Sprachen und Automaten

P. H. Schmitt

Winter 1994/95

Vorwort

Dieses Skriptum umfaßt im wesentlichen den Inhalt der Spezialvorlesung *Sprachen und Automaten*, die ich im Wintersemester 1994/95 an der Fakultät für Informatik der Universität Karlsruhe gelesen habe. Zum letzten Mal war diese Vorlesung im Wintersemester 1990/91 gehalten worden. Aus diesem großen zeitlichen Abstand, aufgrund von Veränderungen im Curriculum unserer Fakultät und natürlich auch aufgrund meiner persönlichen Vorlieben erklärt sich der umfangreiche Wechsel in den behandelten Themen.

Die elementaren Tatsachen der Automatentheorie sowie der Chomsky-Hierarchie werden den Studierenden schon während des Vordiplomstudiums vermittelt. Diese Gegenstände sind in diesem Skriptum nur noch einmal kurz zusammengefaßt und nur dort ausführlich dargestellt, wo an späterer Stelle ein direkter Rückgriff notwendig wird. In dieser Zusammenfassung sind die Abgeschlossenheitseigenschaften der elementaren Sprachklassen etwas zu kurz weggekommen. Aber das beeinträchtigt nur die Vollständigkeit des Skriptes, denn dieses Material läßt sich leicht in den Textbüchern nachlesen.

Neu aufgenommen wurden die Abschnitte über ω-Wörter, das heißt unendlich langer Wörter, die in der Modellierung zeitabhängiger Schaltungen und in der Temporallogik immer mehr Bedeutung gewinnen.

Wesentlich erweitert gegenüber dem letzten Vorlesungskript wurde die Behandlung von LR(k)- und LALR(k)-Grammatiken.

In dem Bemühen das Skriptum den Hörern der Vorlesung möglichst rasch zur Verfügung zu stellen, habe ich in den lezten beiden Abschnitte über LALR(1)-Grammatiken und über das Parsererzeugungsprogramm *Bison* auf einen ausformulierten Text verzichtet und auch nicht mehr das gesamte in der Vorlesung behandelte Material aufnehmen können.

Ich bedanke mich an dieser Stelle sehr herzlich bei den Hörern der Vorlesung, die durch ihre aufmerksame und engagierte Teilnahme eine hervorragende Atmossphäre geschaffen haben, in der häufig ein lebhafter Dialog an die Stelle einseitigen *Vorlesens* treten konnte. Ein Glücksfall, der in den Zeiten der Massenuniversität selten geworden ist.

Peter H. Schmitt Karlsruhe im Februar 1995

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Die Automatenhierarchie

In genauer Entsprechung zu den Sprachklassen der Chomsky-Hierarchie gibt es eine Hierarchie von Maschinenmodellen, die in diesem Kapitel beschrieben werden soll.

1.1 Endliche Automaten

1.1.1 Deterministische endliche Automaten

Definition 1 (Endlicher Automat (finite-state-automaton))
Ein endlicher Automat ist gegeben durch

- *eine endliche Menge S von Zuständen*
- *eine Menge VT von terminalen Zeichen*
- *einer Übergangsfunktion $\delta : S \times VT \to S$*

Außerdem ist ein Element $s_0 \in S$ als Anfangszustand ausgezeichnet und eine nichtleere Teilmenge $S_1 \subseteq S$ als Menge von Endzuständen festgelegt.

Jeder endliche Automat EA akzeptiert eine Menge von Wörtern $L(EA)$ auf die folgende Weise:

Ein gegebenes Wort $w = a_1 \dots a_k$ denken wir uns auf das Eingabeband des EA geschrieben. Im Anfangszustand s_0 liest der Automat den ersten Buchstaben a_1 und geht in den Zustand $\delta(s_0, a_1) = s_i$ über. In diesem Zustand wird nur der zweite Buchstabe von w eingelesen und der nächste Zustand $\delta(s_i, a_2) = s_j$ erreicht. Das Wort gehört zu $L(EA)$, wird also von EA akzeptiert, wenn nach Einlesen des letzten Buchstabens a_k von w ein Endzustand aus S_1 erreicht wird.

Formal wird $L(EA)$ definiert, indem man die Übergangsfunktion δ zu einer Funktion

$$\delta : S \times VT^* \rightarrow S$$

fortsetzt.

Definition 2 (Fortsetzung von δ)
Der Funktionswert $\delta(s, w)$ wird induktiv über die Länge von w wie folgt definiert:

$$\begin{array}{lcl} \delta(s, \varepsilon) & = & s \\ \delta(s, aw_1) & = & \delta(s', w_1) \textit{ wobei } \delta(s, a) = s' \end{array}$$

Definition 3 (L(EA))

$$L(EA) = \{w \in VT^* \mid \delta(s_0, w) \in S_1\}$$

Varianten:

Gelegentlich wird in Beispielen die Übergangsfunktion nicht für alle Paare (s, a) definiert. Wird während der Abarbeitung eines Wortes w eine Situation (s, a) erreicht, für die $\delta(s, a)$ nicht definiert ist, so gilt w als nicht akzeptiert. Ein endlicher Automat, so daß $\delta(s, a)$ für alle $s \in S$ und $a \in VT$ definiert ist, heißt ein **vollständiger endlicher Automat**.

Beispiel 1 *Der endliche Automat N_{bba} ist gegeben durch*

$$\begin{array}{lcl} S & = & \{s_0, s_1, s_2, s_3\} \\ V & = & \{a, b\} \\ S_1 & = & \{s_0, s_1, s_2\} \end{array}$$

$L(N_{bba}) = \{w \in \{a, b\}^* \mid bba \textit{ ist kein Teilwort von } w\}$

Beispieldurchlauf für das Eingabewort *bbba*

Das Wort *bba* wird also von N_{bba} akzeptiert.

1.1.2 Nichtdeterministische endliche Automaten

Definition 4 (Nichtdeterministische endliche Automaten)
Ein nichtdeterministischer endlicher Automat wird durch die folgenden Bestimmungsstücke gegeben:

- *eine endliche Menge S von Zuständen,*
- *ein Alphabet V,*
- *ein Anfangszustand $s_0 \in S$,*
- *eine Übergangsfunktion $\delta : S \times V \to Pot(S)$,*
- *eine Menge $F \subseteq S$ von Finalzuständen,*

Die Änderung gegenüber den deterministischen endlichen Automaten besteht also darin, daß die Übergangsfunktion δ als Werte Mengen von Zuständen annimmt:

Die Fortsetzung von δ zu einer Funktion

$$\delta : S \times V^* \to Pot(S)$$

wird jetzt wie folgt definiert:

$$\begin{aligned}
\delta(s, \varepsilon) &= \{s\} \\
\delta(s, aw_1) &= \{s' : \text{ es gibt } s_1 \in S \text{ mit } s_1 \in \delta(s, a) \\
&\quad\ \text{und } s' \in \delta(s_1, w_1)\}
\end{aligned}$$

Die von einem nichtdeterministischen endlichen Automaten NEA akzeptierte Sprache $L(NEA)$ wird jetzt durch

$$L(NEA) = \{w \in V^* : \delta(s_0, w) \cap F \neq \emptyset\}$$

definiert.

Varianten: In der angegebenen Definition kann ein nichtdeterministischer endlicher Akzeptor im Zustand s, ohne einen Buchstaben zu lesen, nur nach s übergehen, $\delta(s, \varepsilon) = \{s\}$. Häufig begegnet man der Variante, die für $\delta(s, \epsilon)$ eine beliebige Teilmenge von S zuläßt. Übergänge von s zu $s_1 \in \delta(s, \varepsilon)$ nennt man dann spontane Übergänge.

Definition 5 (Endliche Automaten mit spontanen Übergängen)
Ein endlicher Automat mit spontanten Übergängen wird durch die folgenden Bestimmungsstücke gegeben:

- *eine endliche Menge S von Zuständen,*
- *ein Alphabet V,*
- *ein Anfangszustand $s_0 \in S$,*
- *eine Menge $F \subseteq S$ von Finalzuständen,*
- *eine Übergangsfunktion $\delta : S \times (V \cup \{\varepsilon\}) \to Pot(S)$*

Um die Fortsetzung der Transitionsfunktion δ für diese Variante endlicher Automaten zu beschreiben benötigt man die folgende Hilfsfunktion:

Definition 6 *Sei $A = (S, V, s_0, \delta, F)$ ein endlicher Automat mit spontanen Übergängen, dann ist die Funktion*

$$\varepsilon - cl : Pot(S) \to Pot(S)$$

für $I \subseteq S$ definiert als:

ε-cl(I) ist die kleinste Teilmenge $J \subseteq S$ mit

1. *$I \subseteq J$*
2. *für alle $s \in J$ gilt $\delta(s, \varepsilon) \subseteq J$.*

*Die Bezeichnung ε-cl soll an ε-*closure *erinnern.*

Die Fortsetzung von

$$\delta : S \times (V \cup \{\varepsilon\}) \to Pot(S)$$

zu

$$\bar{\delta} : S \times V^* \to Pot(S)$$

kann jetzt definiert werden als:

$$\begin{array}{lcl}\bar{\delta}(s,\varepsilon) & = & \varepsilon\text{-}cl(\delta(s,\varepsilon)) \\ \bar{\delta}(s,w_1 a) & = & \varepsilon\text{-}cl(\{s' : \text{ es gibt } s_1 \in \bar{\delta}(s_1,w_1) \text{ so, daß } s' \in \delta(s,a)\})\end{array}$$

Der folgende Satz ist von grundlegender Bedeutung für die Theorie endlicher Automaten und wird auch in allen einführenden Texten behandelt. Da wir später, auch auf Details des Satzes, zurückgreifen wollen, ist er auch hier noch einmal explizit aufgeführt.

Satz 1
Zu jedem nichtdeterministischen endlichen Automaten $A = (S, V, s_0, \delta, F)$ gibt es einen deterministischen endlichen Automaten $B = (Q, V, q_0, \Delta, G$ mit

$$L(A) = L(B)$$

Dabei kann A spontane Übergänge enthalten und muß auch nicht vollständig sein.

Beweis:
Die neuen Zustände sind Teilmengen von alten Zuständen, genauer

$$\begin{array}{lcl}Q & = & \{X \subseteq S \mid \varepsilon - cl(X) = X\} \\ q_0 & = & \varepsilon - cl(\{s_0\}) \\ \Delta(X,a) & = & \varepsilon - cl(\bigcup_{s \in X} \delta(s,a)) \\ G & = & \{X \in Q \mid F \cap X \neq \emptyset\}\end{array}$$

Offensichtlich ist B deterministisch.
Man zeigt nun leicht, durch Induktion über die Länge des Wortes $w \in V^*$,

$$\bar{\delta}(s_0, w) = \Delta(q_0, w)$$

■

1.1.3 Reguläre Ausdrücke

Neben den Operationen, die für beliebige Mengen erklärt werden können, wie Vereinigung ($\cup$), Durchschnitt ($\cap$), relative Komplementbildung ($\setminus$) betrachtet wir hier Operationen die nur auf Mengen von Wörtern Sinn machen:

Definition 7 (Operationen mit Wortmengen)
Seien $L, L_1, L_2 \subseteq V^*$.

1. $L_1 L_2 = \{w_1 w_2 \mid w_1 \in L_1, w_2 \in L_2\}$,
2. $L^* = \{w_1 \dots w_n \mid n \geq 0, w_i \in L\}$

Manche Autoren verwenden für die Konkatenation von Wörtern w_1, w_2 und Sprachen L_1, L_2 die Notation $w_1 \cdot w_2$ bzw. $L_1 \cdot L_2$

Definition 8 (Reguläre Ausdrücke)
Die Menge Reg_V der regulären Ausdrücke über einem Alphabet V wird induktiv definiert durch:

1. $\emptyset \in Reg_V$,
2. $\varepsilon \in Reg_V$,
3. *für jedes* $a \in V$ *ist* $a \in Reg_V$,
4. *für* $t \in Reg_V$ *gilt auch* $(t)^* \in Reg_V$,
5. *für* $t_1, t_2 \in Reg_V$ *gilt auch* $(t_1 t_2) \in Reg_V$ *und* $(t_1 + t_2) \in Reg_V$.

Definition 9 (Semantik regulärer Ausdrücke)
Durch die folgende Vorschrift wird jedem regulären Ausdruck t über V eine Menge $S(t)$ von Wörtern in V^* zugeordnet.

1. $S(\emptyset) = \emptyset$,
2. $S(\varepsilon) = \{\varepsilon\}$,
3. $S(a) = \{a\}$,

4. $S((t)^*) = S(t)^*$,
5. $S((t_1 t_2)) = S(t_1)S(t_2)$ *und* $S((t_1 + t_2)) = S(t_1 \cup S(t_2)$.

Wir benutzen im folgenden stillschweigend die Assoziativität der Konkatenation und von $+$ um in regulären Ausdrücken Klammern einzusparen, also $(a + b + c)$ anstelle von $((a + b) + c)$.

Satz 2

1. *Für jeden regulären Ausdruck t ist $S(t)$ eine reguläre Sprache.*
2. *Zu jedem endlichen Automaten A gibt es einen regulären Ausdruck t mit $S(t) = L(A)$.*

Beweis:
Für 1. siehe z.B. [Moll *et al.*, 1988], Proposition 2, Seite 34 oder [Hopcroft & Ullmann, 1979], Theorem 2.3, Seite 30.
Für 2. siehe z.B.[Moll *et al.*, 1988], Proposition 12, Seite 38 oder [Hopcroft & Ullmann, 1979], Theorem 2.4, Seite 33.

∎

Reguläre Ausdrücke sind in vielen Bereichen der Informatik ein alltägliches Hilfsmittel geworden. Als ein Beispiel dafür wollen wir etwas ausführlicher auf die Benutzung regulärer Ausdrücke in im GNU Emacs Editor eingehen, siehe z.B. [Stallman, 1988] oder online help für emacs. Während einer Editiersitzung kann man das Kommando `C-M-s` eingeben, worauf in der Kommandozeile `Regexp I-search:` erscheint. Hinter dem Doppelpunkt kann man jetzt einen regulären Ausdruck t in der emacs-Syntax (Abschnitt 13.5 in [Stallman, 1988]) eingeben. Der Kursor springt dann unmittelbar auf die erste, von der aktuellen Kursorposition aus gesehen, Zeichenkette im Textpuffer die in $S(t)$ liegt. Das Kommando `C-s` liefert das nächste Vorkommen einer Zeichenkette in $S(t)$. Abgesehen von der Nützlichkeit dieser Kommandos ist es auch instruktiv zu sehen, wie das theoretische Konzept regulärer Ausdrücke in praktisches umgesetzt wird.
Die erste Frage die zu beantworten ist lautet: Was ist das Alphabet V_{emacs}? Antwort V_{emacs} besteht aus den 256 Zeichen des erweiterten ASCII Zeichensatzes. Dazu gehören auch Zeichen, die man normalerweise nicht auf dem

Bildschirm zu sehen bekommt, z.B. das ASCII Zeichen mit dem Oktalcode 012. Dieses Zeichen heißt LF , für line feed. Man sieht es nicht selbst auf dem Bildschirm, sondern nur seine Auswirkung, daß eben an dieser Stelle eine neue Zeile beginnt. Aber LF kann genau so in einem regulären Ausdruck benutzt werden, wie jedes andere Zeichen aus V_{emacs} auch. An dieser Stelle sollte man wissen, wie nicht-graphische Zeichen von emacs dargestellt werden und wie man sie eingeben kann ohne ein eventuell damit verbundenes Editorkommando auszulösen. Das kann man in [Stallman, 1988]in den Abschnitten 2 und 4.1 nachlesen. Ein interessanteres Problem tritt auf, wenn wir nach Zeichenketten suchen wollen, die mit einem x beginnen auf welches eine geraden Anzahl von y's folgt. Wir würden gern den regulären Ausdruck $x(yy)^*$ eingeben. Die Klammern sind notwendig um den gewünschten regulären Ausdruck von dem ungewünschten $(xyy)^*$ oder $xy(y)^*$ zu unterscheiden. Andererseits sind die beiden Klammern (und) Zeichen in V_{emacs}. Gibt man nach dem Prompt `Regexp I-search:` $x(yy)^*$ ein so findet man alle Ausdrücke der Form $x(yy)\ldots)$ mit einer beliebigen Anzahl schliessender Klammern. Dieses Problem wird gelöst, wie an vielen anderen Stellen in der Informatik auch, in dem man V_{emacs} unterteilt in Sonderzeichen und normale Zeichen. An dieser Stelle geht die Einfachheit des theoretischen Konzepts verloren unter vielfachen Ausnahme- und Sonderregeln, Kompatibilitätsrücksichten und historischen Gewöhnheiten. In der emacs Syntax regulärer Ausdrücke sind zunächst einmal die folgenden neun Zeichen Sonderzeichen:

`$,^,.,*,+,?,[,],\`

Wir betrachten zunächst `\`. Es hat zwei Funktionen. Erstens dient es dazu die Sonderzeichen zu normalen Zeichen zu machen, d.h. für den reguläre Ausdruck `\$` gilt $S(\backslash\$) = \{\$\}$ und konsequenterweise auch $S(\backslash\backslash) = \{\backslash\}$. Zweitens macht `\` aus einigen normalen Zeichen Sonderzeichen. Hier ist eine Liste einiger auf diese Weise gebildeter Sonderzeichen:

`\|`, `\(`, `\)`, `'`, `` ` ``, `\b`, `\B`, `\<`, `\>`, `\w`, `\W`,`\scode`,`\Scode`
und `\z` für jede Ziffer `z`.

Wir wollen nur die wichtigsten dieser Sonderzeichen erklären. Eine vollständige Beschreibung findet man in [Stallman, 1988] in Abschnitt 13.5.

Für reguläre emacs Ausdrücke t_1, t_2 gilt $S(t_1|\backslash t_2) = S(t_1) \cup S(t_2)$, d.h. `\|` in regulären emacs Ausdrücken entspricht dem $+$ in den regulären Ausdrücken aus Definition 8. Die Sonderzeichen `\(` und `\)` dienen der Klammerung. Der oben erwähnte reguläre Ausdruck $x(yy)^*$ wird also in emacs Syntax geschrieben als $x\backslash(yy\backslash)*$. Hiermit ist auch gleich verraten, daß der Stern * in den regulären emacs Ausdrücken dieselbe Bedeutung hat wie in unseren regulären Ausdrücken. Da auch die Konkatenation von regulären emacs Ausdrücken mit der üblichen Semantik möglich ist, entspricht als jedem regulären Ausdruck nach Definition 8 ein regulärer emacs Ausdruck mit derselben Bedeutung. Wir richten im folgenden unser Augenmerk auf die Frage, ob reguläre emacs Ausdrücke vielleicht ausdrucksmächtiger sind als unsere regulären Ausdrücke.
Das Sonderzeichen . besitzt als seine Interpretation $S(.)$ die Menge aller Zeichen mit der einzigen Ausnahme des Zeichens LF. Da kann man in der Syntax aus Definition 8 auch hinschreiben als eine Summe mit 255 Summanden. Die Sonderzeichen `+` und `?` sind wie `*` postfix Operatoren, wobei `+` die übliche Bedeutung, $t+ \equiv t(t)^*$, hat und $t? \equiv (\varepsilon + t)$. Die Sonderzeichen `^` und `$` suchen nach dem leeren Wort am Anfang bzw. am Ende einer Zeile. Genauer gilt für einen regulären emacs Ausdruck t $S(\hat{\ }t) = \{w \in S(t) \mid w\text{kommt am Anfang einer Zeile vor}\}$ und $S(t\$) = \{w \in S(t) \mid w\text{kommt am Ende einer Zeile vor}\}$.
Die Sonderzeichen `\[` und `\]` erlauben eine bequeme Beschreibung endlicher Mengen von Zeichen. Es gilt $S([a_1 \dots a_k]) = \{a_1, \dots, a_k\}$, jedenfalls dann, wenn die a_i keine Sonderzeichen enthalten. Allerdings gelten innerhalb der $[\dots]$ Paare ganz andere Sonderzeichenregeln als bisher. Hier sind nämlich

`]`, `^`, `-`

die einzigen Sonderzeichen. Warum `]` ein Sonderzeichen ist, ist klar. Das Minuszeichen `-` wird benutzt um einen Bereich aufeinanderfolgender Zeichen zu beschreiben, z.B. $S([a - z]) =$ Menger aller Kleinbuchstaben. Schließlich ist $S([\hat{\ }a_1 \dots a_k]) = \{z \in V_{emacs} \mid z \neq a_1 \text{ und } z \neq \dots \text{ und} z \neq a_k\}$. Was macht man, wenn man eines der drei Sonderzeichen in einem $[\dots]$ Paar verwenden möchte? Anstelle von `-` schreibt man `---`, `]` ist keine Sonderzeichen, wenn es unmittelbar nach `[` steht und `^` ist nur dann ein Sonderzeichen, wenn es unmittelbar nach `[` steht. Es gibt noch einige kleinere Subtilitäten, aber auf die brauchen wir hier nicht einzugehen.

Mancher Leser mag sich gefragt haben was in dem oben erwähnten Prompt `Regexp I-search:` wohl das `I` bedeutet? Es steht für *inkrementell*. Die Suche nach der ersten Zeichenkette, die zu dem regulären Ausdruck gehört erfolgt inkrementell. Tippt man als regulären Ausdruck a ein, so springt der Kursor zum ersten im Puffer vorkommenden a. Tippt man jetzt noch ein b dazu, so springt der Kursor zum ersten Vorkommen der Zeichenkette ab. Wie verhält sich die Suche bei Eingabe von $a\backslash|b$? Da waren sich die Implementierer nicht immer einig. In meiner emascs Version beginnt die Suche nach Eingabe von $a\backslash|$ nochmal von vorn, so daß auch die bs gefunden werden, die vor dem ersten a im Text stehen.

