaten ohne ε-Übergänge äquivalent sind. Das gleiche gilt für Baumautomaten, d. h. durch die Hinzunahme von ε-Übergängen gewinnen wir nicht eine höhere Mächtigkeit, sondern vereinfachen lediglich die Art und Weise, wie man bestimmte Baumautomaten formulieren kann. Wir schreiben einen solchen Übergang vereinfacht $q \to q'$ und meinen damit, dass ein Term $q(f(q_1, \ldots, q_n))$ in $q'(f(q_1, \ldots, q_n)))$ übersetzt werden kann.

1.4.3 Deterministische Baumautomaten

Die bisher vorgestellten Baumautomaten heißen *nichtdeterministische* endliche Baumautomaten, da es mehrere Regeln mit gleicher linker Regelseite geben kann. Das

bedeutet, dass es mehrere Regeln geben kann, die auf den gleichen Eingabeterm angewendet werden können. Diese Wahlmöglichkeit führt zur Unklarheit darüber, welches Ergebnis ein Baumautomat gegeben einen bestimmten Eingabeterm erreicht. Im einfachsten Fall gibt es verschiedene erfolgreiche Läufe, man kommt also auf unterschiedlichen Wegen zum gleichen Ergebnis, nämlich dass eine Eingabe akzeptiert wird. Andererseits kann es Läufe geben, die nicht erfolgreich sind, andere aber schon. In der Praxis werden solche Probleme dann durch Backtracking gelöst, d. h. ein Programm bestimmt, an welchen Stellen eines Laufs eine nichtdeterministische Entscheidung gefällt wurde, und ändert den Lauf so lange, bis es keine weiteren Optionen mehr gibt. Wird durch diesen Prozess kein erfolgreicher Lauf erreicht, so kann man sagen, dass die Eingabe nicht akzeptiert wird.

Alternativ zu dieser Herangehensweise kann man auch mit deterministischen endlichen Baumautomaten (DFTA) arbeiten, die erst gar keine Doppeldeutigkeiten in den Übergangsregeln zulassen. Wenn also eine Regel auf einen Eingabeterm zutrifft, dann ist es die einzige solche Regel. Umgekehrt kann man sagen, dass falls in einem Lauf keine Regel mehr anwendbar ist, der Eingabeterm nicht akzeptiert wird, womit die Notwendigkeit für Backtracking entfällt. Außerdem lässt sich zeigen, dass jeder NFTA in einen äquivalenten DFTA umgewandelt werden kann. Ein entsprechender Beweis und der zugehörige Algorithmus finden sich in Comon u. a. (2008, S. 25–27). Betrachten wir nun abschließend die formale Definition von deterministischen endlichen Baumautomaten.

Definition 14 (Comon u. a. 2008, S. 25)**.** *Ein Baumautomat $\mathcal{A} = (Q, \mathcal{F}, Q_f, \Delta)$ ist ein **deterministischer endlicher Baumautomat** (DFTA, deterministic finite tree automaton), falls es keine zwei Regeln mit gleicher linker Regelseite und keine ε-Übergänge gibt. Ein DFTA ist deshalb nicht ambig, d. h. für alle Basisterme $t \in T(\mathcal{F})$ gibt es jeweils höchstens ein $q \in Q_f$, so dass $t \to_{\mathcal{A}}^{*} q$.*

In diesem Kapitel haben wir gesehen, was Baumsprachen sind und wie sie sich von Wortsprachen unterscheiden. Außerdem haben wir gesehen, wie sich Grammatiken, reguläre Ausdrücke und endliche Automaten auf Baumsprachen übertragen lassen. Im folgenden Abschnitt wollen wir nun versuchen einen Baumautomaten zu konstruieren, der in der Lage ist, Possessionskonstruktionen in Dependenzbäumen zu finden.

Literatur

Comon, Hubert u. a. (2008). *Tree Automata Techniques and Applications*. URL: `http://tata.gforge.inria.fr/`.

Index