1.1.4 Übungsaufgaben

Übungsaufgabe 1.1.1 *Zeigen Sie, daß es zu jedem endlichen Automaten EA einen vollständigen endlichen Automaten EAc gibt mit $L(EA) = L(EAc)$. Kann EAc immer gebildet werden, ohne die Menge der Zustände von EA zu vergrößern?*

Übungsaufgabe 1.1.2 *Zeigen Sie, daß es zu jedem nichtdeterministischen endlichen Automaten NEA mit spontanen Übergängen einen nichtdeterministischen endlichen Automaten NEA_0 ohne spontane Übergänge gibt mit $L(NEA) = L(NEA_0)$. ([Hopcroft, Ullman 79], p. 24.)*

1.2 Kellerautomaten

1.2.1 Deterministische Kellerautomaten

Definition 10 (Deterministische Kellerautomaten) *Ein deterministischer Kellerautomat (push-down-automaton) ist gegeben durch*

- *eine endliche Menge von Zuständen S*
- *ein Eingabealphabet VE*
- *ein Kelleralphabet VK*

- *zwei partielle Übergangsfunktionen* d_1, d_2*:*
  - $d_1 : VK \times S \to VK^* \times S$
  - $d_2 : VK \times S \times VE \to VK^* \times S$

Ein Element $s_0 \in S$ *ist als Startzustand ausgezeichnet, ein Element* $z_0 \in VK$ *ist als Anfangselement festgelegt, eine nichtleere Teilmenge* $S_1 \subseteq S$ *ist als Menge der Endzustände ausersehen, und die folgenden Einschränkungen müssen erfüllt sein:*

d_1 *für alle* s *und* a *muß für* $d_1(z_0, s) = (w, s')$ *bzw.* $d_2(z_0, s, a) = (w, s')$ *der erste Buchstabe von* w *ebenfalls* z_0 *sein.*

d_2 d_1 *und* d_2 *dürfen sich nicht überlappen, d. h. ist* $d_1(z, s)$ *definiert, dann darf für kein* a $d_2(z, s, a)$ *definiert sein, und umgekehrt ist für mindestens ein* a $d_2(z, s, a)$ *definiert, dann darf* $d_1(z, s)$ *nicht definiert sein.*

Ein Kellerautomat arbeitet mit einem Eingabeband und einem Keller. Die Abarbeitung beginnt mit einem Wort w auf dem Eingabeband im Anfangszustand s_0 und dem Buchstaben z_0 im Keller. Die Kellermaschine kann zwei Arten von Operationen ausführen:

1. Ist im Zustand s mit z als oberstem Kellerzeichen $d_1(z, s)$ definiert mit $d_1(z, s) = (w, s')$, dann wird das oberste Kellerelement z gelöscht, w im Keller abgelegt und der Automat geht in den Zustand s' über, ohne einen Buchstaben des Eingabewortes eingelesen zu haben.

2. Ist im Zustand s mit z als oberstem Kellerzeichen und a als nächstem noch nicht gelesenen Buchstaben des Eingabewortes $d_2(z, s, a)$ definiert mit Wort (w, s'), so wird wie unter 1 das Kellerelement z gelöscht und w im Keller abgelegt. Jetzt wird jedoch noch a eingelesen und s' als Nachfolgezustand eingenommen.

Ein Wort w wird von einem deterministischen Kellerautomaten DKA akzeptiert, $w \in L(DKA)$, falls nach dem Einlesen des letzten Buchstabens von w ein Zustand $s \in S_1$ erreicht wird. Welches Wort zu diesem Zeitpunkt im Keller steht, spielt keine Rolle. Wird während der Abarbeitung einmal ein

Zustand s mit z als oberstem Kellerzeichen und a als nächstem einzulesenden Buchstaben erreicht, so daß weder $d_1(z,s)$ noch $d_2(z,s,a)$ definiert ist, so gilt w als nicht akzeptiert.

Zur formalen Beschreibung von $L(DKA)$ brauchen wir den Begriff einer Augenblicksbeschreibung (instantaneous description) von DKA. Diese faßt alle zur Beschreibung der im Laufe der Abarbeitung eines Eingabewortes w gerade erreichten Situation nötige Information zusammen. Eine Augenblicksbeschreibung ist ein Wort der Form

$$w_K s w_E$$

wobei $w_K \in VK^*$, $s \in S$, $w_E \in VE^*$. Der erste Buchstabe von w_K wird stets z_0 sein, w_E kann auch leer sein. Gilt für die Augenblicksbeschreibung

$$w_K s w_E = w_0 z s a w_1$$

$d_1(z,s) = (w_2, s')$, dann wird als nächste Augenblicksbeschreibung

$$w_0 w_2 s' a w_1$$

erreicht. Wir schreiben

$$w_0 z s a w_1 \rightarrow_1 w_0 w_2 s' a w_1$$

Gilt dagegen $d_2(z,s,a) = (w_2, s')$, dann wird als nächste Augenblicksbeschreibung

$$w_0 w_2 s' w_1$$

erreicht. Wir schreiben dafür

$$w_0 z s a w_1 \rightarrow_1 w_0 w_2 s' w_1$$

Die transitive Hülle der Relation $\rightarrow_1$ bezeichnen wird mit $\rightarrow$. Falls erforderlich, mit $\rightarrow_{KA}$.

$$L(KA) = \{w \in VE^* \mid z_0 s_0 w \rightarrow_{KA} w_K s_1 \text{ mit } s_1 \in S_1\}$$

1.2.2 Kellerautomaten

Die Bezeichnung deutet schon an, daß bei Kellerautomaten die nichtdeterministische Version eine größere Rolle spielen wird als die deterministische.

Die Definition nichtdeterministischer Kellerautomaten entsteht aus der Definition deterministischer Kellerautomaten durch die Liberalisierung der Übergangsfunktion:

- $d_1 : (VK \cup e) \times S \rightarrow Pot(VK^* \times S)$
- $d_2 : (VK \cup e) \times S \times VE \rightarrow Pot(VK^* \times S)$

und den Wegfall der einschränkenden Bedingungen d_1 und d_2. Da die Bedingung d_1 nicht mehr sicherstellt, daß der Keller nie leer wird, sind die Übergangsfunktionen d_1, d_2 auch für den leeren Keller e definiert.

Die Definition der unmittelbaren Nachfolgerbeziehung $\rightarrow_1$ zwischen zwei Augenblicksbeschreibungen lautet jetzt:

$w_0 zsaw_1 \rightarrow_1 w_0 w_2 s' aw_1$ genau dann, wenn $(w_2, s') \in d_1(z, s)$

und

$w_0 zsaw_1 \rightarrow_1 w_0 w_2 s' w_1$ genau dann, wenn $(w_2, s') \in d_2(z, s, a)$

Natürlich kann es jetzt zu einer Augenblicksbeschreibung u mehrere Augenblicksbeschreibungen $u_1, u_2, \ldots$ geben mit $u \rightarrow_1 u_1$, $u \rightarrow_1 u_2$ usw. Ist $\rightarrow$ die transitive Hülle von $\rightarrow_1$, so bleibt die Definition von $L(KA)$ wie im deterministischen Fall.

1.2.3 Varianten zu Kellerautomaten

Akzeption durch leeren Keller:

Zu einem Kellerautomaten M mit Anfangszustand s_0 und Anfangszeichen z_0 bildet man gelegentlich die durch M mittels leerem Keller akzeptierte Sprache $T(M)$ wie folgt:

$$T(M) = \{w \in V_E^* : z_0 s_0 w \rightarrow_M s \text{ für ein } s \in S\},$$

d.h. $T(M)$ besteht genau aus den Wörtern w, so daß eine Ableitung gefunden werden kann, so daß nach Abarbeitung des gesamten Wortes w ein leerer Keller bleibt.

$$
\begin{array}{lcl}
S & = & \{s_0 = \text{ lese}, s_1 = \text{ vergleiche}\} \\
s_0 & = & \text{Anfangszustand} \\
V_E & = & \{a, b\} \\
V_K & = & \{z_0, A, B\} \\
z_0 & = & \text{Anfangssymbol}
\end{array}
$$

$d_1(z_0, s_0) = \{(e, s_0)\}$	
$d_2(A, s_0, a) = \{(AA, s_0)\}$	$d_2(A, s_0, b) = \{(AB, s_0)\}$
$d_2(B, s_0, a) = \{(BA, s_0)\}$	$d_2(B, s_0, b) = \{(BB, s_0)\}$
$d_2(e, s_0, a) = \{(A, s_0)\}$	$d_2(e, s_0, b) = \{(B, s_0)\}$
$d_1(A, s_0) = \{(A, s_1)\}$	
$d_1(B, s_0) = \{(B, s_1)\}$	
$d_2(A, s_1, a) = \{(e, s_1)\}$	
$d_2(B, s_1, B) = \{(e, s_1)\}$	

Abbildung 1.1: Der Kellerautomat M_R

Für ein Beispiel zur Akzeption mittels leerem Keller betrachten wir den in Abbildung 1.1 beschriebenen Kellerautomaten.

Beispiel einer Berechnung mit M_R:

Keller	Zustand	Restwort
z_0	s_0	$abbbba$
ε	s_0	$abbbba$
A	s_0	$bbbba$
AB	s_0	$bbba$
ABB	s_0	bba
ABB	s_1	bba
AB	s_1	ba
A	s_1	a
ε	s_1	ε

Also $abbbba \in T(MR)$. Es gilt

$$T(M_R) = \{ww^R : w \in VE^*\}$$

(w^R ist das zu w inverse Wort).

Die beiden Akzeptionsarten sind gleichmächtig, d. h.

Theorem 3 *Zu jedem Kellerautomaten M gibt es einen Kellerautomaten N, so daß*
$$T(M) = L(N)$$

und

Zu jedem Kellerautomaten M gibt es einen Kellerautomaten N, so daß

$$L(M) = T(N)$$

Beweise:

Sei M gegeben mit Startzustand s_0, Startzeichen z_0 und den Übergangsfunktionen d_1, d_2. Die Menge der Endzustände ist für die Definition von $T(M)$ nicht relevant. Der Kellerautomat N entsteht durch Hinzunahme eines neuen Zustandes s_F, $S_N = S_M \cup \{s_F\}$, der der einzige Finalzustand von N sein soll und der Änderung der Übergangsfunktion d_1 zu $d_{1,N}$ durch

$$d_{1,N}(e, s) = d_1(\varepsilon, s) \cup \{(\varepsilon, s_F)\}$$

für jedes $s \in S_M$. Für alle Argumente (z, s) mit $z \in VK$ gilt $d_{1,N}(z, s) = d_1(z, s)$ und d_2 bleibt unverändert beim Übergang von M zu N.

Liegt $w \in T(M)$, dann gilt $z_0 s_0 w \rightarrow_M s$ für ein $s \in S_M$. Dann gilt nach Definition von N auch $z_0 s_0 w \rightarrow_N s$ und nach Wahl von $d_{1,N}$ auch $z_0 s_0 w \rightarrow_N s_F$, also $w \in L(N)$.

Gilt umgekehrt $w \in L(N)$, so muß $z_0 s_0 w \rightarrow_N w_K s_F$ gelten, da s_F der einzige Endzustand von N ist. Es gibt nur eine Übergangsregel, die s_F als Nachfolgezustand produziert, nämlich

$$(\varepsilon, s_F) \in d_{1,N}(\varepsilon, s).$$

In jeder N-Herleitung kann dieser Übergang nur als letzter Schritt vorkommen, denn s_F kommt nicht in den Argumenten, für die d_1 oder d_2 definiert sind, vor. Also
$$z_0 s_0 w \rightarrow_N s \rightarrow_1 s_F$$

Da die neue Übergangsfunktion nicht in der Ableitung $z_0 s_0 w \rightarrow_N s$ auftreten kann, gilt auch $z_0 s_0 w \rightarrow_M s$ und damit $w \in T(M)$. Insgesamt ist damit $T(M) = L(N)$ gezeigt.

Sei jetzt die Sprache $L(M)$ gegeben. Der Kellerautomat N entstehe aus M durch Hinzunahme eines neuen Zustandes q_N und durch die Änderung von d_1 zu $d_{1,N}$:

$$d_{1,N}(z,q) = d_1(z,q) \cup \{(\varepsilon, q_N)\}$$

für jedes s aus der Menge der Finalzustände S_1 von M und jedes $z \in VK$ und

$$d_{1,N}(z,q) = \{(\varepsilon, q_N)\}$$

für jedes $z \in VK$.

1.2.4 Übungsaufgaben

Übungsaufgabe 1.2.1 *Führen Sie den Beweis Theorem 3 von zuende.*

Übungsaufgabe 1.2.2
Zu jedem Kellerautomaten M werde ein modifizierter Kellerautomat N_1 definiert, der dieselbe Menge von Zuständen, dasselbe Alphabet, denselben Anfangszustand und dasselbe Anfangszeichen wie M hat. Auch d_2 ist für N_1 und M diesselbe Übergangsfunktion. Lediglich $d_{1,N}$ unterscheidet sich von d_1:

$$\begin{array}{lcl} d_1N(z,q) & = & d_1(z,q) \text{ falls } q \text{ kein Endzustand von } M \text{ ist} \\ d_1N(z,q) & = & d_1(z,q) \cup \{(\varepsilon, q)\} \text{ falls } q \text{ ein Endzustand von } M \text{ ist} \end{array}$$

Geben Sie einen Kellerautomaten M an und ein Wort $w \in w_E^*$, so daß $w \in T(N)$ und $w \notin L(M)$.

Übungsaufgabe 1.2.3

1. *Geben Sie einen deterministischen Kellerautomaten M an mit $L(M) = \{w \in \{a,b\}^* \mid w \text{ enthält gleich viele Vorkommen von } a \text{ und } b\}$ Dabei soll die Übergangsfunktion d_1 von M nirgends definiert sein.*

2. *Geben Sie einen deterministischen Kellerautomaten N an mit $L(N) = \{w \in \{a,b\}^* \mid w \text{ enthält doppelt soviele Vorkommen von } a \text{ wie von } b\}$. Kann auch hier N gefunden werden, so daß die Übergangsfunktion d_1 nirgends definiert ist?*

Übungsaufgabe 1.2.4
Sei KA ein Kellerautomat deterministisch oder nicht. Falls eine natürliche Zahl n existiert, so daß bei der Abarbeitung eines beliebigen Wortes $w \in V^*$ der Keller nie mehr als n Buchstaben enthält, dann ist die von KA akzeptierte Sprache regulär.

Übungsaufgabe 1.2.5
Geben Sie einen deterministischen Kellerautomaten KA an, der eine nicht reguläre Sprache akzeptiert, so daß für alle Funktionswerte (w, q) der Übergangsfunktionen d_1, d_2 gilt $w \neq \varepsilon$.

1.3 Linear beschränkte Automaten

Die Namensgebung soll wieder anzeigen, daß die nichtdeterministischen Automaten die wichtigeren sind.

Die einzige Änderung liegt wieder in der Liberalisierung der Übergangsfunktion:

$$d : S \times (VT \setminus \{e_1, e_2\}) \to Pot(S \times VT \times \{R, N, L\})$$

Die Definition der unmittelbaren Nachfolgerrelation zwischen Augenblicksbeschreibungen muß wie schon in Unterabschnitt 1.2.2 der geänderten Definition von d angepaßt werden.

1.3.1 Deterministische linear beschränkte Automaten

Definition 11 (Deterministische linear beschränkte Automaten)
Ein linear beschränkter Automat (linear bounded automaton) wird durch die folgenden Bestimmungsstücke gegeben:

- *eine endliche Menge S von Zuständen*
- *ein Alphabet V*
- *eine Teilmenge $V_E \subseteq V$, das Eingabealphabet*

- *ein Anfangszustand* $s_0 \in S$
- *eine Menge* $S_1 \subseteq S$
- *zwei Endelemente* $e_1 e_2 \in VT$
- *eine Übergangsfunktion* $d : S \times (V \setminus \{e_1 e_2\}) \to S \times V \times \{R, N, L\}$

Wir stellen uns vor, daß ein linear beschränkter Automat mit einem Schreib- und Lesekopf ausgestattet ist, der es ihm erlaubt, jeweils einen Buchstaben auf dem Eingabeband zu lesen und möglicherweise auch zu überschreiben. Dieser Schreib- und Lesekopf kann innerhalb des Eingabewortes nach rechts oder links verschoben werden, jedoch nicht über die Grenzen des Eingabewortes hinaus. Wird im Zustand s gerade der Buchstabe a gelesen und ist $d(s, a) = (s_1, b, R)$ so wird der Buchstabe a durch b überschrieben, der Schreib- und Lesekopf um eine Position nach rechts verschoben und der Automat geht in den Zustand s_1 über. Für $d(s, a) = (s_1, b, N)$ bzw. $d(s, a) = (s_1, b, L)$ wird der Lesekopf nicht bewegt bzw. einen Buchstaben nach links bewegt. Eine Augenblicksbeschreibung für einen linear beschränkten Automaten LBA hat die Form

$$e_1 w_1 s w_2 e_2 \text{ oder } s e_1 w e_2$$

mit $w_1, w_2 \in (VT\ e_1, e_2)^*$ und $s \in S$.

Der Übergang von einer Augenblicksbeschreibung zur nächsten, der wieder mit $\to_1$ beschrieben wird, ist wie folgt definiert:

$$e_1 w_1 c s a w_2 e_2 \to_1 e_1 w_1 c b s_1 w_2 e_2$$

falls $d(s, a) = (s_1, b, R)$

$$e_1 w c s a w_2 e_2 \to_1 e w_1 s_1 c b w_2 e_2$$

falls $d(s, a) = (s_1, b, L)$

$$e_1 w c s a w_2 e_2 \to_1 e w_1 c s_1 b w_2 e_2$$

falls $d(s, a) = (s_1, b, N)$

Ist $\to$ bzw. genauer $\to_{LBA}$ wieder die transitive Hülle von $\to_1$, so können wir definieren:

$$L(LBA) = \{w \in VE^* \mid e_1 s_0 w e_2 \to e_1 w' s_1 e_2 \text{ mit } s_1 \in S_1\}$$

1.4 Turingmaschinen

Eine Turingmaschine ist durch dieselben Bestimmungsstücke gegeben wie ein linear beschränkter Automat mit dem einzigen Unterschied, daß VT ein Symbol 0 enthält, das als Leerzeichen gedeutet wird. Der wesentliche Unterschied besteht darin, daß die Turingmaschine auch über die Grenzen des Eingabewortes nach links oder rechts hinweg arbeiten kann. Das endliche Eingabewort steht auf einem unendlichen Band, das nach beiden Seiten mit dem Symbol 0 aufgefüllt ist. Als Augenblicksbeschreibungen dienen Wörter der Form

$$e_1 w_1 s w_2 e_2$$

mit$s \in S$, $w_1, w_2 \in (VT \setminus \{e_1, e_2\})^*$

Die unmittelbare Nachfolgerrelation $\rightarrow_1$ für die Augenblicksbeschreibungen einer Turingmaschine TA wird gegeben

falls $d(s, a) = (s_1, b, R)$ durch

$$e_1 w_1 csa w_2 e_2 \rightarrow_1 e_1 w_1 cb s_1 w_2 e_2 \text{ wenn } w_2 \neq e$$

bzw.

$$e_1 w_1 csa e_2 \rightarrow_1 e_1 w_1 cb s_1 0 e_2 \text{ sonst}$$

falls $d(s, a) = (s_1, b, L)$ durch

$$e_1 w_1 csa w_2 e_2 \rightarrow_1 e_1 w_1 s_1 cb w_2 e_2$$

bzw.

$$e_1 sa w_2 e_2 \rightarrow_1 e_1 s_1 0 b w_2 e_2$$

falls $d(s, a) = (s_1, b, N)$ durch

$$e_1 w_1 csa w_2 e_2 \rightarrow_1 e_1 w_1 c s_1 b w_2 e_2$$

1.4.1 Nichtdeterministische Turingmaschinen

Die einzige Änderung betrifft, wie schon in Unterabschnitt 1.3, die Übergangsfunktion d.

Im Fall der endlichen Automaten und der Turingmaschinen kann man zeigen, daß die deterministischen und die nichtdeterministischen Versionen dieselbe Erzeugungsmächtigkeit besitzen, d. h. zu jeder nichtdeterministischen Maschine N der jeweiligen Klasse gibt es eine deterministische Maschine D aus der jeweiligen Klasse mit $L(N) = L(D)$. Die umgekehrte Inklusion ist sowieso klar. Im Falle der Kellerautomaten gibt es Sprachen, die von (nichtdeterministischen) Kellerautomaten akzeptiert werden, aber von keinem deterministischen. Die Frage, ob deterministische und nichtdeterministische linear beschränkte Automaten dieselbe Erzeugungsmächtigkeit besitzen, gehört zu den großen ungelösten Problemen der Automatentheorie.

1.5 Abschlußeigenschaften

Definition 12 *Für zwei Sprachen $L_1, L_2 \subseteq V^*$ über demselben Alphabet ist das* Produkt $L_1 \cdot L_2$ *definiert durch:*

$$L_1 \cdot L_2 = \{w_1 w_2 \quad | \quad w_1 \in L_1, w_2 \in L_2\}$$

Wir lassen meist den Punkt $\cdot$ weg und schreiben $L_1 L_2$. Das Produkt zweier Sprachen wird häufig auch als Verkettung *(concatenation) bezeichnet.*

Definition 13 *Ist $L \subseteq V^*$ eine Sprache, dann ist*

$$L^* = \{w_1 \ldots w_n \quad | \quad n \in I\!N, w_i \in L \text{ für alle } i\}$$

Definition 14 *Seien V_1,V_2 zwei Alphabete.*
Eine Abbildung $\mu : V_1 \rightarrow Pot(V_2^*)$, die jedem Buchstaben $a \in V_1$ eine Teilmenge von Wörtern in V_2 zuordnet, nennen wir eine Substitution.
Eine Substitution heißt regulär, kontextfrei *usw, wenn für jedes $a \in V_1$ die Sprache $\mu(a)$ regulär, kontextfrei, usw ist.*
Durch die Definitionen

$$\mu(\varepsilon) = \varepsilon$$

$$\mu(a_1 \ldots a_k) = \mu(a_1) \ldots \mu(a_k)$$

wird die Abbildung $\mu : V_1 \rightarrow Pot(V_2^*)$ zu einer Abbildung $\mu : V_1^* \rightarrow Pot(V_2^*)$ fortgesetzt.

Definition 15 *Eine Abbildung $\mu : V_1 \rightarrow V_2^*$, die jedem Buchstaben $a \in V_1$ ein Wort $\mu(a)$ in V_2^* zuordnet heißt ein* Homomorphismus.

Wir können Homomorphismen als Spezialfall von Substitutionen betrachten, in dem für jedes $a \in V_1$ die Menge $\mu(a)$ genau ein Element besitzt.

Satz 4 *Alle Chomsky Klassen sind abgeschlossen unter $\cup, \cdot, ^*$*

Beweis: siehe [Moll *et al.*, 1988] Theorem 3, p.29. Die Beweise sind konstruktiv, d.h. zu gegebenen Grammatiken G_1, G_2 mit $L(G_i) = L_i$ werden Grammatiken für die Sprachen $L_1 \cup L_2$, $L_1 L_2$ und L_1^* explicit angegeben.

Satz 5 *Die Klasse der regulären Sprachen ist abgeschlossen unter Durchschnitten und Komplementen.*

Beweis: siehe [Moll *et al.*, 1988] Theorem 4, p. 32 und Proposition 10, p.37. Ist L durch einen deterministischen endlichen Automaten $A = (S, V, s_0, \delta, F)$ gegeben, dann wird das Komplement von A durch den Automaten $A^c = (S, V, s_0, \delta, S \setminus F)$ gegeben. Der Abschluß unter Durchschnitten folgt jetzt wegen $L_1 \cap L_2 = c(c(L_1) \cup c(L_2))$.

Lemma 6 *Die Klasse der kontextfreien Sprachen ist nicht abgeschlossen unter Durchschnitten.*

Beweis: Das Standardgegenbeispiel: L_1, L_2 sind kontextfrei

$$\begin{aligned} L_1 &= \{a^n b^n c^m \mid n, m > 0\} \\ L_2 &= \{a^n b^m c^m \mid n, m > 0\} \end{aligned}$$

aber der Durchschnitt

$$L_1 \cap L_2 = \{a^n b^n c^n \mid n > 0\}$$

ist nicht kontextfrei.

Satz 7 *Die Klasse der kontextfreien (regulären) Sprachen ist abgeschlossen unter kontextfreien (regulären) Substitutionen.*

Beweis: siehe [Moll *et al.*, 1988], Theorem 8 und 9 auf Seite 33.

Satz 8 *Die Klasse der kontextsensitiven Sprachen ist abgeschlossen unter Komplementen und damit auch unter Durchschnitten.*

Beweis: Die Frage, ob die Klasse der kontextsensitiven Sprachen unter Komplementen abgeschlossen ist, war lange ein offenes Problem, das 1964 zum ersten Mal explizit gestellt wurde und erst 1988 unabhängig von Immerman [Immermann, 1988] und Szelepcséngi [Szelepcséngi, 1988] gelöst wurde. Die Abgeschlossenheit unter Durchschnitten folgt dann wegen $A \cap B = c(c(A) \cup c(B))$.

∎

Beweise für die genannten Abschlusseigenschaften findet man auch in dem Buch [Carroll & Long, 1989] in Kapitel 5 für die Klasse der regulären und in Kapitel 9 für kontextfreie Sprachen.

Lemma 9 *Die Klasse der sternfreien Sprachen ist nicht abgeschlossen unter Homomorphismen*

Beweis: Die Sprache $L = (a)^* \subseteq V^*$ für $V = \{a\}$ ist sternfrei, siehe Beispiel 9. Unter dem Homomorphismus $h(a) = aa$ wird sie in die nicht sternfreie Sprache $h(L) = (aa)^*$ abgebildet, siehe Beispiel 12.

∎

Satz 10 *Die Klasse der ω-regulären Sprachen ist abgeschlossen unter $\cup$, $\cap$ und Komplementbildung $V^\omega \setminus L$.*

Beweis: Siehe Proposition III.3.2 in [Straubing, 1994], p.30.

∎

Satz 11 *Ist $L_1 \subseteq V^*$ eine kontexfreie Sprache und $L_2 \subseteq V^*$ eine reguläre Sprache, dann ist $L_1 \cap L_2$ kontextfrei.*

Beweis: Siehe [Carroll & Long, 1989], Theorem10.6. p. 352.

∎

Satz 12 *Ist $h : \Sigma \to \Gamma^*$ ein Homomorphismus und $L \subseteq \Gamma^*$ regulär, dann ist auch*

$$h^{-1}(L) = \{w \in V^* \mid h(w) \in L\}$$

eine reguläre Sprache.
Man sagt: die Klasse der regulären Sprachen ist unter inversen Homomorphismen abgeschlossen.

Beweis: siehe [Carroll & Long, 1989], p. 166.

Operation	ω-reg	reg-*	reg	ktf	kts	ra	rek
Produkt	-	ja	ja	ja	ja	ja	ja
*	-	nein	ja	ja	ja	ja	ja
Vereinigung	ja	ja	ja	ja	ja	ja	ja
Durchschnitt	ja	ja	ja	nein	ja	ja	ja
Komplement	ja	ja	ja	nein	ja	nein	ja
Substitution		nein	ja	ja			
Homomorphismus		nein	ja	ja			

ω-reg = ω-regulär (siehe Def.53)
reg-* = sternfrei (siehe Def.46)
reg = regulär
ktf = kontextfrei
kts = kontextsensitive
ra = rekursiv aufzählbar
rek = rekursiv

Abbildung 1.2: Abschlußeigenschaften

Definition 16 (Quotientensprache)
Zu zwei Sprachen $L_1, L_2 \subseteq V^*$ ist die Quotientensprache L_1/L_2 definiert durch

$$L_1/L_2 = \{w \in V^* \mid \text{ es gibt } u \in L_2 \text{ mit } wu \in L_1\}$$

Satz 13 *Sind $L_1, L_2 \subseteq V^*$ reguläre Sprachen, dann ist auch L_1/L_2 regulär.*

Beweis: siehe [Carroll & Long, 1989], p. 169.

1.5.1 Übungsaufgaben

Übungsaufgabe 1.5.1
Sei $V = \{a, b, c\}$ und $L = \{w \in V^* \mid \#_a(w) = \#_b(w) = \#_c(w)\}$. Hierbei ist $\#_a(w)$ die Anzahl der Vorkommen von a in w.
Zeigen Sie, daß L keine kontextfreie Sprache ist.
Hinweis: Versuchen Sie Satz 11 zu benutzen.

Problem	reg-*	reg	ktf	kts	ra	rek
$w \in L(G)$	e	e	e	e	u	e
$L(G) = \emptyset$	e	e	e	u	u	u
$L(G_1) = L(G_2)$	e	e	u	u	u	u
$L(G_1) \subseteq L(G_2)$	e	e	u	u	u	u

reg-*	=	sternfrei	e	=	entscheidbar
reg	=	regulär	u	=	unentscheidbar
ktf	=	kontextfrei			
kts	=	kontextsensitive			
ra	=	rekursiv aufzählbar			
rek	=	rekursiv			

Abbildung 1.3: Entscheidbarkeitseigenschaften

1.6 Entscheidbarkeit

Satz 14 *Das Problem zu einer vorgegebenen Grammatik G und einem Wort w über dem terminalen Alphabet von G zu entscheiden, ob $w \in L(G)$ gilt ist*

1. *entscheidbar für die Klasse der regulären, kontextfreien, kontextsensitiven Grammatiken,*

2. *unentscheidbare für allgemeine Grammatiken.*

Beweis: Siehe [Moll *et al.*, 1988] Theorem 7 auf Seite 103 oder [Salomaa, 1978] , pp 265.

Definition 17 (Postsches Korrespondenzproblem)
Ein **Postsches System** *über dem endlichen Alphabet Σ ist eine Menge von geordneten Paaren*

$$\{(x_i, y_i) \mid 1 \leq i \leq n, x_i, y_i \in (\Sigma^* \setminus \{\varepsilon\})\}.$$

Eine Lösung eines Postschen Korrespondenzproblems besteht darin, eine Zahl $k > 0$ und Indizes $i_1, \ldots, i_k$ zu finden, so daß

$$x_{i_1} \ldots x_{i_k} = y_{i_1} \ldots y_{i_k}$$

Beispiel 2 *Das Postsche System über dem Alphabet* $\Sigma = \{a, b\}$

$$P_1 = \{(aa, aab), (bb, ba), (abb, b)\}$$

besitzt die Lösung $k = 4, i_1 = 1, i_2 = 2, i_3 = 1, i_4 = 3$, *da*

$$aa\ bb\ aa\ abb = aab\ ba\ aab\ b$$

Die Grenzen zwischen den Teilwörtern wurden durch einen Leerraum sichtbar gemacht, um dem Leser die Zusammensetzung der rechten und linken Seite durch die Paare aus P_1 *leicht erkennbar zu machen.*

Das Postsche System über $\Sigma = \{a\}$*:*

$$P_2 = \{(aaa, aa), (aaaa, aaa)\}$$

besitzt keine Lösung.

Satz 15 *Das Postsche Korrespondenzproblem ist unentscheidbar.*

Beweis:

Siehe z.B. [Salomaa, 1978] p. 269 - 275 oder [Moll *et al.*, 1988] p. 103 - 106.

Satz 16 *Das Problem, von einer beliebigen kontextfreien Grammatik zu entscheiden, ob sie eindeutig ist, ist unentscheidbar.*

Beweis:

Angenommen es gäbe einen Algorithmus $\mathcal{A}$, der für beliebige kontextfreie Grammatiken entscheidet, ob sie eindeutig sind oder nicht. Wir werden einen Algorithmus $\mathcal{B}$ angeben, der das Postsche Korrespondenzproblem löst, was nach Satz 15 zu einem Widerspruch führt. Zu einer Eingabe

$$P = \{(x_i, y_i) \mid 1 \leq i \leq n\}$$

für $\mathcal{B}$ mit $x_i, y_i \in (\Sigma^* \setminus \{\varepsilon\})$ konstruieren wir die kontextfreie Grammatik G_P mit

Terminalzeichen	$\Sigma \cup \{a_1, \ldots, a_n\}$
Nichtterminalzeichen	$\{S, S_1, S_2\}$
Produktionen	$S ::= S_1 \mid S_2$
	$S_1 ::= x_1 S_1 a_1 \mid \ldots \mid x_n S_1 a_n$
	$S_1 ::= x_1 a_1 \mid \ldots \mid x_n a_n$
	$S_2 ::= y_1 S_2 a_1 \mid \ldots \mid y_n S_2 a_n$
	$S_2 ::= y_1 a_1 \mid \ldots \mid y_n a_n$

Man sieht leicht, daß gilt

$$\begin{array}{rcl} L(G_P) & = & \{x_{i_1} \ldots x_{i_k} a_{i_k} \ldots a_{i_1} \mid k \geq 1, 1 \leq i_j \leq n\} \\ & & \cup \\ & & \{y_{i_1} \ldots y_{i_k} a_{i_k} \ldots a_{i_1} \mid k \geq 1, 1 \leq i_j \leq n\} \end{array}$$

Die Grammatik G ist genau dann mehrdeutig, wenn es ein $k \geq 1$ und $i_j \in \{1, \ldots, n\}$ gibt mit

$$x_{i_1} \ldots x_{i_k} a_{i_k} \ldots a_{i_1} = y_{i_1} \ldots y_{i_k} a_{i_k} \ldots a_{i_1}$$

Diese Gleichung ist gleichbedeutend mit

$$x_{i_1} \ldots x_{i_k} = y_{i_1} \ldots y_{i_k}$$

Zusammenfassend gilt: G ist mehrdeutig gdw das Postsche System lösbar ist.

∎

Kapitel 2

Reguläre Sprachen

2.1 Minimale endliche Automaten

2.1.1 Rechtskongruenzen

Definition 18 (Äquivalenzrelation) *Eine binäre Relation R keißt eine Äquivalenzrelation, wenn sie die folgenden Eigenschaften besitzt:*

$$\begin{array}{rlcll} \forall x & & & R(x,x) & \textit{Reflexivität} \\ \forall x \forall y & R(x,y) & \Leftrightarrow & R(y,x) & \textit{Symmetrie} \\ \forall x \forall y \forall z & R(x,y) \wedge R(y,z) & \Rightarrow & R(x,z) & \textit{Transitivität} \end{array}$$

Für jedes Element a des Definitionsbereichs einer Äquivalenzrelation R bezeichnet $[a]_R$ die Äquivalenzklasse von a:

$$[a]_R = \{b \mid R(a.b)\}$$

Die Anzahl der Äquivalenzklasse heißt der Index *von R, $index(R)$.*

Definition 19 (Rechtskongruenz) *Eine Äquivalenzrelation R auf V^*, der Menge aller Wörter über einem Alphabet V, heißt eine* Rechtskongruenz, *wenn für alle w_1,$w_2 \in V^*$ gilt*

$$\textit{wenn } w_1 R w_2, \textit{ dann auch } w_1 u R w_2 u \textit{ für alle } u \in V^*$$

Beispiel 3 *für $a \in V$ sei $\#_a(w)$ die Anzahl der Vorkommen des Buchstabens a in dem Wort w und $\mid w \mid$ die Länge des Wortes w.*

$$\begin{array}{lll} w_1 R_1 w_2 & gdw & \#_a(w_1) = \#_a(w_2) \\ w_1 R_2 w_2 & gdw & \mid w_1 \mid = \mid w_2 \mid \\ w_1 R_3 w_2 & gdw & w_1 \text{ und } w_2 \text{ enden mit demselben} \\ & & \text{Buchstaben.} \end{array}$$

$$\begin{array}{lll} index(R_1) & = & \infty \\ index(R_2) & = & \infty \\ index(R_3) & = & card(V) \end{array}$$

Definition 20 (R_L) *Sei $L \subseteq V^*$ eine Sprache.*

$$w_1 R_L w_2$$
$$gdw$$
$$\text{für alle } u \in V^* \text{ gilt } w_1 u \in L \Leftrightarrow w_2 u \in L$$

Beispiel 4

$$\begin{array}{lll} L_1 & = & \{w \in \{a,b\}^* \mid \#_a(w) = 3\} \\ L_2 & = & \{w \in \{a,b\}^* \mid w \text{ enthält nicht das Teilwort } ab\} \\ L_3 & = & \{w \in \{a,b\}^* \mid \#_a(w) = \#_b(w)\} \end{array}$$

$index(R_{L_1}) = 4$
$index(R_{L_2}) = 3$
$index(R_{L_3}) = \infty$

Definition 21 (R^A) *Sei $A = (S, V_T, s_0, \delta, F)$ ein deterministischer endlicher Automat, $w_i \in V_T^*$*

$$w_1 R^A w_2 \text{ gdw } \delta(s_0, w_1) = \delta(s_0, w_2)$$

Lemma 17 *Die Relationen R_L aus Definition 20 und R^A aus Definition 21 sind Rechtskongruenzen.*

Beweis:
Einfaches Nachrechnen.

■

2.1.2 Der Satz von Myhill-Nerode

Satz 18 (Myhill-Nerode) *Für eine Sprache $L \subseteq V^*$ sind die folgenden Aussagen äquivalent:*

1. *es gibt einen endlichen Automaten A mit $L(A) = L$,*
2. *es gibt eine Rechtskongruenz R auf V^* mit endlichem Index, so daß L die Vereinigung einiger Äquivalenzklassen von R ist,*
3. *index(R_L)$< \infty$.*

Beweis von $1 \Rightarrow 2$:
Sei $A = (S, V_T, s_0, \delta, F)$ ein deterministischer endlicher Automat, so daß $L(A) = L$ gilt.
Wir benutzen die Rechtskongruenz

$$w_1 R_1^A w_2 \text{ gdw } \delta(s_0, w_1) = \delta(s_0, w_2)$$

aus dem früheren Beispiel.
Die Äquivalenzklassen sind:

$$K_s = \{w \in V^* \mid \delta(s_0, w) = s\}$$

für $s \in S$ und

$$L = \bigcup_{s \in F} K_S$$

Beweis von $2 \Rightarrow 1$:
Sei R die gegebene Rechtskongruenz auf V^*, für $w \in V^*$ bezeichne $[w]_R$ die R-Äquivalenzklasse von w ,K_i für $0 \leq i \leq n$ seien alle Äquivalenzklassen von R, wobei $[\varepsilon]_R = K_0$ und $L = K_{j_1} \cup \ldots \cup K_{j_r}$
Der endliche Automat $A_R = (S, V_T, s_0, \delta, F)$ werde definiert durch

$$\begin{array}{lcl} S & = & \{K_i \mid 0 \leq i \leq n\} \\ V_T & = & V \\ s_0 & = & K_0 \\ \delta(K, a) & = & [wa]_R \text{ für } w \in K \\ F & = & \{K_{j_1}, \ldots, K_{j_r}\} \end{array}$$

Man zeigt für alle w: $\delta(s_0, w) = [w]_R$ und damit

$$\begin{array}{rcl} L(A) & = & \{w \mid \delta(s_0, w) \in F\} \\ & = & K_{j_1} \cup \ldots \cup K_{j_r} \\ & = & L \end{array}$$

Beweis von $1 \Rightarrow 3$:
Es genügt zu zeigen, daß für $L = L(A)$ die Relation R^A feiner ist als R_L, d.h. $R_L \subseteq R^A$. Denn dann gilt index$(R_L) \leq$ index$(R^A) \leq card(S)$.

$$\begin{array}{rcl} w_1 R^A w_2 & \Leftrightarrow & \text{für alle } u \in V^* \text{ gilt } \delta(s_0, w_1 u) = \delta(s_0, w_2 u) \\ & \Rightarrow & \text{für alle } u \in V^* \text{ gilt } \delta(s_0, w_1 u) \in F \Leftrightarrow \delta(s_0, w_2 u) \in F \\ & \Leftrightarrow & \text{für alle } u \in V^* \text{ gilt } w_1 u \in L \Leftrightarrow w_2 u \in L \\ & \Leftrightarrow & w_1 R_L w_2 \end{array}$$

■

Lemma 19 *Sei L eine reguläre Sprache, R_L die zu L gehörige Rechtskongruenz aus Definition 20 und schließ lich $A_{R_L} = (S, V, s_0, \delta, F)$ der im Beweis von Satz 18 konstruierte endliche Automat zur Relation R_L, dann gilt:*

$$L(A_{R_L}) = L$$

Beweis:

$$\begin{array}{rcll} w \in A_{R_L} & \Leftrightarrow & \delta(s_0, w) \in F & \text{nach Def. von } L(A_{R_L}) \\ & \Leftrightarrow & [w]_{R_L} \in F & \text{wegen } \delta(s_0, w) = [w]_{R_L} \\ & \Leftrightarrow & w \in L & \text{nach Def. von } F \end{array}$$

■

2.1.3 Automaten-Homomorphismen

Definition 22 (Homomorphismen, Isomorphismen)
Seien $A_1 = (S_1, V_T, s_0^1, \delta_1, F_1)$, $A_2 = (S_2, V_T, s_0^2, \delta_2, F_2)$ zwei endliche deterministische Automaten über demselben Alphabet.

Eine Abbildung $\mu : S_1 \to S_2$ heißt ein Homomorphismus *von A_1 nach A_2 wenn gilt:*

1. $\mu(s_0^1) = s_0^2$

2. für alle $s \in F_1$ *gilt auch* $\mu(s) \in F_2$

3. für alle $s \in S_1$ *und alle* $a \in V_T$ *gilt*

$$\mu(\delta_1(s,a)) = \delta_2(\mu(s), a)$$

Ist die Abbildung μ *surjektiv, so heißt* A_2 *ein* homomorphes Bild *von* A_1*. Gilt sogar*

2. für alle $s \in S$ *gilt* $s \in F_1 \Leftrightarrow \mu(s) \in F_2$

dann nennen wir μ *einen* starken Homomorphismus.
μ *heißt ein* Isomorphismus*, wenn* μ *eine Homomorphismus ist und zusätzllich bijektiv ist.*

Lemma 20
Seien $A_1 = (S_1, V_T, s_0^1, \delta_1, F_1)$ *und* $A_2 = (S_2, V_T, s_0^2, \delta_2, F_2)$ *Mealy Automaten und*

$$\mu : A_1 \to A_2$$

ein Homomorphismus.

Dann gilt $L(A_1) \subseteq L(A_2)$*.*

Ist μ *ein starker Homomorphismus, dann gilt sogar* $L(A_1) = L(A_2)$*.*

Beweis:

$$\begin{array}{rcl} w \in L(A_1) & \Leftrightarrow & \delta_1(s_0^1, w) \in F_1 \\ & \Rightarrow & \mu(\delta_1(s_0^1, w)) \in F_2 \\ & \Leftrightarrow & \delta_2(\mu(s_0^1), w) \in F_2 \\ & \Leftrightarrow & \delta_2(s_0^2, w) \in F_2 \\ & \Leftrightarrow & w \in L(A_2) \end{array}$$

∎

Definition 23 *Ein Zustand s eines endlichen Automaten* $A = (S, V, s_0, \delta, F)$ *heißt* erreichbar*, wenn es ein* $w \in V^*$ *gibt mit* $\delta(s_0, w) = s$*. Alle anderen Zustände heißen unerreichbar.*

Lemma 21 *Der endliche Automat* $A_1 = (S_1, V, s_0, \delta_1, F_1)$ *entstehe aus* $A = (S, V, s_0, \delta, F)$ *durch Entfernung der unerreichbaren Zustände und entsprechender Restriktion von* δ *zu* δ_1*. Dann gilt* $L(A_1) = L(A)$*.*

Beweis:
trivial.

∎

Lemma 22 *Sei* $A_1 = (S_1, V, s_0^1, \delta_1, F_1)$ *ein deterministischer endlicher Automat ohne unereichbare Zustände mit* $L(A_1) = L$ *und sei* $A_{R_L} = (S, V, s_0, \delta, F)$ *der in Lemma 19 betrachtete endliche Automat.*
Dann ist A_{R_L} *ein homomorphes Bild von* A_1*.*

Beweis:

Die Abbildung $\mu : S_1 \to S$ werde für $s \in S_1$ definiert, indem zuerst ein Wort $w \in V^*$ gewählt wird mit $\delta_1(s_0^1, w) = s$. Ein solches Wort existiert, da A_1 keine unereichbaren Zustände enthält. Wir setzen:

$$\mu(s) = [w]_{R_L}$$

Ist w' ein weiteres Wort mit $\delta_1(s_0^1, w') = s$, dann gilt

$$\begin{array}{llrcl} & \text{für alle } u \in V^* & \delta_1(s_0^1, wu) & = & \delta_1(s_0^1, w'u) \\ \Rightarrow & \text{für alle } u \in V^* & wu \in L & \leftrightarrow & w'u \in L \\ \Rightarrow & & & & wR_Lw' \\ \Rightarrow & & [w]_{R_L} & = & [w']_{R_L} \end{array}$$

Zum Nachweis der Homomorphieeigenschaft von μ haben wir für jedes $s \in S_1$ und jedes $a \in V$ zu zeigen

$$\mu(\delta_1(s, a)) = \delta(\mu(s), a)$$

Sei $w \in V^*$ mit $\delta_1(s_0^1, w) = s$, dann gilt $\delta_1(s, a) = \delta_1(s_0^1, wa)$ und damit
$\mu(\delta_1(s, a)) = [wa]_{R_L}$ nach Definition von μ
Andererseits gilt auch $\mu(s) = [w]_{R_L}$ und nach Definition von δ in A_{R_L} folgt
$\delta([w]_{R_L}, a) = [wa]_{R_L}$

Außerdem gilt $\mu(s_0^1) = [\varepsilon]_{R_L} = s_0$ und für ein $s \in F_1$ und ein $w \in V^*$ mit $\delta_1(s_0^1, w) = s$ gilt $w \in L$ und daher auch $\mu(s) = [w]_{R_L} \in F$.

∎

Definition 24 *A heißt ein* minimaler *Automat für l, falls $L(A) = L$ gilt und jeder andere Automat A_1 mit $L(A_1) = L$ nicht weniger Zustände als A besitzt.*

Satz 23

1. *A_{R_L} ist ein minimaler Automat für L.*

2. *Je zwei minimale Automaten für L sind isomorph.*

Beweis: zu 1
Nach Lemma 19 gilt $L(A_{R_L}) = L$. Sei jetzt A ein weitere deterministischer Automat mit $L(A) = L$ und A_1 der Automat, der aus A durch Weglassen der unereichbaren Zustände entsteht, dann ist nach Lemma 22 A_{R_L} ein homomorphes Bild von A_1 und besitzt damit nicht mehr Zustände als A.
Beweis: zu 2
Ist A ein weiterer minimaler Automat für L, dann gibt es nach Lemma 22 einen Homomorphismus $\mu : A \to A_{R_L}$. Da A_{R_L} und A als minimale Automaten dieselbe Anzahl von Zuständen besitzen, ist μ bijektiv und damit ein Isomorphismus.

Definition 25 *Eine Kongruenzrelation $\approx$ auf einem deterministischen endlichen Automaten $A = (S, V, s_0, \delta, F)$ ist eine Äquivalenzrelation auf der Menge S der Zustände des Automaten mit der zusätzlichen Eigenschaft:*

$$\textit{aus } s \approx r \textit{ folgt für alle } a \in V \textit{ auch } \delta(s, a) \approx \delta(r, a)$$

Beispiel 5 $s \approx_A r \Leftrightarrow$ *für alle* $w \in V^*$ *gilt* $\delta(s, w) \in F \leftrightarrow \delta(r, w) \in F$

Definition 26 *Sei $\approx$ eine Kongruenzrelation auf dem deterministischen endlichen Automaten $A = (S, V, s_0, \delta, F)$, der Quotientenautomat $A_\approx = (S_\approx, V, s_0^\approx, \delta_\approx, F_\approx 0$ ist wir folgt definiert:*

$$
\begin{array}{lcl}
S_\approx & = & \{[s]_\approx \mid s \in S\} \\
V & = & V \\
s_0^\approx & = & [s_0]_\approx \\
\delta_\approx([s]_\approx, a) & = & [\delta(s,a)]_\approx \text{ für } a \in V \\
F_\approx & = & \{[s]_\approx \mid s \in F\}
\end{array}
$$

wobei $[s]_\approx = \{r \in S \mid s \approx r\}$ *die Kongruenzklasse von* s *ist.*

Man muß sich natürlich davon überzeugen, daß die Definition von $\delta_\approx$ unabhängig von der Wahl des Repräsentanten ist, d.h. für $s, r \in S$ mit $[s]_\approx = [r]_\approx$ muß für jedes $a \in V$ auch $[\delta(s,a)]_\approx = [\delta(r,a)]_\approx$ gelten. Aber das ist gerade die definierende Eigenschaft einer Kongruenzrelation auf Automaten.

Lemma 24

1. *Ist* $\approx$ *eine Kongruenzrelation auf dem deterministischen endlichen Automaten* A*, dann ist* $A_\approx$ *ein starkes homomorphes Bild von* A*. Insbesondere ist* $L(A) = L(A_\approx)$*.*
2. *Ist* $\mu : A \to A_1$ *ein Homomorphismus, dann wird durch* $s \approx_\mu r \Leftrightarrow \mu(s) = \mu(r)$ *eine Kongruenzrelation auf* A *definiert.*
3. *In der Situation unter 2. gilt*

$$A_1 \textit{ is isomorph zu } A_{\approx_\mu}$$

Beweis zu 1:
Setze $\mu(s) = [s]_\approx$. Dann gilt insbesondere $\mu(s_0) = s_0^\approx$ und für alle $s \in S$ auch $\mu(s) \in F_\approx \Leftrightarrow s \in F$. Die Homomorphieeigenschaft von μ folgt unmittelbar aus der Definition von $\delta_\approx$.
Beweis zu 2:
Gilt $s \approx_\mu r$, dann ist nach Definition $\mu(s) = \mu(r)$. Mit Hilfe der Homomorphieeigenschaft von μ folgt hieraus:

$$\mu(\delta(s,a)) = \delta_1(\mu(s), a) = \delta_1(\mu(r), a) = \mu(\delta(r,a))$$

Also gilt auch $\delta(s,a) \approx_\mu \delta(r,a)$
Beweis zu 3:
Übungsaufgabe.

■

Definition 27 *Wir nennen einen deterministischen endlichen Automaten A* reduziert*, wenn die in Beispiel 5 definierte Relation $\approx_A$ die Identitätsrelation ist, d.h. für alle $s, r \in S$, so daß für alle $w \in V^*$ gilt $\delta(s, w) \in F \Leftrightarrow \delta(r, w) \in F$, muß $s = r$ sein.*

Satz 25 *Ein deterministischer endlicher Automat A ist minimal für die Sprache $L = L(A)$ genau dann, wenn A reduziert ist und keine unerreichbaren Zustände aufweist.*

Beweis:

Ist A minimal für $L(A)$, dann kann A keine unerreichbaren Zustände enthalten, denn der um die unerreichbaren Zustände verkleinerte Automat hätte weniger Zustände und würde denoch dieselbe Sprache akzeptieren. Die Kongruenzrelation $\approx_A$ muß, wie übrigens jede andere Kongruenzrelation auf A, die Identitätsrelation sein, da sonst der Quotientenautomat $A_{\approx_A}$, immer noch L akzeptiert und echt weniger Zustände als A hätte.
Sei jetzt $A = (S, V, s_0, \delta, F)$ ein reduzierter Automat ohne unerreichbare Zustände. Nach Lemma 22 gibt es einen starken Homomorphismus $\mu : A \rightarrow A_{R_L}$. Es gilt die folgende Argumentationskette:

$$\begin{array}{lcll}
\mu(s_1) = \mu(s_2) & \Leftrightarrow & [w_1]_{R_L} = [w_2]_{R_L} & \text{für } w_i \in V^* \text{ mit} \\
 & & & \delta(s_0, w_i) = s_i \\
 & \Leftrightarrow & w_1 R_L w_2 & \\
 & \Leftrightarrow & w_1 u \in L \leftrightarrow w_2 u \in L & \text{für alle } u \in V^* \\
 & \Leftrightarrow & \delta(s_0, w_1 u) \in F \leftrightarrow \delta(s_0, w_2 u) \in F & \text{für alle } u \in V^* \\
 & \Leftrightarrow & \delta(s_1, u) \in F \leftrightarrow \delta(s_2, u) \in F & \text{für alle } u \in V^* \\
 & \Leftrightarrow & s_1 \approx_A s_2 & \\
 & \Leftrightarrow & s_1 = s_2 &
\end{array}$$

Wegen der Reduzierheit von A gilt somit

$$\mu(s_1) = \mu(s_2) \Leftrightarrow s_1 = s_2$$

Also ist μ ein Isomorphismus, A isomorph zu A_{R_L} und damit minimal.

■

Eingabe	ein Automat $A = (S, V, s_0, \delta, F)$
Initialisierung	$\Pi_{neu} = \{(S \setminus F), F\}$
repeat until	$\Pi = \Pi_{neu}$
	$\Pi := \Pi_{neu}$
	berechne Π_{neu} aus Π
end repeat	

Dabei ist Π_{neu} eine Verfeinerung der Partition Π mit der Eigenschaft:
für alle $a \in V$ und alle s, t, die in derselben Π_{neu}-Klasse liegen, liegen $\delta(s, a), \delta(t, a)$ in derselben Π-Klasse.

Abbildung 2.1: Algorithmus zur Berechnung des minimalen Automaten

2.1.4 Minimierungsalgorithmen

Die Partition Π_{neu} kann z.B. auf die folgende Weise aus der Partition $\Pi = \{K_1, \ldots, K_r\}$ berechnet werden:
für $1 \leq i, j \leq r$ und $a \in V$ setze

$$K_{i,j,a} = \{s \in K_i \mid \delta(s, a) \in K_j\}$$

$$K_{i,\sigma} = \bigcap_{a \in V} K_{i,\sigma(a),a}$$

und

$$\Pi_{neu} = \{K_{i,\sigma} \mid \sigma : 1 \leq i \leq r, V \to \{1, \ldots, r\}\}$$

Leere Mengen in Π_{neu} kann man natürlich sofort weglassen. Weitere Optimierungen sind möglich, z.B. braucht man für $1 \leq i \leq r$ mit einelementigem K_i nichts zu rechnen.

2.1.5 Übungsaufgaben

Übungsaufgabe 2.1.1 *Beweisen Sie Teil 3 von 24.*

Übungsaufgabe 2.1.2 *Berechnen Sie index(L) für*

b							
c							
d							
e							
f							
g							
h							
	a	b	c	d	e	f	g

Abbildung 2.2: Berechnung des minimalen Automaten, 0.Schritt

b							
c	×	×					
d			×				
e			×				
f			×				
g			×				
h			×				
	a	b	c	d	e	f	g

Abbildung 2.3: Berechnung des minimalen Automaten, 1.Schritt

b	×						
c	×	×					
d	×	×	×				
e		×	×	×			
f	×	×	×		×		
g	×	×	×	×	×	×	
h	×		×	×	×	×	×
	a	b	c	d	e	f	g

Abbildung 2.4: Berechnung des minimalen Automaten, Endergebnis

1. $L = \{ww^r \mid w \in V^*\}$
2. $L = \{a^n b^n \mid n \geq 0\}$
3. $L = \{a^n b^m c^k \mid n, m, k \geq 0\}$

Übungsaufgabe 2.1.3
Sei $\mu : A_1 \to A_2$ ein Homomorphismus. Falls A_2 keine unerreichbaren Zustände enthält, dann ist μ surjektiv.

Übungsaufgabe 2.1.4
Sei $A = (S, V, s_0, \delta, F)$ ein deterministischer endlicher Automat und $\Pi = \{K_1, \ldots, k_r\}$ eine Partition der Menge S aller Zustände. Die binäre Relation R auf S werde definiert durch

$$R(s,p) \Leftrightarrow \exists i (s \in K_i \text{ und } p \in K_i)$$

1. *Zeigen Sie, daß R eine Äquivalenzrelation ist.*

2. *Gilt für die Partition Π zusätzlich*
   für alle $a \in V$,für alle $1 \leq i \leq r$ und je zwei Zustände $s, t \in K_i$ gibt es ein $1 \leq j \leq r$, so daß $\delta(s,a), \delta(t,a) \in K_j$,
   dann ist R eine Kongruenzrelation auf A.

Ist umgekehrt eine Äquivalenzrelation R auf der Menge S gegeben, so ist die Menge der Äquivalenzklassen eine Partition von S.

3. *ist R eine Kongruenzrelation auf A, dann besitzt die durch die Äquivalenzklassen gegebene Partition die unter 2. genannte Eigenschaft.*

Übungsaufgabe 2.1.5

Definition 28 *Eine Relation R_1 heißt* feiner *als eine Relation R_2, wenn für alle a, b gilt:*

$$\text{wenn } aR_2b \text{ dann } aR_1b$$

Die Kongruenzrelation $\approx_A$ *aus Beispiel 5 auf einem endlichen Automaten* $A = (S, V, s_0, \delta, F)$ *ist die feinste Äquivalenzrelation, die* F *und* $(S \setminus F)$ *trennt. Genauer heißt das, daß* $\approx_A$ *eine Äquivalenzrelation auf* A *ist, so daß für alle* $q \in F$ *und alle* $r \in (S \setminus F)$ $q \not\approx_A r$ *gilt, was wir schon wissen, und daß jede Äquivalenzrelation* R *auf* A*, für die ebenfalls* $\neg(qRr)$ *gilt für alle* $q \in F$ *und alle* $r \in (S \setminus F)$ *gröber ist als* $\approx_A$*, d.h. für alle* $q_1, q_2 \in S$ *gilt die Implikation*

$$\textit{wenn } q_1 R q_2 \textit{ dann } q_1 \approx_A q_2$$

2.2 Endliche Automaten mit Ausgabe

Definition 29 (Mealy-Automaten)
Ein Mealy-Automat $A = (S, V_e, V_a, s_0, \delta, \lambda)$ *ist gegeben durch*

1. *eine endliche Menge* S *von Zuständen*
2. *einem Anfangszustand* s_0*,*
3. *einem Eingabealphabet* V_e*,*
4. *einer Übergangsfunktion* $\delta : S \times V_e \to S$
5. *einer Ausgabefunktion* $\lambda : S \times V_e \to V_a$

Definition 30 (Moore-Automaten)
Ein Moore-Automat $A = (S, V_e, V_a, s_0, \delta, \lambda)$ *ist wie ein Mealy-Automat mit dem einzigen Unterschied, daß die Ausgabefunktion nur vom Zustand abhängt, d.h.*

5. *einer Ausgabefunktion* $\lambda : S \to V_a$

Wie schon die Übergangsfunktion zu einer Funktion $\delta : S \times V_e^* \to S$ fortgesetzt werden konnte wird auch die Ausgabefunktion fortgesetzt zu einer Funktion $\lambda : S \times V_e^* \to V_a^*$.

Definition 31 (Erweiterte Ausgabefunktion)

$$\begin{aligned} \lambda(s, \varepsilon) &= \varepsilon \\ \lambda(s, wa) &= \lambda(s, w)\lambda(\delta(s, w), a) \end{aligned}$$

Definition 32 (Übersetzungsfunktion)
Ist M *ein Mealy oder Moore Automat, so heißt die Funktion* $f_M : V_e^* \to V_a^*$*, definiert durch*

$$f_M(w) = \lambda(s_0, w)$$

die Übersetzungsfunktion *von* M*.*
Zwei Mealy oder Moore Automaten M_1*,*M_2 *heißen* äquivalent*, falls* $f_{M_1} = f_{M_2}$*.*

2.2.1 Charakterisierungs- und Erhaltungssätze

Definition 33 *Eine beliebige Abbildung $f : V_e^* \to V_a^*$ von Wörtern über einem Alphabet V_e in Wörter über einem Alphabet V_a heißt*

längentreu *falls für jedes Wort $w \in V_e^*$ w und $f(w)$ dieselbe Länge haben,*

anfangstreu *für je zwei Worte $w_1, w_2 \in V_e^*$ gilt: wenn w_1 ein Anfangsstück von w_2 ist, dann ist auch $f(w_1)$ ein Anfangsstück von $f(w_2)$,*

Ableitung *wegen der Anfangstreue gibt es zu jedem Wort $w \in V_e^*$ eine durch f eindeutig bestimmte Funktion $f_w : V_e^* \to V_a^*$, genannt die* Ableitung von f an der Stelle w, *mit*

$$f(wu) = f(w)f_w(u)$$

ableitungsendlich *wenn die Menge $\{f_w \mid w \in V_e^*\}$ endlich ist.*

Lemma 26 *Die Übersetzungsfunktion f_M eines endlichen Automaten mit Ausgabe $M = (S, V_e, V_A, s_0, \delta, \lambda)$ ist*

1. *längentreu*
2. *anfangstreu*
3. *ableitungsendlich*

Beweis:
muß noch ergänzt werden.

■

Satz 27 *Ist $f : V_e^* \to V_a^*$ eine längentreue, anfangstreue und ableitungsendliche Funktion, dann gibt es einen endlichen Automaten mit Ausgabe $M = (S, V_e, V_a, s_0, \delta, \lambda)$, so daß f die Übersetzungsfunktion f_M von M ist.*

Beweis:
Die folgende Gleichung für Wörter $u, v, w \in V_e^*$ wird im Laufe des Beweises mehrfach nützlich sein:

$$f_u(vw) = f_u(v) f_{uv}(w) \tag{2.1}$$

Zum Beweis dieser Gleichung schreiben wir $f(uvw)$ nach Definition der Ableitung f_u als:

$$f(uvw) = f(u) f_u(vw) \tag{2.2}$$

Derselbe Funktionswert läßt sich auch mit Hilfe der Ableitung f_{uv} schreiben als:

$$f(uvw) = f(uv) f_{uv}(w) \tag{2.3}$$

Indem wir jetzt noch $f(uv)$ in termini von f_u schreiben erhalten wir

$$f(uvw) = f(u) f_u(v)(f_{uv}(w) \tag{2.4}$$

In den Gleichungen 2.2 und 2.4 sind die linken Seiten gleich und die rechten Seiten haben das gleiche Anfangstück. Daher müssen auch die Endstücke der rechten Seiten gleich sein. Das ist aber gerade die Gleichung 2.1, die wir beweisen wollten.
Zur Konstruktion von $M = (S, V_e, V_a, s_0, \delta, \lambda)$ setzt man:

$$\begin{aligned}
S &= \{f_w \mid w \in V_e^*\} \\
s_0 &= f_\varepsilon \\
\delta(f_w, a) &= f_{wa} \\
\lambda(f_w, a) &= f_w(a)
\end{aligned}$$

Diese Definition macht nur Sinn, wenn sie unabhängig ist von der Bezeichung der Funktion f_w, d.h. wenn für Wörter $u, v \in V_e^*$ und $a \in V_e$ aus $f_u = f_w$ auch $f_{ua} = f_{wa}$ folgt. Um das einzusehen schreiben wir mit Hilfe der Gleichung 2.1:

$$\begin{aligned}
f_u(az) &= f_u(a) f_{ua}(z) & (2.5)\\
f_w(az) &= f_w(a) f_{wa}(z) & (2.6)
\end{aligned}$$

Unter der Voraussetzung, daß $f_u = f_w$ gilt, sind die linken Seiten der Gleichungen 2.5 und 2.6 gleich und die rechten Seiten beginnen mit demselben Anfangsstück $f_u(a) = f_w(a)$. Also müssen auch die Endstücke der rechten Seiten gleich sein, d.h. $f_{ua}(z) = f_{wa}(z)$ für jedes $z \in V_e^*$.

Satz 28 *Sei $L \subseteq V_a^*$ eine reguläre Sprache, $M = (S, V_e, V_a, s_0, \delta, \lambda)$ ein Mealy-Automat und f_M die Übersetzungsfunktion von M. Dann ist auch*

$$f_M^{-1}(L) = \{w \in V_e^* \mid f_M(w) \in L\}$$

eine reguläre Sprache.

Beweis:
Beweisidee: Ist der Index von R_L endlich, dann ist auch der Index der Relation $R_{f_M^{-1}(L)}$ endlich.

Satz 29 *Sei $L \subseteq V_e^*$ eine reguläre Sprache, $M = (S_M, V_e, V_a, s_0^M, \delta_M, \lambda_M)$ ein Mealy-Automat und f_M die Übersetzungsfunktion von M. Dann ist auch*

$$f_M(L) = \{f_M(w) \mid w \in L\} \subseteq V_a^*$$

eine reguläre Sprache.

Beweis:
Da L regulär ist gibt es einen endlichen deterministischen Automaten $A = (S_A, V_e, s_0^A, \delta_A, F_A)$ mit $L(A) = L$. Wir geben einen nichtdeterministischen endlichen Automaten $B = (S_B, V_a, s_0^B, \delta_B, F_B)$ an, so daß $L(B) = f_M(L)$. B wird gegeben durch:

$$\begin{array}{lcl}
S_B & = & \{(q,r) \mid q \in S_A, r \in S_M\} \\
s_0^B & = & (s_0^A, s_0^M) \\
F_B & = & \{(q,r) \mid q \in F_A, r \in S_M\} \\
\delta_B((q,r),b) & = & \{(q',r') \mid \text{ es gibt } a \in V_e \text{ mit} \\
 & & \lambda_M(r,a) = b \\
 & & \delta_M(r,a) = r' \\
 & & \delta_A(q,a) = q'\}
\end{array}$$

Durch Induktion über die Länge des Wortes $w \in V_a^*$ zeigt man

$$\begin{array}{lcl}
\delta_B((q,r),w) & = & \{(q',r') \mid \text{ es gibt } u \in V_e^* \text{ mit} \\
 & & \lambda_M(r,u) = w \\
 & & \delta_M(r,u) = r' \\
 & & \delta_A(q,u) = q'\}
\end{array}$$

$$
\begin{aligned}
w \in L(B) \quad &\Leftrightarrow \quad \delta_B((s_0^A, s_0^M), w) \cap F_B \neq \emptyset \\
&\Leftrightarrow \quad \text{es gibt } u \in V_e^* \text{ mit} \\
&\qquad\quad \lambda_M(s_0^M, u) = w \\
&\qquad\quad \delta_A(s_0^A, u) \in F_A \\
&\Leftrightarrow \quad \text{es gibt } u \in V_e^* \text{ mit} \\
&\qquad\quad f_M(u) = w \text{ und } u \in L(A) \\
&\Leftrightarrow \quad w \in f_M(L)
\end{aligned}
$$

■

2.2.2 Minimale Automaten mit Ausgabe

Definition 34 *Ein Mealy-Automat M heißt* minimal, *wenn jeder Mealy-Automat M_1 mit $f_M = f_{M_1}$ nicht weniger Zustände als M besitzt.*

Definition 35 *Seien zwei Mealy-Automaten $M_1 = (S_1, V_e, V_a, s_0^1, \delta_1, \lambda_1)$, $M_2 = (S_2, V_e, V_a, s_0^2, \delta_2, \lambda_2)$ gegeben. Ein* Homomorphismus $\mu : M_1 \rightarrow M_2$ *ist eine Abbildung $\mu : S_1 \rightarrow S_2$, so daß gilt:*

1. *$\mu(s_0^1) = s_0^2$,*
2. *$\mu(\delta_1(s, a)) = \delta_2(\mu(s), a)$ für alle $a \in V_e$ und alle $s \in S_1$,*
3. *$\lambda_1(s, a) = \lambda_2(\mu(s), a)$ für alle $a \in V_e$ und alle $s \in S_1$.*

Lemma 30 *Sei $\mu : M_1 \rightarrow M_2$ ein Homomorphismus. Dann gilt für alle $s \in S_1$ und alle $w \in V_e^*$*

$$\lambda_1(s, w) = \lambda_2(\mu(s), w)$$

Insbesondere gilt

$$f_{M_1} = f_{M_2}$$

Beweis: Einfaches Nachrechnen.

■

Definition 36 *Für jeden Mealy-Automaten $M = (S, V_e, V_a, s_0, \delta, \lambda)$ ist die Standard-Kongruenzrelation E_M auf S definiert durch*

$$rE_M q \quad \Leftrightarrow \quad \textit{für alle } w \in V_e^* \textit{ gilt } \lambda(r, w) = \lambda(q, w)$$

Definition 37 *Ein Mealy-Automat M heißt* reduziert*, wenn die Standard-Kongruenzrelation die Gleichheit ist:*

$$qE_M r \Leftrightarrow q = r$$

Definition 38 *Zu einem Mealy-Automaten $M = (S, V_e, V_a, s_0, \delta, \lambda)$ wird der Quotientenautomat $M/E = (S_E, V_e, V_a, s_0^E, \delta_E, \lambda_E)$ definiert durch*

$$\begin{array}{lcl} S_E & = & \{[q]_E \mid q \in S\} \\ s_0^E & = & [s_0]_E \\ \delta_E([q]_E, a) & = & [\delta(q, a)]_E \\ \lambda_E([q]_E, a) & = & \lambda(q, a) \end{array}$$

Hierbei bezeichnet $[q]_E$ die Kongruenzklasse von q bezüglich der Relation E_M. Zur Wohldefiniertheit siehe Aufgabe 2.2.3.

Lemma 31 *Der Quotientenautomat M/E_M eines Mealy-Automaten M ist reduziert und äquivalent zu M.*

Satz 32 *Ein Mealy-Automat M ist minimal, genau dann wenn M*

1. *reduziert und*
2. *ohne unerreichbare Zustände ist.*

Beweis:
Es ist leicht, analog zu den Beweisen für endliche Automaten, einzusehen, daß jeder minimale Mealy-Automat reduziert sein muß und keine unerreichbaren Zustände haben kann.
Sei jetzt

$$M_1 = (S_1, V_e, V_a, s_0^1, \delta_1, \lambda_1)$$

ein reduzierter Mealy-Automat ohne unerreichbare Zustände und

$$M = (S, V_e, V_a, s_0, \delta, \lambda)$$

ein weiterer Mealy-Automat mit $f_M = f_{M_1}$. Wir wollen zeigen, daß M nicht mehr Zustände haben kann als M_1. Dazu können ohne Einschränkung der Allgemeinheit annehmen, daß M keine unerreichbaren Zustände hat. Wir definieren eine Abbildung $\mu : S \to S_1$ durch die folgende, inzwischen bekannte, Konstruktion. Zu jedem Zustand $s \in S$ gibt es ein Wort $w \in V_e^*$ mit $\delta(s_0, w) = s$, da M keine unerreichbaren Zustände enthält. Wir setzen

$$\mu(s) = \delta_1(s_0^1, w)$$

Es bleibt zu zeigen, daß diese Definition unabhängig ist von der Wahl des Wortes w, d.h. aus der Voraussetzung $\delta(s_0, w_1) = \delta(s_0, w_2)$ müssen wir $\delta_1(s_0^1, w_1) = \delta_1(s_0^1, w_2)$ folgern.

$$\begin{array}{lcll} f_M(w_1 w) & = & f_{M_1}(w_1 w) & \text{Voraus.} \\ \lambda(s_0, w_1 w) & = & \lambda_1(s_0^1, w_1 w) & \text{Def. von } f_M, f_{M_1} \\ \lambda(s_0, w_1)\lambda(\delta(s_0, w_1), w) & = & \lambda_1(s_0^1, w_1)\lambda_1(\delta_1(s_0^1, w_1), w) & \text{Def. von } \lambda(s, u) \end{array}$$

Analog erhält man für w_2 anstelle von w_1

$$\lambda_1(s_0^1, w_2)\lambda_1(\delta_1(s_0^1, w_2), w) \quad = \quad \lambda_1(s_0^1, w_2)\lambda_1(\delta_1(s_0^1, w_2), w)$$

Da wegen

$$\lambda_1(s_0^1, w_i) = f_{M_1}(w_i) = f_M(w_i) = \lambda(s_0, w_i)$$

die Anfangsstücke auf beiden Seiten gleich sind folgt auch die Gleichheit der Endstücke, d.h. die beiden Gleichungen:

$$\begin{array}{lcl} \lambda(\delta(s_0, w_1), w) & = & \lambda_1(\delta_1(s_0^1, w_1), w) \\ \lambda(\delta(s_0, w_2), w) & = & \lambda_1(\delta_1(s_0^1, w_2), w) \end{array}$$

Nach Wahl von w_i gilt $\delta(s_0, w_1) = \delta(s_0, w_2)$ und damit sind die linken Seiten der beiden letzten Gleichungen und damit auch ihre rechten Seiten gleich, d.h.

$$\lambda_1(\delta_1(s_0^1, w_1), w) = \lambda_1(\delta_1(s_0^1, w_2), w)$$

Diese Gleichung gilt für alle $w \in V_e^*$. Daraus folgt per Definition von E_{M_1}

$$\delta_1(s_0^1, w_2) E_{M_1} \delta_1(s_0^1, w_1)$$

und wegen der Reduziertheit von M_1 auch

$$\delta_1(s_0^1, w_2) = \delta_1(s_0^1, w_1)$$

Da M_1 keine unerreichbaren Zustände besitzt ist die Abbildung μ surjektiv. Damit ordnet die Umkehrabbildung μ^{-1} jedem Zustand von M_1 mindestens einen Zustand von M zu, d.h. M kann nicht weniger Zustände besitzen als M_1.

∎

Satz 33 *Zwei reduzierte Mealy-Automaten ohne unerreichbare Zustände sind*

isomorph gdw sie äquivalent

sind.

Beweis:
Trivialerweise sind zwei isomorphe Mealy-Automaten auch äquivalent.
Seien jetzt $M_1 = (S_1, V_e, V_a, s_0^1, \delta_1, \lambda_1)$ und $M = (S, V_e, V_a, s_0, \delta, \lambda)$ reduzierte Mealy-Automat ohne unerreichbare Zustände. Wir wollen zeigen, daß M_1 und M isomorph sind. Wir betrachten wieder die im Beweis von Satz 32 konstruierte Abbildung $\mu : M \rightarrow M_1$ und zeigen, daß μ sogar ein Isomorphismus ist. Die Homomorphieeigenschaft von μ ist leicht nachzuweisen. Die Injektivität von μ wird nach demselben Muster bewiesen, wie die Wohldefiniert heit im Beweis von Satz 32.

∎

2.2.3 Übungsaufgaben

Übungsaufgabe 2.2.1 *Beweisen Sie Satz 28.*

Übungsaufgabe 2.2.2 *Konstruieren Sie zu einem gegebenen Mealy Automaten M einen äquivalenten Moore Automaten. (Die umgekehrte Aufgabe ist trivial).*

Übungsaufgabe 2.2.3 *Zeigen Sie, daß die Funktionen δ_E und λ_E in Definition 38 wohldefiniert, d.h. unabhängig von der Wahl eines Repräsentanten für die beteiligten Äquivalenzklassen sind.*

Übungsaufgabe 2.2.4 *Sei $f : V_e^* \to V_a^*$ eine längentreue, anfangstreue und ableitungsendliche Funktion und M_f der in Satz 27 konstruierte Mealy-Automat, so daß f die Übersetzungsfunktion M_f ist.*
Zeigen Sie, daß M_f reduziert ist und keine unerreichbaren Zustände besitzt.

2.3 Definierbarkeit

In diesem Abschnitt soll eine logische Charakterisierung regulärer Sprachen, die auf Büchi und McNaughton zurückgeht, vorgestellt werden. Wir folgen dabei [Straubing, 1994], [Thomas, 1990] und [Pin, 1994].

2.3.1 Monadische Logik 2. Stufe

Unter einer logischen Charakterisierung verstehen wir eine Definition, die auf Begriffe der Logik zurückgreift, wie den Begriff einer Formel, einer Struktur und der Gültigkeit einer Formel in einer Struktur. Wir beginnen mit der zentralen und anfänglich verblüffenden Idee Wörter als Strukturen darzustellen. Dabei entspricht jedem Wort w über einem Alphabet V eine Struktur $\mathcal{M}_w$ wie folgt.

Definition 39 *Die Struktur $\mathcal{M}_w$ besteht aus*

- *dem Universum $\{0, \ldots n-1\}$, wobei n die Länge des Wortes w ist,*
- *der Nachfolgerrelation $y = x + 1$ und der Gleichheitsrelation $y = x$ auf den natürlichen Zahlen, natürlich eingeschränkt auf des jeweilige Anfangsstück,*
- *für jedes $a \in V$ einer einstelligen Relation $Q_a(x)$, wobei $Q_a(i)$ für $0 \leq i < n$ genau dann wahr ist, wenn der i-te Buchstabe von w ein a ist.*

Im Gegensatz zur gewohnten Definition prädikatenlogischer Strukturen sind hier auch leere Strukturen zugelassen, nämlich $\mathcal{M}_\epsilon$

Beispiel 6 *Für das Wort $w = abbca \in \{a, b, c, d\}^*$ ist $\mathcal{M}_w$ die Struktur mit dem Universium $\{0, 1, 2, 3, 4\}$ mit der naheliegenden Interpretation der Gleichheits- und der Nachfolgerrelation und*

$$\begin{array}{rcl} Q_a^{\mathcal{M}_w} & = & \{0, 4\} \\ Q_b^{\mathcal{M}_w} & = & \{1, 2\} \\ Q_c^{\mathcal{M}_w} & = & \{3\} \\ Q_d^{\mathcal{M}_w} & = & \emptyset \end{array}$$

Als nächstes interessieren wir uns für die Gültigkeit logischer Formeln in Strukturen der Form $\mathcal{M}_w$.

Beispiel 7 *Für $w \in \{a,b,c\}^*$ gilt*

$$\mathcal{M}_w \models \exists x \exists y (y = x + 1 \wedge Q_a(x) \wedge Q_a(y))$$

gdw
w enthält zwei aufeinanderfolgende a's.

Beispiel 8 *Für $w \in \{a,b,c\}^*$ gilt*

$$\mathcal{M}_w \models \forall \forall y (y = x + 1 \wedge Q_a(x) \rightarrow \neg Q_a(y))$$

gdw
w enthält keine zwei aufeinanderfolgende a's.

Definition 40 *Die Sprache $\mathcal{L}_1^2$ der monadischen Logik zweiter Stufe zu einem gegebenen Vokabular V besteht aus folgenden Formeln:*

1. *für je zwei Objektvariable x,y liegen $y = x + 1$ und $y = x$ in $\mathcal{L}_1^2$,*

2. *für jeden Buchstaben $a \in V$ und jede Objektvariable x liegt $Q_a(x)$ in $\mathcal{L}_1^2$,*

3. *für jede Mengenvariable X und jede Objektvariable x liegt $X(x)$ in $\mathcal{L}_1^2$,*

4. *für ϕ_1 und ϕ_2 in $\mathcal{L}_1^2$ liegen auch $\phi_1 \wedge \phi_2$, $\phi_1 \vee \phi_2$ und $\neg \phi_2$ in $\mathcal{L}_1^2$,*

5. *liegt ϕ in $\mathcal{L}_1^2$, dann auch $\exists x \phi$ und $\forall x \phi$ für jede Objektvariable x,*

6. *liegt ϕ in $\mathcal{L}_1^2$, dann auch $\exists X \phi$ und $\forall X \phi$ für jede Mengenvariable X.*

Definition 41 *Ist $\phi(x_1, \ldots, x_n, X_1, \ldots, X_m)$ eine Formel in $\mathcal{L}_1^2$ über V mit den freien Objektvariablen $x_1, \ldots, x_n$ und den freien Mengenvariablen $X_1, \ldots, X_m$, so ist für ein Wort $w \in V^*$, Elemente $i_1, \ldots, i_n$ aus $\{0, \ldots \mid w \mid -1\}$ und Teilmengen $P_1, \ldots, P_m$ von $\{0, \ldots \mid w \mid -1\}$ die Relation*

$$\mathcal{M}_w \models \phi[i_1, \ldots, i_n, P_1, \ldots, P_m]$$

wie üblich definiert. Die Objektquantoren $\forall x$, $\exists x$ laufen dabei über alle Elemente des Universums, die Mengenquantoren $\forall X$, $\exists X$ über alle Teilmengen des Universums. Die Belegungen der freien Variablen werden hinter der Formel in eckigen Klammern an gegeben.

Nachdem klar ist, was gemeint sein soll, schreiben wir manchmal $w \models \phi[i, P]$ anstelle von $\mathcal{M}_w \models \phi[i, P]$

Definition 42 *Sei Φ eine Formel ohne freie Variablen aus $\mathcal{L}_1^2$ zum Vokabular V.*
Die durch Φ definierte Sprache $L(\Phi)$ ist gegeben durch:

$$L(\Phi) = \{w \in V^* \mid w \models \Phi\}$$

Satz 34 *Zu jeder regulären Sprache $L \subseteq V^*$ gibt es eine Formel Φ ohne freie Variablen aus $\mathcal{L}_1^2$ zum Vokabular V, so daß*

$$L(\Phi) = L$$

Beweis:
Wir gehen aus von einem deterministischen endlichen Automaten $A = (S, V, s_0, \delta, F)$ mit $L(A) = L$. Die Menge der Zustände sei dabei $S = \{s_0, \dots, s_n\}$
Wir formulieren zunächst die Definition für $w \in L(A)$ um und zeigen anschließend daß die dabei entstehenden Teilaussagen in $\mathcal{L}_1^2$ formalisiert werden können.
Sei $w = a_0 \dots a_{k-1} \in V^*$. Dann ist $w \in L(A)$ äquivalent zu der Konjunktion der folgenden Aussagen:

1. es gibt Teilmengen $X_0, \dots, X_n$ von $U = \{0, \dots k-1\}$
   (wir denken dabei an die Interpretation:
   $r \in X_i$ genau dann, wenn vor dem Einlesen des r-ten Buchstabens befindet sich der Automat im Zustand i.

2. $\bigcup_{0 \leq i \leq n} X_i = U$,

3. für $i \neq j$ sind X_i und X_j disjunkt,

4. $0 \in X_0$,

5. für alle $0 \leq r < k-1$ gilt, wenn $X_i(r)$ und $X_j(r+1)$, dann muß $\delta(s_i, a_r) = s_j$ sein,

6. gilt $X_i(k-1)$, dann gilt $\delta(s_i, a_{k-1}) \in F$.

Hier die Formalisierung in $\mathcal{L}_1^2$:

2. $\forall x \bigvee_{0 \leq i \leq n} X_i(x)$,

3. $\forall x \bigwedge_{0 \leq i < j \leq n} \neg(X_i(x) \wedge X_j(x))$,

4. $\forall x (\forall y \neg(x = y + 1) \rightarrow X_0(x))$,

5. $\forall x \forall y (y = x + 1 \rightarrow \bigwedge_{0 \leq i,j \leq n}(X_i(x) \wedge X_j(y) \rightarrow \bigvee_{a \in S_{i,j}} Q_a(x))$,
   wobei $S_{i,j} = \{a \in V \mid \delta(s_i, a) = s_j\}$,

6. $\forall x (\neg \exists y (y = x + 1) \rightarrow \bigvee_{(a,i) \in Final}(Q_a(x) \wedge X_i(x))$
   wobei $Final = \{(a, i) \mid \quad a \in V \text{ und } \delta(s_i, a) \in F\}$.

Bezeichnen wir mit $\Phi_2, \ldots, \Phi_6$ diese Formeln, so erhalten wir die gesuchte L definierende Formel als

$$\Phi = \exists X_0 \ldots \exists X_n (\Phi_2 \wedge \ldots \wedge \Phi_6)$$

Der aufmerksame Leser wird bemerkt haben, daß die obige Argumentation für das leere Wort ε nicht funktioniert. Man muß also für den Fall, daß $\varepsilon \in L$ gilt die folgende Modifikation anbringen. Man bestimmt zunächst, wie oben beschrieben, eine Formel $\Phi_1 \in \mathcal{L}_1^2$ mit $l(\Phi_1) = L \setminus \{\varepsilon\}$ und setzt $\Phi = \Phi_1 \vee \forall x (\neg x = x)$.

∎

Satz 35 *Für jede Formel $\Phi \in \mathcal{L}_1^2$ ohne freie Variablen ist $L(\Phi)$ eine reguläre Sprache.*

Der Beweis dieses Satzes verlangt einiges an Vorbereitung, so daß wir ihn noch etwas zurückstellen müssen. Der Beweis wird letzten Endes durch Induktion über der Formelaufbau von Φ geführt werden. Um in dieser Induktion den Fall der Quantifizierung behandeln zu können, müssen wir gezwungenermaßen auch Formeln mit freien Objekt- und Mengenvariablen betrachten. Die Gültigkeit solcher Formeln hängt nun nicht mehr nur von einer Struktur ab, die durch ein Wort $w \in V^*$ eindeutig bestimmt ist, sondern auch von der Belegung der freien Variablen. Diese Belegungen müssen auch durch Wörter kodiert werden. Als ersten Schritt dazu gehen wir vom gegebenen Alphabet V zu neuen Alphabeten $V_{p,q}$ über für natürliche Zahlen $0 \leq p, q$.

Definition 43 $V_{p,q} = V \times \{0,1\}^p \times \{0,1\}^q$

Für $p = q = 0$ haben wir $V_{p,q} = V$. Für $p = 3$, $q = 2$ und $a \in V$ sind

$$\begin{aligned} b_1 &= \langle a, 0, 0, 0, 0, 0 \rangle \\ b_2 &= \langle a, 0, 0, 1, 0, 0 \rangle \\ b_3 &= \langle a, 0, 1, 0, 0, 1 \rangle \end{aligned}$$

Buchstaben in $V_{3,2}$. Wir wollen das Wort $b_1 b_2 b_3 \in V_{3,2}^*$ in der folgenden Art anordnen:

	b_1	b_2	b_3
u_0	a	a	a
u_1	0	0	0
u_2	0	0	1
u_3	0	1	0
u_4	0	0	0
u_5	0	0	1

Im allgemeinen wird ein Wort $c_0 \dots c_{k-1} \in V_{p,q}^*$ angeordnet als Hierbei ist u_0 ein Wort in V^* und u_i für $1 \leq i \leq p + q$ ein 0-1-Wort. Zur Kodierung der Belegung freier Variablen in $\mathcal{L}_1^2$-Formeln benötigen wir die folgenden Teilmengen $K_{p,q} \subseteq V_{p,q}^*$:

Definition 44 *$K_{p,q}$ besteht genau aus denjenigen Wörtern $c \in V_{p,q}^*$, so daß jedes der 0-1-Wörter u_i für $1 \leq i \leq p$ genau eine 1 enthält.*

	c_0	$\dots$	c_{k-1}
u_0	a_0	$\dots$	a_{k-1}
u_1	$b_{1,0}$	$\dots$	$b_{1,k-1}$
$\dots$	$\dots$	$\dots$	$\dots$
u_p	$b_{p,0}$	$\dots$	$b_{p,k-1}$
u_{p+1}	$b_{p+1,0}$	$\dots$	$b_{p+1,k-1}$
$\dots$	$\dots$	$\dots$	$\dots$
u_{p+q}	$b_{p+q,0}$	$\dots$	$b_{p+q,k-1}$

Abbildung 2.5: Wörter in $V_{p,q}^*$

Wir wollen Formeln aus $\mathcal{L}_1^2$ deren freie Objektvariablen in $\{x_1, \dots, x_p\}$ und deren freie Mengenvariablen in $\{X_1, \dots, X_q\}$ vorkommen in einer Struktur $\mathcal{M}_w$ interpretieren für $w \in K_{p,q}$.

Definition 45 *Für $w \in K_{p,q}$ ist*

$$\mathcal{M}_w = (\mathcal{M}_{u_0}, s_1, \dots, s_p, S_1, \dots, S_q)$$

wobei $\mathcal{M}_{u_0}$ wie in Definition 39 gegeben ist und für $1 \leq i \leq p$
s_i der eindeutig bestimmte Index $0 \leq r \leq\mid w \mid -1$ ist mit $u_i(r) = 1$
und für $1 \leq j \leq q$
$S_j = \{r \mid 0 \leq r \leq\mid w \mid -1, u_{p+j}(r) = 1\}$

Satz 36 *Für jede Formel $\Phi(x_1, \dots, x_p, X_1, \dots, X_q)$ aus $\mathcal{L}_1^2$ ist*

$$L(\Phi(\bar{x}, \bar{X})) = \{w \in K_{p,q} \mid \mathcal{M}_w \models \Phi(x_1, \dots, x_p, X_1, \dots, X_q)\}$$

eine reguläre Sprache in $V_{p,q}^*$.

Satz 35 ist der Spezialfall dieses Satzes für $p = q = 0$.

Beweis:
Wir stellen als erstes fest, daß für alle $0 \leq p, q$ die Sprache $K_{p,q} \subseteq V_{p,q}^*$ regulär ist. Siehe Übungsaufgabe 2.3.4.

Wir benutzen im folgenden ohne weitere Erwähnung die Notation aus Abbildung 2.5.
Im Induktionsanfang ist für die atomaren Formeln Φ

$$x_i = x_j \quad x_i = x_j + 1 \quad Q_a(x_i) \quad X_i(x_j)$$

zu zeigen, daß $L(\Phi) \subseteq V_{p,q}^*$ regulär ist. Wir betrachten den ersten Fall näher und überlassen dem Leser die restlichen Nachweise. Seien also Indixes $1 \leq i, j \leq p$ gegeben. Nach Definition gilt:

$L(x_i = x_j) = \{w \in K_{p,q} \mid u_i(r) = u_j(r) = 1$ für ein $0 \leq r < k\}$

Sei

$C_{i,j} = \{\langle a, c_1, \ldots, c_{p+q}\rangle \in V_{p,q} \mid a \in V, c_k \in \{0,1\}, c_i \neq c_j\}$ und

$G_{i,j} = \{w \in V_{p,q}^* \mid w$ enthält keinen Buchstaben aus $C_{i,j}\}$

Offensichtlich ist $G_{i,j}$ eine reguläre Sprache und, da die Klasse der regulären Sprachen unter Durchschnitten abgeschlossen ist, ist auch $L(x_i = x_j) = K_{p,q} \cap G_{i,j}$ regulär.
Da die Klasse der regulären Sprachen abgeschlossen ist unter Vereinigungen, Durchschnitten und Komplementen sind die Induktionsschritte für die Disjunktion, Konjunktion und Negation simpel.
Sei die Behauptung des Satzes für $\Phi(x_1, \ldots, x_p, X_1, \ldots, X_q)$ schon bewiesen. Wir wollen sie für

$$\exists x_i \Phi(x_1, \ldots, x_p, X_1, \ldots, X_q) \text{ und } \exists X_j \Phi(x_1, \ldots, x_p, X_1, \ldots, X_q)$$

zeigen.
Die Substitution $\mu : V_{p,q} \to Pot(V_{p,q}^*)$ werde definiert durch

$$\begin{aligned}\mu(\langle a, c_1, \ldots, c_{i-1},\, c_i, c_{i+1}, \ldots c_{p+q}\rangle) \\ = \{\langle a, c_1, \ldots, c_{i-1}, x, c_{i+1}, \ldots c_{p+q}\rangle \mid \quad x \in \{0,1\}\}\end{aligned}$$

Man überzeugt sich leicht, dass

$$L(\exists x_i \Phi) = \mu(L(\Phi))$$

gilt. Da reguläre Sprachen unter Substitutionen abgeschlossen sind folgt aus der vorausgesetzten Regularität von $L(\Phi)$ die Regularität von $L(\exists x_i \Phi)$. Das Argument kann unverändert für $\exists X_j \Phi$ übernommen werden.
Bemerkung: Wir betrachten in diesem Beweis sowohl $L(\Phi)$ als auch $L(\exists x_i \Phi)$ und $L(\exists X_j \Phi)$ als Sprachen über demselben Alphabet $V_{p,q}$. Vergleiche dazu Übungsaufgabe 2.3.5.

2.3.2 Sternfreie Sprachen

Definition 46 *Die Klasse der* sternfeien Sprachen *über einem Alphabet V wird induktiv definiert durch:*

1. *die leere Menge $\emptyset$ ist eine sternfreie Sprache,*
2. *die Menge $\{\varepsilon\}$, die als einiziges Element das leere Wort enthält, ist eine sternfreie Sprache,*
3. *die Einermengen $\{a\}$ sind für jeden Buchstaben $a \in V$ sternfreie Sprachen,*
4. *sind L_1, L_2 sternfreie Sprachen, dann auch $L_1 \cup L_2$, $L_1 L_2$ und $V^* \setminus L_1$.*

Wir wiederholen zum Vergleich die Definition einer regulären Menge von Wörtern über dem Alphabet V:

Definition 47

1. *die leere Menge $\emptyset$ ist eine reguläre Menge,*
2. *die Menge $\{\varepsilon\}$, die als einiziges Element das leere Wort enthält, ist eine reguläre Menge,*
3. *die Einermengen $\{a\}$ sind für jeden Buchstaben $a \in V$ reguläre Mengen,*
4. *sind L_1, L_2 reguläre Mengen, dann auch $L_1 \cup L_2$, $L_1 L_2$ und L_1^*.*

Da reguläre Sprachen unter Komplementen abgeschlossen sind ist offensichtlich jede sternfreie Sprache auch regulär.

Beispiel 9 (aus [Pin, 1994])
Sei $V = \{a, b\}$. Die Sprachen

$$V^* \qquad\qquad a^* \qquad\qquad (ab)^*$$

sind sternfrei. Das zeigt man am besten, indem man nachweist, daß die

Komplemente dieser Mengen sternfreie Mengen sind:

$$\begin{array}{lcll} V^* \setminus V^* & = & \emptyset & \textit{1 der Definition} \\ V^* \setminus a^* & = & V^*bV^* & \textit{3,4 der Definition} \\ V^* \setminus (ab)^* & = & bV^* \cup V^*a \cup V^*aaV^* \cup V^*bbV^* & \end{array}$$

Satz 37 *Sei eine reguläre Sprache $L \subseteq V^*$ gegeben.*
L ist sternfrei gdw der minimale endliche Automat $A = (S, V, s_0, \delta, F)$ für L die folgende Eigenschaft besitzt.

> *Es gibt eine natürliche Zahl n, so daß für jedes Wort $w \in V$ für alle $s \in S$ gilt $\delta^n(s, w) = \delta^{n+1}(s, w)$*

Beweis: Siehe [Schützenberger, 1965].

Dieser Satz eignet sich besonders gut, um die engen Verbindungen theoretischer Aspekte formaler Sprachen mit Konzepten aus der Algebra aufzuzeigen. Die Aussage des Satzes 37 ist nämlich eine verkürzte Beschreibung eines algebraischen Sachverhaltes, den wir jetzt etwas ausführlicher erklären wollen. Zunächst einige Definitionen.

Definition 48 (Monoid)
Eine **Halbgruppe** *ist eine Menge G mit einer assoziativen, zweistelligen Operation $\cdot$, d.h.*

$$(G, \cdot) \models \forall x \forall y \forall z((x \cdot y) \cdot z = x \cdot (y \cdot z))$$

Ein **Monoid** $(G, \cdot, e)$ *ist eine Halbgruppe $(G, \cdot)$ mit einem neutralen Element e:*

$$(G, \cdot, e) \models \forall x(x \cdot e = e \cdot x = x)$$

Typische Beispiele für Monoide sind Mengen von einstelligen Funktionen einer Menge in sich selbst, wobei die Operation $\cdot$ die Hintereinanderausführung von Funktionen ist. Das funktioniert auch für Mengen partieller Funktionen. Als neutrales Element fungiert in diesen Fällen die Identitätsfunktion.

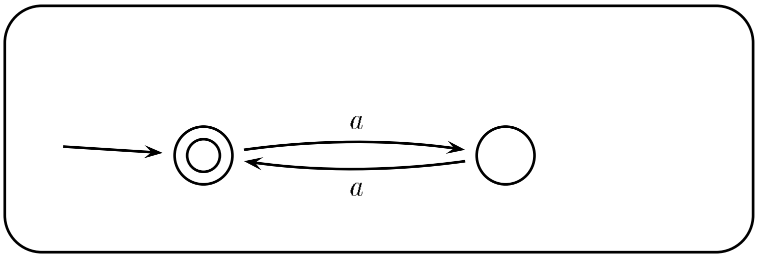
Definition 49 (Syntaktisches Monoid)
Einem deterministischen endlichen Automaten $A = (S, V, s_0, \delta, F)$ ordnet man ein Monoid $mon(A) = (G, \cdot, e)$ wie folgt zu. Das Universum G von $mon(A)$ besteht aus allen (partiellen) Funktionen der Form $f(s) = \delta(s, w)$ von S in S für ein $w \in V^*$. Also

$$G = \{\delta(x, w) \mid w \in V^*\}$$

Die Halbgruppenoperation $\cdot$ ist die Komposition von Funktionen und das neutrale Element die Identitätsfunktion, $e = \delta(x, \varepsilon)$.
Das **syntaktische Monoid**, *$mon(L)$, einer regulären Sprache L ist das Monoid des minimalen Automaten für L.*

Man beachte, daß das Universum von $mon(A)$ unter $\cdot$ abgeschlossen ist, wegen $\delta(x, w) \cdot \delta(x, u) = \delta(x, uw)$, siehe Aufgabe 2.3.6.

Beispiel 10 *Sei A der folgende deterministische endliche Automat:*



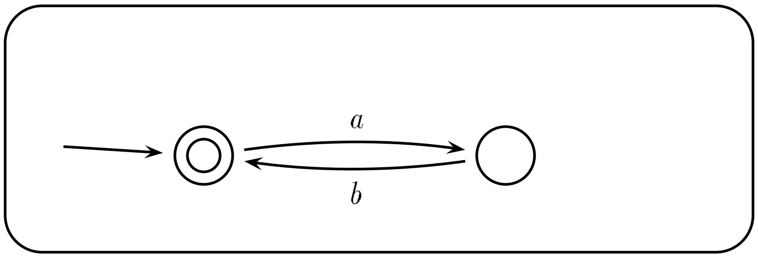
Man sieht leicht, daß für jedes $w \in \{a, b\}^*$ die Funktion $\delta(x, w)$ gleich einer der drei Funktionen

$$\begin{array}{lllll} e & = & \delta(x, \varepsilon) & = & \textit{Identitätsfunktion} \\ f_a & = & \delta(x, a) & = & \textit{Vertauschung von 1 und 2} \\ 0 & = & \delta(x, b) & = & \textit{die nirgends definierte partielle Funktion} \end{array}$$

ist. Es ergibt sich somit für $mon(A)$ die folgende Multiplikationstabelle

$\cdot$	e	f_a	0
e	e	f_a	0
f_a	f_a	e	0
0	0	0	0

Beispiel 11 *Sei B der folgende deterministische endliche Automat:*



Man sieht leicht, daß für jedes $w \in \{a,b\}^*$ $\delta(x,w)$ *gleich einer der folgenden sechs Funktionen ist*

$$
\begin{array}{lcl}
e &=& \delta(x,\varepsilon) &=& \textit{Identitätsfunktion} \\
f_a &=& \delta(x,a) &=& f_a(1)=2, f_a(2)\uparrow \\
f_b &=& \delta(x,b) &=& f_a(1)\uparrow, f_a(2)=1 \\
f_{ab} &=& \delta(x,ab) &=& f_{ab}(1)=1, f_{ab}(2)\uparrow \\
f_{ba} &=& \delta(x,ba) &=& f_{ba}(1)\uparrow, f_{ba}(2)=2 \\
0 &=& \delta(x,aa) &=& \textit{die nirgends definierte partielle Funktion}
\end{array}
$$

Dabei bedeutet $g(a)\uparrow$, *daß die Funktiojn g für das Argument a nicht definiert ist.*
Die Multiplikationstabelle für $mon(B)$ *berechnet sich jetzt als:*

$\cdot$	e	f_a	f_b	f_{ab}	f_{ba}	0
f_a	f_a	0	f_{ab}	0	f_a	0
f_b	f_b	f_{ba}	0	f_b	0	0
f_{ab}	f_{ab}	f_a	0	f_{ab}	0	0
f_{ba}	f_{ba}	0	f_b	0	f_{ba}	0
0	0	0	0	0	0	0

Definition 50 (Aperiodisches Monoid)
Ein Monoid $M = (G, \cdot, e)$ *heißt* **aperiodisch**, *wenn es ein* $k \in I\!N$ *gibt, sodaß für alle* $m \in G$ *gilt* $m^{k+1} = m^k$.

Satz 37: (Zweite Fassung)

Sei eine reguläre Sprache $L \subseteq V^*$ gegeben.
L ist sternfrei gdw das syntaktische Monoid von L aperiodisch ist.

Wir illustrieren die Aussage dieses Satzes an zwei Beispielen.

Beispiel 12

1. *Der Automat A aus Beispiel 10 akzeptiert die Sprache* $(aa)^*$ *und ist auch, wie man unschwer sieht, der minimale Automat für diese Sprache. Aus der oben gegeben Multiplikationstabelle für das syntaktische Monoid von* $(aa)^*$ *sieht man, daß dieses nicht aperiodisch ist und damit die Sprache selbst nicht sternfrei.*

2. *Der Automat B aus Beispiel 11 akzeptiert die Sprache* $(ab)^*$ *und ist wieder der minimale Automat für diese Sprache. Diesmal zeigt die Inspektion der Multiplikationstabelle, daß das syntaktische Monoid aperiodisch und damit* $(ab)^*$ *sternfrei ist.*

In Abschnitt 2.3.1 haben wir die Sprache $\mathcal{L}_1^2(V, =, +)$, die als numerische Basisrelationen die Gleichheit und die Nachfolgerrelation enthielt. Aus den Übungsaufgaben 2.3.1(1) und 2.3.3 folgt, daß diese Sprache dieselbe Ausdrucksstärke besitzt wie die Sprache $\mathcal{L}_1^2(V, <)$, die nur über $<$ als numerische Basisrelation verfügt. Betrachten wir nur Logiken erster Stufe, dann ist $\mathcal{L}^1(V, =, +)$ echt ausdrucksärmer als $\mathcal{L}^1(V, <)$. Für das folgende Resultat müssen wir die stärkere Sprache benutzen. Zu einem Wort $w \in V^*$ bezeichnen wir die Stuktur zur Sprache $\mathcal{L}^1(V, <)$ wieder mit $\mathcal{M}_w$ und nehmen die kleine Ungenauigkeit in Kauf, daß jetzt andere numerische Basisrelationen vorkommen.

Satz 38 *Die folgenden Aussagen sind für eine Sprache* $L \subseteq V^*$ *äquivalent:*

1. *es gibt eine Formel erster Stufe* $\phi \in \mathcal{L}^1(V, <)$ *ohne freie Variablen mit* $L(\phi) = L$

2. L *ist eine sternfreie Sprache.*

Beweis: siehe [McNaughton & Pappert, 1971].

2.3.3 Übungsaufgaben

Übungsaufgabe 2.3.1

1. *Geben Sie eine Formel* $\Phi_<(x,y)$ *aus* $\mathcal{L}_1^2$ *mit zwei freien Objektvariablen an, so daß für jedes Vokabular* V*, jedes Wort* $w \in V^*$ *und je zwei Zahlen* $i, j <\mid w \mid$ *gilt*

$$w \models \Phi_<[i,j] \text{ gdw } i < j$$

2. *Geben Sie für* $m > 0$ *eine Formel* $\Theta_m(x)$ *aus* $\mathcal{L}_1^2$ *mit einer freien Objektvariablen an, so daß für jedes Vokabular* V*, jedes Wort* $w \in V^*$ *und jede Zahl* $i <\mid w \mid$ *gilt*

$$w \models \Theta_m[i] \text{ gdw } i \equiv \quad 0 \mod m$$

Übungsaufgabe 2.3.2
Zeigen Sie, daß es keine Formeln der Prädikatenlogik erster Stufe gibt mit den in der vorangegangenen Übungsaufgabe, 2.3.1, verlangten Eigenschaften.

Übungsaufgabe 2.3.3
Mit $\mathcal{L}_1^2(<)$ *bezeichnen wir eine Sprache, die aufgebaut ist wie* $\mathcal{L}_1^2$ *mit dem einzigen Unterschied, daß anstelle der beiden Relationen* $x = y$ *und* $x = y + 1$ *die Relation* $<$ *auftritt. Zeigen Sie, daß in* $\mathcal{L}_1^2(<)$ *die weggelassenen Relationen* $x = y$ *und* $x = y + 1$ *in termini von* $<$ *definiert werden können, sogar durch Formeln erster Stufe.*

Übungsaufgabe 2.3.4
Zeigen Sie, daß die Sprache $K_{p,q} \subseteq V_{p,q}^*$ *regulär ist.*

Übungsaufgabe 2.3.5
Sei Φ *eine Formel aus* $\mathcal{L}_1^2$ *ohne freie Variable. Aus Satz 36 folgt, daß* $L(\Phi)$ *eine reguläre Menge in* $V_{p,q}^*$ *ist, für geeignet gewählte* p *und* q*.*

1. *Zeigen Sie, daß dann* $L(\Phi)$ *auch eine reguläre Teilmenge von* V^* *ist.*

2. *Wann sind* p *und* q *geeignet gewählt?*

Übungsaufgabe 2.3.6
Zeigen Sie, daß für die Übergangsfunktion δ eines endlichen Automaten über dem Vokabular V gilt:

$$\delta(x, w) \cdot \delta(x, u) = \delta(x, uw)$$

Dabei bezeichne $\cdot$ die Hintereinandnerausführung der beiden Funktionen $\delta(x, w)$ und $\delta(x, u)$ in dem Sinne $\delta(\delta(x, w), u)$.

Übungsaufgabe 2.3.7
Es gilt der Satz:
Ein endliches Monoid $M = (G, \cdot, e)$ ist aperiodisch
genau dann, wenn
$(G, \cdot)$ keine nichttriviale Untergruppe enthält.

Man beachte, daß dabei das neutrale Element e des Monoids keine Rolle spielt. Eine nichttriviale Gruppe ist eine Gruppe mit mehr als einem Element.

1. *Zeigen Sie zuerst die einfache Richtung: wenn für ein $k \in I\!N$ für alle $m \in G$ gilt $m^{k+1} = m^k$, dann besitzt $(G, \cdot)$ keine nichttriviale Untergruppe.*

2. *Sei $m \in G$. Wegen der Endlichkeit von G gibt es Zahlen $k, r > 0$ mit $m^k = m^{k+r}$. Zeigen Sie, daß die Teilmenge*

$$G_m = \{m^k, m^{k+1}, \ldots, m^{k+r-1}\} \subseteq G$$

unter der Operation $\cdot$ abgeschlossen ist.

3. *Zeigen sie weiter, daß ein $a \in G_m$ existiert, so daß für alle $g \in G_m$ gilt $a \cdot g = g \cdot a = g$,*

4. *Zeigen sie, daß für jedes $g \in G_m$ ein $h \in G_m$ existiert mit $g \cdot h = a$.*

5. *Aus den vorangegangenen Überlegungen folgt, daß für jedes $m \in G$ die Struktur G_m eine Gruppe ist. Unter der Voraussetzung, daß in G keine nichttrivialen Untergruppen auftreten folgt also $r = 1$, d.h. $m^k = m^{k+1}$. Vervollständigen Sie, die kleine Lücke, die noch zu einem kompletten Beweis fehlt.*

Lösungshinweise findet man in [Straubing, 1994] , Seite 59.

Übungsaufgabe 2.3.8

Zeigen Sie, daß es keine Formel in $\mathcal{L}_1^2$ *gibt, welche die Addition auf* $I\!N$ *definiert.*

2.4 Unendliche Wörter

Manche Automaten besitzen keine terminalen Zustände, man möchte im Gegenteil sogar, daß sie nicht terminieren. Ein Beispiel dafür sind Sender- und Empfängerautomaten für Übertragungsprotokolle. Das führt auf der theoretischen Seite zur Einführung unendlich langer Wörter.

Definition 51 (Unendliche Wörter)
Sei V ein (weiterhin endliches) Alphabet.
Mit V^ω bezeichnen wir die Menge der unendlich langen Wörter mit Buchstaben aus V.
Für $n \in N$ bezeichnet $w(n)$ den n-ten Buchstaben in w und $w \downarrow (n)$ das endliche Anfangstück $w(0) \ldots w(n)$ von w.
Wir nennen ein Wort $w \in V^\omega$ manchmal auch ein ω-Wort über V.

Wer Schwierigkeiten hat sich eine unendlich lange Zeichenkette vorzustellen, der interpretiere ein Wort $w \in V^\omega$ als eine Funktion $w : I\!N \to V$, von den natürlichen Zahlen in das Alphabet. Die Präzisierung beantwortet auch die Frage, des mit der Theorie der unendlichen Mengen vertrauten Lesers, welchen Grad der Unendlichkeit wir meinen: unendliche Wörter sind Folgen vom Ordnungstyp ω.
Man beachte außerdem, daß das leere Wort ε nicht in V^ω vorkommt.

Definition 52 (Operationen)

1. *Ist $K \subseteq V^*$ eine Menge (endlicher) Wörter, so bezeichnet K^ω die Menge der unendlichen Wörter der Form*

   $$w_1 \ldots w_i \ldots \text{ mit } w_i \in K \text{ für alle } i$$

2. *Die Verkettung zweier unendlich langer Wörter macht keinen Sinn, es ist jedoch möglich einem unendlich langen Wort ein endliches vorzuschalten. Ist $K \subseteq V^*$ und $J \subseteq V^\omega$, dann setzen wir:*

   $$KJ = \{w_1 w_2 \quad | \quad w_1 \in K, w_2 \in J\}$$

3. *Ist $K \subseteq V^*$ eine Menge endlicher Wörter, dann ist*

   $$\vec{K} = \{w \in V^\omega \mid w \downarrow (n) \in K \text{ für unendlich viele } n\}$$

   Manche Autoren benutzen $lim(K)$ anstelle von $\vec{K}$.

2.4.1 Büchi Automaten

Definition 53 (Büchi Automaten) *Sei $A = (S, V, s_0, \delta, F)$ ein nicht deterministischer endlicher Automat. Für ein ω-Wort $w \in V^\omega$ nennen wir eine Folge $s_0, \ldots, s_n, \ldots$ eine* **Berechnungsfolge** *für w, wenn für alle $0 \leq n$ gilt*

$$s_{n+1} \in \delta(s_n, w(n))$$

Die von A akzeptierte ω-Sprache wird definiert durch

$$L^\omega(A) = \{w \in V^\omega \mid \text{ es gibt eine Berechnungsfolge für } w \text{ mit unendlich vielen Finalzuständen}\}$$

Wir nennen eine Menge L von ω-Wörtern **ω-regulär**, *wenn es einen endlichen Automaten A gibt mit $L^\omega(A) = L$.*

Ein Büchi Automat A unterscheidet sich also in nichts von einem endlichen Automaten, nur die Definition, wann A ein ω-Wort akzeptiert ist neu.

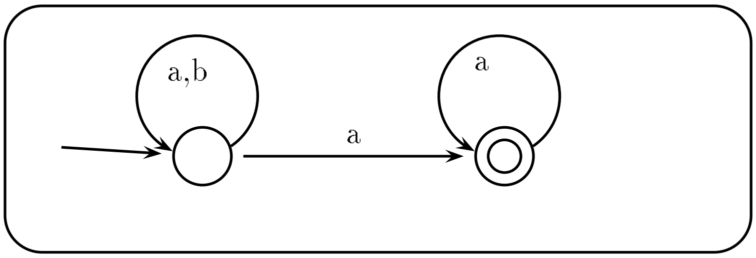


Abbildung 2.6: Büchi Automat für $\{a, b\}^* a^\omega$

Lemma 39 *Sei A ein deterministischer endlicher Automat, $K = L(A)$ und $L = L^\omega(A)$. Dann gilt*

$$L = \vec{K}$$

Beweis:
Zu einem Wort $w \in V^\omega$ gibt es genau eine Berechnungsfolge, die wir mit $\rho(w)$ bezeichnen wollen. Nach Definition gilt

$$w \in L^\omega(A) \Leftrightarrow R = \{n \in I\!N \mid \rho(w)(n) \in F\} \text{ ist unendlich}$$

Für alle $n \in R$ gilt $\delta(s_0, w\downarrow(n)) = \rho(w)(n) \in F$ und daher $w\downarrow(n) \in K$.

■

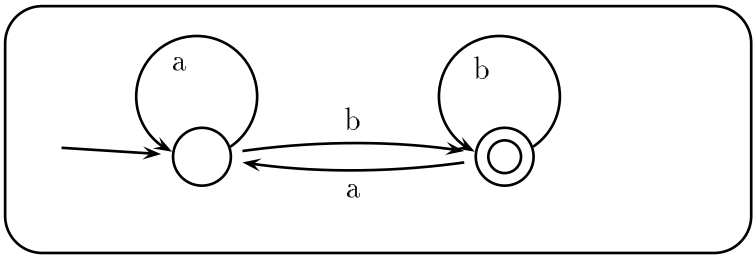


Abbildung 2.7: Büchi Automat für $\{a^*ba^*\}^\omega$

Korollar 40 *Eine Sprache $L \subseteq V^\omega$ wird von einem deterministischen Büchi Automaten akzeptiert, gdw es eine reguläre Sprache $K \subseteq K^*$ gibt mit $L = \vec{K}$.*

Beweis:
Wird L von einem deterministischen Büchi Automaten akzeptiert, so kann L nach Lemma 39 auch in der angegebenen Weise dargestellt werden. Nehmen wir jetzt umgekehrt an, das $L = \vec{K}$ für eine reguläre Sprache $K \subseteq V^*$ gilt. Dann gibt es einen deterministischen endlichen Automaten A mit $K = L(A)$. Man überzeugt sich leicht, daß in diesem Fall $\vec{K} = L^\omega(A)$ gilt.

∎

Korollar 41 *Es gibt Sprachen $L \subseteq V^\omega$, die von einem nicht deterministischen Büchi Automaten akzeptiert werden, aber von keinem deterministischen.*

Beweis:
Sei $V = \{a, b\}$ und

$$L = \{w \in V^\omega \mid w \downarrow (n) = a \text{ nur für endlich viele } n\}$$

Wir zeigen, daß L nicht in der Form $L = \vec{K}$ für eine reguläre Menge $K \subseteq V^*$ darstellbar ist. Wir nehmen an, das wäre doch der Fall und versuchen einen Widerpsruch herzuleiten.
Es gibt ein $k_1 > 0$, so daß $b^{k_1} \in K$, da $b^\omega \in L$. Als nächstes muß es dann auch ein $k_2 > 0$ geben, so daß $b^{k_1}ab^{k_2} \in K$ liegt, weil $b^{k_1}ab^\omega \in L$ gilt. In dieser Weise fortfahrend gibt es $k_i > 0$, so daß $b^{k_1}ab^{k_2}a \ldots ab^{k_i} \in K$ gilt für alle i. Nach Annhme über die Darstellbarkeit von L folgt daraus auch $b^{k_1}ab^{k_2}a \ldots ab^{k_i}a \ldots \in L$ im Widerspruch zur Definition von L.

∎

2.4.2 Abschlußeigenschaften

Satz 42 *Sind L_1, L_2 ω-reguläre Sprachen und ist K eine reguläre Sprache, dann ist auch*

1. $L_1 \cup L_2$ *ω-regulär,*
2. K^ω *ω-regulär, falls* $\varepsilon \notin K$,
3. KL_1 *ω-regulär,*
4. $V^\omega \setminus L_1$ *ω-regulär,*
5. $L_1 \cap L_2$ *ω-regulär.*

Beweis:
zu 1. Seien zwei ω-reguläre Sprachen $L_1, L_2 \subseteq V^\omega$ gegeben und $A_i = (Q_i, V, s_0^i, \delta_i, F_i)$ Büchi-Automaten mit $L_i = L_i^\omega(A_i)$. Wir können ohne Beschränkung der Allgemeinheit annehmen, daß $Q_1 \cap Q_2 = \emptyset$
Wir konstruieren einen Büchi-Automaten $A = (Q, V, s_0, \delta, F)$ wie folgt, wobei s_0 ein neuer Zustand ist, der weder in Q_1 noch in Q_2 vorkommt.

$$\begin{array}{lcll} Q & = & Q_1 \cup Q_2 \cup \{s_0\} & \\ \delta(q,a) & = & \delta_i(q,a) & \text{falls } q \in Q_i \\ \delta(s,a) & = & \delta_1(s_0^1, a) \cup \delta_2(s_0^2, a) & \\ F & = & F_1 \cup F_2 & \end{array}$$

Man zeigt leicht, daß $L^\omega(A) = L_1 \cup L_2$.
zu 2. Sei $A = (Q_A, V, s_0^A, \delta_A, F_A)$ ein nichtdeterministischer endlicher Automat mit $L(A) = K$. Wir definieren einen neuen Automaten $B = (Q_B, V, s_0^B, \delta_B, F_B)$ durch:

$$\begin{array}{lcll} Q_B & = & Q_A & \\ \delta_B(q,a) & = & \delta_A(q,a) & \text{falls } q \notin F_A \\ \delta_B(q,a) & = & \delta_A(q,a) \cup \delta_A(s_0^A, a) & \\ F_B & = & F_A & \end{array}$$

Es gilt $L^\omega(B) = K^\omega$

zu 3.	Übungsaufgabe 2.4.1
zu 4.	zurückgestellt auf später, Satz 47
zu 5.	folgt unmittelbar aus 1. und 4.

∎

Satz 43 *Eine Teilmenge $L \subseteq V^\omega$ ist eine ω-reguläre Menge von ω-Wörtern, genau dann, wenn L eine endliche Vereinigung von Mengen der Form*

$$JK^\omega$$

für reguläre Mengen $J, K \subseteq V^*$ ist, wobei $\varepsilon \notin K$.

Beweis:

Sei $A = (Q, V, s_0, \delta, F)$ ein Büchi-Automat mit $L^\omega(A) = L$. Für $p, q \in Q$ sei

$$L_{p,q} = \{w \in V^* \mid q \in \delta(p, w)\}$$

Offensichtlich ist jedes $L_{p,q} \subseteq V^*$ eine reguläre Menge. Man überzeugt sich leicht, daß

$$L = \bigcup_{p \in F} L_{s_0,p} L^\omega_{p,p}$$

gilt. Die umgekehrter Richtung der Behauptung folgt direkt aus Satz 42.

∎

Die Abgeschlossenheit ω-regulärer Sprachen unter Komplementbildung in V^ω muß noch bewiesen werden. Der Beweis bedarf einiger Vorbereitung. Eine zentrale Rolle spielt dabei die folgende Äquvalenzrelation.

Definition 54 *Sei $A = (Q, V, s_0, \delta, F)$ ein Büchi-Automat. Dann definieren wir für $p, q \in Q$ und $u, w \in V^*$*

1. $L_{p,q} = \{w \in V^* \mid q \in \delta(p, w)\}$
   (wie im Beweis von Satz 43)
2. $L^F_{p,q} = \{w = a_0 \dots a_k \in V^* \mid$ *es gibt eine Folge von Zuständen* $q_0, \dots q_k$ *mit*
$$\begin{array}{lcll} q_0 & = & p & \\ q_k & = & q & \\ q_{i+1} & \in & \delta(q_i, a_i) & \textit{für alle } 0 \le i < k \\ q_i & \in & F & \textit{für ein } 0 \le i \le \end{array} \}$$

3. *$u \equiv_A w$ gdw*
   für alle $p, q \in Q$ gilt
   $u \in L_{p,q} \;\Leftrightarrow\; w \in L_{p,q}$
   $u \in L^F_{p,q} \;\Leftrightarrow\; w \in L^F_{p,q}$

Offensichtlich gilt stets $L^F_{p,q} \subseteq L_{p,q}$ und falls $p \in F$ oder $q \in F$ auch $L^F_{p,q} = L_{p,q}$. Ebenso leicht sieht man, daß $\equiv_A$ eine Äquivalenzrelation ist.

Lemma 44 *Die Äquivalenzrelation $\equiv_A$ hat folgende Eigenschaften:*

1. *für alle $p, q \in Q$ sind $L^F_{p,q}$ und $L_{p,q}$ reguläre Mengen,*
2. *jede Äquivalenzklasse von $\equiv_A$ ist von der Form*
$$\bigcap_{(p,q)\in P} L_{p,q} \cap \bigcap_{(p,q)\notin P} \sim L_{p,q} \cap \bigcap_{(p,q)\in R} L^F_{p,q} \cap \bigcap_{(p,q)\notin R} \sim L^F_{p,q}$$
   für geeignete Teilmengen $P, R \subseteq Q \times Q$.
3. *$\equiv_A$ besitzt endlich viele Äquivalenzklassen,*
4. *jede Äquivalenzklasse von $\equiv_A$ ist eine reguläre Menge,*
5. *sind U, V Äquivalenzklassen von $\equiv_A$, dann folgt aus $UV^\omega \cap L^\omega(A) \neq \emptyset$ schon $UV^\omega \subseteq L^\omega(A)$*

Beweise:
zu 1 Für $L_{p,q}$ wurde das schon im Beweis von Satz 43 benutzt. Wegen $L^F_{p,q} = \bigcup_{f\in F} L_{p,f}L_{f,q}$ folgt auch die Regularität von $L^F_{p,q}$.
zu 2 folgt direkt aus der Definition von $\equiv_A$.
zu 3 unmittelbare Konsequenz aus 2.
zu 4 folgt aus 1, 2 und den Abschlußeigenschaften regulärer Mengen.
zu 5 Sei $w \in UV^\omega \cap L^\omega(A)$ gegeben und u ein weiteres Wort aus UV^ω. Es ist $u \in L^\omega(A)$ zu seigen.
Wegen $w \in UV^\omega$ können wir w wie folgt zerlegen:

$$w = w_1 w_2 \ldots w_n \ldots$$

wobei $w_1 \in U$ und $w_i \in V$ für alle $i > 1$. Wegen $w \in L^\omega(A)$ gibt es eine Berechnungsfolge σ für w in A in der unendlich viele Zustände aus F

auftreten. Wir interessieren uns für bestimmt Stützpunkte in der Folge σ; mit q_n bezeichen wir den Zustand in σ, der nach Einlesen des Teilwortes w_{n-1} und vor Beginn des Einlesens von w_n eingenommen wird. Insbesondere sei $q_0 = s_0$ der Anfangszustand. Sei schließlich $u \in UV^\omega$, genauer

$$u = u_1 u_2 \dots u_n \dots$$

mit $u_1 \in U$ und $u_i \in V$ für alle $i > 1$. Da U und V nach Voraussetzung $\equiv_A$ Äquivalenzklassen sind gilt für alle i

$$w_i \equiv_A u_i$$

Wir konstruieren eine Berechnungsfolge ρ für u wie folgt. Dabei wird ρ die Zustände q_n in derselben Reihenfolge durchlaufen, wie σ, aber welche Zustände auf dem Weg von q_n nach q_{n+1} auftreten ist von vornherein noch nicht festgelegt, ja es ist zunächst noch nicht einmal klar ob ein solcher Weg gefunden werden kann. Nach Definition der q_i gilt auf jeden Fall $w_n \in L_{q_{n-1},q_n}$. Wegen $w_n \equiv_A u_n$ gilt dann auch $u_n \in L_{q_{n-1},q_n}$. Man kann also den Weg von q_{n-1} nach q_n in ρ interpolieren. Kommt in σ ein Finalzustand zwischen q_{n-1} und q_n vor, so gilt $w_n \in L^F_{q_{n-1},q_n}$ und wegen $w_n \equiv_A u_n$ auch $u_n \in L^F_{q_{n-1},q_n}$. Das Teilstück von ρ zwischen q_{n-1} und q_n kann demnach auch so gewählt werden, daß darin ein Finalzustand vorkommt. Das zeigt, daß ρ eine akzeptierende Berechnungsfolge für u ist und damit $u \in L^\omega(A)$.

∎

Satz 45 (Satz von Ramsey)
Sei $P_1, \dots, P_k$ eine endliche Partition aller zweielementigen Teilmenge der Menge $\mathbb{N}$ der natürlicher Zahlen. Dann gibt es eine unendliche Teilmenge $T \subseteq \mathbb{N}$ und ein i mit $1 \leq i \leq k$, so daß alle zweielementigen Teilmengen von Elementen aus T in P_i liegen.
In Formeln:
aus $P_1 \uplus \dots \uplus P_k = \mathbb{N}^{[2]}$ folgt die Existenz einer unendlichen Menge $T \subseteq \mathbb{N}$, so daß $T^{[2]} \subseteq P_i$ für ein i.

Beweis: Siehe etwa [Monk, 1976] p.448 oder [Felscher, 1978] p. 228 oder die Orginalarbeit [Ramsey, 1929]. Wichtig bei der Ausgangssituation des Ramseyschen Satzes ist, daß für zwei natürliche Zahlen $n, m \in \mathbb{N}$ nicht

$(n,m) \in P_i$ und $(m,n) \in P_j$ für $i \neq j$ auftreten kann. In der vorliegenden Formulierung wurde das dadurch ausgeschlossen, daß anstelle von geordneten Paaren ungeordnete Zweiermengen betrachtet wurden. Häufig wird das Problem auch dadurch gelöst, daß nur Paare (n,m) mit $n \leq m$ betrachtet werden.

Lemma 46 *Zu jedem ω-Wort $w \in V^\omega$ gibt es eine $\equiv_A$-Äquivalenzklassen U und V mit $w \in UV^\omega$.*

Beweis:
Seien $U_1, \ldots, U_n$ alle Äquivalenzklassen von $\equiv_A$ und bezeichne für $i < j$ $w_{i,j}$ das endliche Teilwort $w(i) \ldots w(j-1)$ von w. Dann wird durch

$$P_r = \{\{i,j\} \in \mathbb{N}^{[2]} \mid [w_{i,j}]_{\equiv_A} = U_r\}$$

eine Partition von $\mathbb{N}^{[2]}$ definiert. Nach dem Satz von Ramsey gibt es eine unendliche Teilmenge $T \subseteq \mathbb{N}$ und ein i mit $T^{[2]} \subseteq P_i$. Sei i_0 das kleinste Element in T und $U = [w \downarrow (i_0)]_{\equiv_A}$, dann gilt offensichtlich

$$w \in UU_i^\omega$$

∎

Satz 47 *Ist $L \subseteq V^\omega$ eine ω-reguläre Menge, dann auch $V^\omega \setminus L$*

Beweis
Sei A ein Büchi Automat mit $L^\omega(A) = L$.

$$V^\omega \setminus L = \bigcup_{(U,V) \in \mathcal{R}} UV^\omega$$

dabei gilt $(U,V) \in \mathcal{R}$ genau dann, wenn $UV^\omega \cap L = \emptyset$.

∎

2.4.3 Müller Automaten

Definition 55 (Müller Automaten)
Ein Müller Automat $M = (S, V, s_0, \delta, \mathcal{F})$ *ist ein endlicher Automat* $M = (S, V, s_0, \delta)$ *ohne Angabe von Endzuständen, aber stattdessen mit einer Menge* $\mathcal{F}$ *von Endzustandsmengen, d.h. für alle* $F \in \mathcal{F}$ *gilt* $F \subseteq S$*.*
Ist $\sigma = s_1, \ldots, s_n, \ldots$ *eine Zustandsfolge, so bezeichnet* $In(\sigma)$ *die Menge der Zustände, die unendlich oft in* σ *vorkommen, also*

$$In(\sigma) = \{s \in S \mid \text{es gibt unendlich viele } n \text{ mit } s_n = s\}$$

Die von einem Müller Automat $M = (S, V, s_0, \delta, \mathcal{F})$ *akzeptierte* ω*-Sprache wird definiert durch:*

$$L^\omega(M) = \{w \in V^\omega \mid In(\sigma) \in \mathcal{F} \text{ für eine Berechnungsfolge } \sigma \text{ für } w\}$$

Lemma 48 *Die Klasse der von nichtdeterministischen Büchi Automaten akzeptierten* ω*-Sprachen stimmt überein mit der von nichtdeterministischen Müller Automaten akzeptierten* ω*-Sprachen.*

Beweis: Für die einfache Richtung, siehe Übungsaufgabe 2.4.4.

2.4.4 Definierbarkeit

Zu einem ω-Wort kann ganz analog zur Definition 39 die Struktur $\mathcal{M}_w$ gebildet werden. Das Universum von $\mathcal{M}_w$ wird dabei stets die Menge der natürlichen Zahlen sein.

Satz 49 *Sei* $L \subseteq V^\omega$*.*

$$\begin{gathered} L \text{ ist regulär} \\ \text{gdw} \\ \text{es gibt eine Formel } \Phi \text{ aus } \mathcal{L}_1^2 \text{ mit } L(\Phi) = L \end{gathered}$$

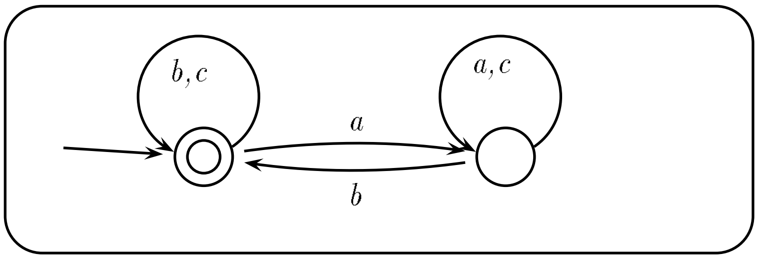
2.4.5 Übungsaufgaben

Übungsaufgabe 2.4.1

Ist L eine ω-reguläre Sprache und K eine reguläre Sprache, dann ist auch KL eine ω-reguläre Sprache.

Übungsaufgabe 2.4.2

Bestimmen Sie die von dem folgenden Büchi Automaten akzeptiert ω-Sprache in dem Alphabet $V = \{a, b, c\}$.



Übungsaufgabe 2.4.3

Der in Abbildung 2.6 gezeigte endliche Automat A ist nichtdeterministisch.

1. *Berechnen Sie den äquivalenten deterministischen endlichen Automaten A_1.*

2. *Bestimmen Sie $L^\omega(A_1)$.*

Übungsaufgabe 2.4.4

Finden Sie zu einem gegebenen (nicht deterministischen) Büchi-Automaten $A = (S_A, V, s_0^A, \delta_A, F)$ einen Müller-Automaten $M = (S_M, V, s_0^M, \delta_M, \mathcal{F})$ mit

$$L^\omega(A) = L^\omega(M)$$

Kapitel 3

Parsing

Die Darstellung in diesem Kapitel folgt im wesentlichen den Büchern [Aho & Ullmann, 1977], [Sippu & Soisalon-Soininen, 1988a], [Sippu & Soisalon-Soininen, 1988b], [Zima, 1988] , wo auch weiterführende Informationen nachgelesen werden können. Wir wiederholen zunächst einige grundlegende Definitionen für kontextfreie Grammatiken und legen unsere Notation fest.

Definition 56 (Kontextfreie Grammatiken)
Eine Grammatik $G = (N, T, P, S)$ ist gegeben durch

$$\begin{array}{lcl} N & = & \textit{Menge der Nichtterminalzeichen} \\ T & = & \textit{Menge der Terminalzeichen} \\ P & = & \textit{Menge der Produktionen} \\ S & = & \textit{Startzeichen} \\ V & = & N \cup T \end{array}$$

G heißt eine **kontextfreie** *Grammatik, wenn jede Produktion $\pi \in P$ die Form*

$$A \to w$$

hat mit $A \in N$ und $w \in V^*$.

Definition 57 (Grundlegende Notation)

$$\begin{array}{lcl} \Sigma^* & = & \textit{Menge aller Wörter über } \Sigma \\ \Sigma^+ & = & \textit{Menge aller nicht leeren Wörter über } \Sigma \\ \varepsilon & = & \textit{das leere Wort} \\ |w| & = & \textit{Länge des Wortes } w \end{array}$$

Für ein Wort $w = a_1 \dots a_n \in \Sigma^*$ $L \subseteq \Sigma^*$ *und eine natürliche Zahl* $k > 0$ *setzen wir*

$$k : w = \begin{cases} w & \textit{falls } n \leq k \\ a_1 \dots a_k & \textit{falls } n > k \end{cases}$$

$$k : L = \{k : w \mid w \in L\}$$

Definition 58 (Notation für ktf Grammatiken)

$w_1 \rightarrow^* w_2$	*es gibt eine Ableitung in* G *von* w_1 *nach* w_2
$w_1 \rightarrow^\pi w_2$	w_2 *kann durch eine Anwendung der Produktion* $\pi \in P$ *aus* w_1 *erhalten werden*
$w_1 \rightarrow_r^* w_2$	*es gibt eine Rechtsableitung in* G *von* w_1 *nach* w_2
$w_1 \rightarrow_r w_2$	w_2 *kann durch einen Schritt einer Rechtsableitung in* G *aus* w_1 *erhalten werden*
$w_1 \rightarrow_r^\pi w_2$	w_2 *kann durch eine Anwendung der Produktion* $\pi \in P$ *in einer Rechtsableitung aus* w_1 *erhalten werden*

$w_1 \rightarrow_r^\pi w_2$ gilt genau dann, wenn

$$\begin{aligned} w_1 &= uAv \text{ mit } u \in V^*, A \in N, v \in T^* \\ \pi &= A \rightarrow w \\ w_2 &= uwv \end{aligned}$$

Definition 59 (Erzeugte Sprache)

$$\begin{aligned} \mathcal{L}(G) &= \{\, w \in T^* \mid \; S \rightarrow w\} \\ & \quad \textit{die durch } G \textit{ erzeugte Sprache,} \\ \mathcal{L}_G(w) &= \{\, u \in T^* \mid \; w \rightarrow u\} \end{aligned}$$

Definition 60

Sei $G = (N, T, P, S)$ *eine kontextfreie Grammatik,* $k > 0$ *und* $w \in (N \cup T)^*$, *dann setzen wir:*

$$\begin{aligned} FIRST_{G,k}(w) &= k : \mathcal{L}_G(w) \\ FOLLOW_{G,k}(w) &= \{k : y \mid S \rightarrow_r^* uwy \textit{ für } u \in (N \cup T)^*, y \in T^*\} \end{aligned}$$

Falls die Referenz auf G aus dem Zusammenhang klar schreiben wir einfach $FIRST_k(w)$ und $FOLLOW_k(w)$ anstelle von $FIRST_{G,k}(w)$ und

$$\begin{array}{llll} 1 & S & \rightarrow & S+T \\ 2 & S & \rightarrow & T \\ 3 & T & \rightarrow & T \times F \\ 4 & T & \rightarrow & F \\ 5 & F & \rightarrow & (S) \\ 6 & F & \rightarrow & id \end{array}$$

Abbildung 3.1: Die kontextfreie Grammatik G_{AEXP}

$FOLLOW_{G,k}(w)$. Schließlich stehen $FIRST(w)$ und $FOLLOW(w)$ für $FIRST_1(w)$ und $FOLLOW_1(w)$
Man beachte, daß $\varepsilon \in FIRST_{G,k}(w)$ gdw $w \rightarrow_r^* \varepsilon$ und $\varepsilon \in FOLLOW_{G,k}(w)$ gdw $S \rightarrow_r^* uw$.

Definition 61 (Erweiterte Grammatik)
Zu einer kontextfreien Grammatik $G = (N, T, P, S)$ entsteht die **erweiterte Grammatik**

$$G' = (N \cup \{S'\}, T \cup \{\$\}, P \cup \{S' \rightarrow S\$\}, S')$$

durch Hinzunahme eines neuen Startsymbols und einer neuen Produktion. Manchmal nennt man G' genauer die \$-erweiterte Grammatik (engl. \$ - augmented grammar)

3.1 Einführendes Beispiel

In diesem Unterabschnitt wollen wir an einem Beispiel die Hauptaufgabe eines Parsers, auf deutsch eines Zerteilers, vorstellen. Das Beispiel ist aus [Aho & Ullmann, 1977] entnommen.
Ausgangspunkt ist eine gegebene Grammatik, z.B. die aus Abbildung 3.1. Gesucht ist ein Algorithmus, der für jedes Wort w aus $\{+, (,), \times, id\}^*$ entscheidet, ob $w \in \mathcal{L}(G_{AEXP})$ oder nicht. Einen solchen Algorithmus und auch das Programm, das diesen Algorithmus realisiert, nennen wir einen **Parser**.

Zustand	Aktion						Sprung		
	id	$+$	$\times$	(	)	\$	S	T	F
0	$s5$			$s4$			1	2	3
1		$s6$				acc			
2		$r2$	$s7$		$r2$	$r2$			
3		$r4$	$r4$		$r4$	$r4$			
4	$s5$			$s4$			8	2	3
5		$r6$	$r6$		$r6$	$r6$			
6	$s5$			$s4$				9	3
7	$s5$			$s4$					10
8		$s6$			$s11$				
9		$r1$	$s7$		$r1$	$r1$			
10		$r3$	$r3$		$r3$	$r3$			
11		$r5$	$r5$		$r5$	$r5$			

Abbildung 3.2: Zerteilertabelle für G_{AEXP}

Der wesentlichste Bestandteil eines Parsers ist die Parsertabelle, oder Zerteilertabelle, zu einer vorgegebenen Grammatik. In Abbildung 3.2 wird eine Parsertabelle für die Beispielgrammatik G_{AEXP} aus Abbildung 3.1 gezeigt. Die Parsertabelle nimmt insbesondere Bezug auf Zustände des Parsers, in dem G_{AEXP}-Beispiel gibt es elf Zustände. Um zu erklären, welche Rolle diese Zustände spielen und wie überhaupt die Parsertabelle benutzt wird, betrachten wir das in Abbildung 3.3 dargestellte Beispiel. Der Parser arbeitet mit zwei Datenstrukturen: einem Keller und der Eingabeliste. Zu Beginn steht im Keller nur das Symbol für den Anfangszustand, 0 in unserem Beispiel. Die Eingabeliste enthält das zu testende Wort. In unserem Beispiel wurde noch durch das Anhängen des Sonderzeichens \$ das Ende des Eingabewortes besonders markiert.

In Abhängigkeit von dem ersten noch nicht gelesenen Buchstaben des Eingabewortes, dem Kellerinhalt und der Parsertabelle wird die auszuführende Aktion ermittelt. Das oberste Kellerzeichen ist immer ein Symbol, das einen Zustand bezeichnet. Wir wollen ihn den aktuellen Zustand nennen. Als ersten Schritt für jede Aktion des Parsers ermitteln wir den Eintrag in die Parsertabelle für den aktuellen Zustand und den nächsten Buchstaben des Eingabewortes.

Zeile	Keller	Eingabe
1	0	$id \times id + id\$$
2	$0id5$	$\times id + id\$$
3	$0F3$	$\times id + id\$$
4	$0T2$	$\times id + id\$$
5	$0T2 \times 7$	$id + id\$$
6	$0T2 \times 7id5$	$+id\$$
7	$0T2 \times 7F10$	$+id\$$
8	$0T2$	$+id\$$
9	$0S1$	$+id\$$
10	$0S1 + 6$	$id\$$
11	$0S1 + 6id5$	$\$$
12	$0S1 + 6F3$	$\$$
13	$0S1 + 6T9$	$\$$
14	$0S1$	$\$$

Abbildung 3.3: Parsingbeispiel für G_{AEXP}

1. entält die Tabelle an der angegebenen Stelle keinen Eintrag, dann endet der parsing-Versuch mit einem Fehlschlag, das Eingabewort wird als nicht zum Sprachschatz gehörig zurückgewiesen,

2. wird aus der Tabelle *acc* abgelesen, dann wird das Eingabewort akzeptiert und keine weitere Aktion mehr ausgeführt,

3. in der Tabelle steht an der betreffenden Stelle si. Der Buchstabe s steht abkürzend für $shift$ und i gibt den nächsten aktuellen Zustand an. Entsprechend wird der nächste Buchstabe des Eingabewortes auf den Keller gelegt und darüber i. Aktionen von dieser Art treten in dem Beispiel aus Abbildung 3.3 beim Übergang zu den Zeilen 2, 5, 6, 10 und 11 auf,

4. der Tabelleneintrag kann schließlich noch die Form rj haben. Dabei steht r abkürzend für *reduce* und die Zahl j bezieht sich auf die Nummer der Produktion von G_{AEXP}, wie in Abbildung 3.1 angegeben. Sei $A \to v$ mit $v = b_1 \dots b_k$ die Produktion mit der Nummer j. Der Keller wird in diesem Fall das Endstück der Form $s_0 b_1 s_1 \dots s_{k-1} b_k j$ aufweisen.

Zunächst wird $b_1 s_1 \dots s_{k-1} b_k j$ durch die linke Seite der Produktion ersetzt, so daß jetzt der Keller auf $s_0 A$ endet. Der neue aktuelle Zustand wird jetzt wieder aus der Parsertabelle ermittelt unter dem Zustand s_0 und der Spalte für A. Besonders instruktiv sind in dem Beispiel in Abbildung 3.3 die reduce-Übergänge zu den Zeilen 8 und 14.

Die an dem Beispiel demonstrierte Vorgehensweise führt zu der folgenden formalen Definition eines LR-Automaten, einer Variante von Kellerautomaten. Alle später auftretenden Parser werden LR-Automaten sein.

Definition 62 (LR-Automaten)
Ein **LR-Automat** $M = (K, E, Z, PT)$ *gehört zu einer gegebenen kontextfreien Grammatik* $G = (N, T, P, S)$ *und wird durch die folgenden Angaben festgelegt:*

$$\begin{array}{lcl} K & = & \mathit{Kelleralphabet} \quad = \quad N \cup T \cup Z \\ E & = & \mathit{Eingabealphabet} \quad = \quad T \cup \{\$\} \\ Z & = & \mathit{Zustandsmenge} \\ PT & = & \mathit{Parsertabelle} \end{array}$$

Die Parsertabelle kann als graphische Darstellung zweier partieller Funktion betrachtet werden:

$$\delta_1 : Z \times (T \cup \{\$\})^* \to \mathit{Aktionen}$$
$$\delta_2 : Z \times N \to Z$$

wobei

$$\mathit{Aktionen} = \{(\mathbf{shift}, z) \mid z \in Z\} \cup P \cup \{acc\}$$

Im typischen Anwendungsfall wird δ_1 eine Funktion von $(T \cup \{\$\})$ in die Menge der Aktionen sein. Im allgemeinsten Fall kann jedoch an der zweiten Argumentstelle von δ_1 auch ein Wort auftreten, das aus mehr als einem Buchstaben besteht. Um in dieser möglichst allgemein gehaltenen Definition jedoch nicht zu weit von den in der Praxis auftretenden Fällen abzuweichen, wollen wir diese Möglichkeit nur zulassen, falls der Funktionswert von δ_1 an einer solchen Argumentstelle eine Produktion in P ist.
Analog zu der Definition für Kellerautomaten kann man Ableitungen für LR-Automaten definieren.

Definition 63 (Ableitungen für LR-Automaten)
Zustandsbeschreibungen für M sind

$$z_0 a_1 z_1 a_2 \ldots a_m z_m \mid c_1 c_2 \ldots c_k \$$$

mit $z_i \in Z$, $a_i \in N \cup T$, $c_i \in T$.

Transitionen:

$$\frac{z_0 a_1 z_1 a_2 \ldots a_m z_m \mid c_1 c_2 \ldots c_k \$}{z_0 a_1 z_1 a_2 \ldots a_m z_m c_1 z \mid c_2 \ldots c_k \$}$$

falls $\delta_1(z_m, c_1) = (\mathbf{shift}, z)$.

$$\begin{array}{l} z_0 a_1 z_1 a_2 \ldots z_{m-n} a_{m-n+1} \ldots a_m z_m \mid c_1 c_2 \ldots c_k \$ \\ \hline z_0 a_1 z_1 a_2 \ldots z_{m-n} A \mid c_1 c_2 \ldots c_k \$ \\ \hline z_0 a_1 z_1 a_2 \ldots z_{m-n} A z \mid c_1 c_2 \ldots c_k \$ \end{array}$$

$$\begin{array}{lll} \textit{falls } \delta_1(z_m, c_1 \ldots c_r) \textit{ mit } r \leq k & = A \rightarrow a_{m-n+1} \ldots a_m & \textit{und} \\ \quad \delta_2(z_{m-n}, A) & = z & \end{array}$$

3.1.1 Übungsaufgaben

Übungsaufgabe 3.1.1 *Sei $G = (N, T, P, S)$ eine kontextfreie Grammatik und G' ihre \$ -Erweiterung nach Definition 61. Zeigen Sie, daß für jedes Wort $w \in V^*$ gilt:*

$$\begin{array}{c} S \rightarrow^*_G w \\ gdw \\ S' \rightarrow^*_{G'} w\$ \end{array}$$

3.2 Präfixe

Definition 64 (LR(k)-Elemente) *Sei $G = (N, T, P, S)$ eine kontextfreie Grammatik.*

$$
\begin{array}{llllllll}
1 & S & \rightarrow & aA & 2 & S & \rightarrow & bB \\
3 & A & \rightarrow & c & 4 & A & \rightarrow & dAd \\
5 & B & \rightarrow & c & 6 & B & \rightarrow & dBd
\end{array}
$$

$$\mathcal{L}(G_{ab}) \;=\; r\{ad^n cd^n \mid n \geq 0\} \cup \{bd^n cd^n \mid n \geq 0\}$$

Abbildung 3.4: Die Beispielgrammatik G_{ab}

1. *$A \rightarrow u \cdot v$ heißt eine punktierte Produktion für G, falls $A \rightarrow uv \in P$.*

2. *$[A \rightarrow u{\cdot}v, y]$ heißt ein LR(k)-Element (engl. LR(k)-item) falls $A \rightarrow u{\cdot}v$ eine punktierte Produktion für G und $y \in T^*$ mit $\mid y \mid \leq k$*
   Ein LR(0)-Element ist damit nichts anderes als eine punktierte Produktion.

Definition 65 *In der folgenden Situation*

$$S \rightarrow_r^* u_0Ay \rightarrow_r^{\pi} u_0uvz$$

$$mit\ u_0 \in V^*,\ z \in T^*,\ A \in N\ und\ \pi = A \rightarrow uv$$

heißt das Wort $w = u_0u$ ein **lebensfähiger Präfix** *(engl. viable prefix) von G*
und
$[A \rightarrow u \cdot v, y]$ ein **gültiges** *LR(k)-Element für w, falls $y = k : z$.*

Beispiel 13
Um die lebensfähigen Präfixe für G_{ab} zu bestimmen betrachten wir den

vollständigen Ableitungsbaum für G_{ab}:

$$
\begin{array}{cccc}
 & & S & \\
 & aA & bB & \\
ac & adAd & bdBd & bc \\
adcd & ad^2Ad^2 & bd^2Bd^2 & bdcd \\
ad^2cd^2 & ad^3Ad^3 & bd^3Bd^3 & bd^2cd^2 \\
\ldots & \ldots & \ldots & \ldots
\end{array}
$$

Hieraus läßt sich die Menge aller lebensfähigen Präfixe wie folgt ablesen:

$$
\begin{array}{lclc}
\{ad^n \mid n \geq 0\} & \cup & \{ad^nc \mid n \geq 0\} & \cup \\
\{ad^nA \mid n \geq 0\} & \cup & \{ad^nAd \mid n \geq 0\} & \cup \\
\{bd^n \mid n \geq 0\} & \cup & \{bd^nc \mid n \geq 0\} & \cup \\
\{ad^nB \mid n \geq 0\} & \cup & \{ad^nBd \mid n \geq 0\} & \cup \\
\{\varepsilon\} & \cup & \{S\} &
\end{array}
$$

Um eine erste Vorstellung von der Nützlichkeit der Kenntniss aller lebensfähigen Präfixe zu geben, betrachten wir zwei LR-Parsingversuche in der Grammatik G_{ab} für das Wort $adcd$

1		$adcd$		$adcd$
2	a	dcd	a	dcd
3	ad	cd	ad	cd
4	adA	d	adB	d
5	$adAd$		$adBd$	
6	aA		aB	
7	S		*Fehlschlag*	

In Zeile 5 ist $adAd$ ein lebensfähiger Präfix für G_{ab}, aber $adBd$ nicht. Das Scheitern des zweiten Parsingversuchs hätte schon hier entdeckt werden können.

Die Auflistung aller Ableitungen einer Grammatik zur Bestimmung aller lebensfähigen Präfixe ist natürlich nur für kleinste Beispielgrammatiken möglich. Wir wenden uns jetzt der Frage zu, wie die lebensfähigen Präfixe auch für große Grammatiken effizient berechnet werden können.

Definition 66 (Die nichtdeterministische LR(k)-Maschine)
Sei $G = (N, T, P, S)$ eine kontextfreie Grammatik, G' ihre \$ -Erweiterung, dann ist die LR(k)-Maschine $M_{G,k} = (Q, V, q_0, \delta, F)$, ein nichtdeterministischer endlicher Automat mit spontanen Übergängen, wie folgt definiert:

$$
\begin{array}{lcll}
Q & = & \textit{alle } LR(k)\textit{-Elemente von } G', & \\
V & = & N \cup T, & \\
q_0 & = & [S' \to \cdot S\$, \varepsilon] & \\
F & = & Q, & \\
[A \to uX \cdot v, y] & \in & \delta([A \to u \cdot Xv, y], X) & \textit{für } X \in V \\
[B \to \cdot v_1, z] & \in & \delta([A \to u \cdot Xv, y], \varepsilon) & \textit{falls } X \in N \\
 & & & \textit{und } z \in FIRST_k(vy)
\end{array}
$$

Die Tatsache, daß jeder Zustand von $M_{G,k}$ ein Finalzustand ist zeigt, daß es auf Finalzustände hier nicht ankommt. Die Auswahl der Wörter, die $M_{G,k}$ akzeptieren soll, wird durch die Partialität der Transitionsfunktion δ getroffen.

Lemma 50
Sei $G = (N, T, P, S)$ eine kontextfreie Grammatik, G' ihre \$ -Erweiterung. Für $w \in V^*$ gilt

$$
\begin{array}{c}
w\$ \in L(M_{G,k}) \\
\textit{gdw} \\
w \textit{ ist ein lebensfähiger Präfix für } G.
\end{array}
$$

Beweis:
Wir zeigen für jedes $w \in V^*$ und jedes LR(k)-Element $[A \to u \cdot v, y]$

$$
\begin{array}{c}
[A \to u \cdot v, y] \in \delta(q_0, w) \\
\text{gdw} \\
S \to_r^* u_0 A z \to_r u_0 u v z
\end{array}
$$

mit

- $w = u_0 u$, $A \to uv \in P$ und $y = k : z$

Wir zeigen zunächst die Implikation von oben nach unten durch Induktion über die Definition von δ.
Der Induktionsanfang, $w = \varepsilon$ ist offensichtlich.
Sei jetzt

$$[A \to uX \cdot v_0, y] \in \delta(q_0, wX)$$

mit

$$[A \to u \cdot Xv_0, y] \in \delta(q_0, w)$$

Nach Induktionsvoraussetzung gilt

$$S \to_r^* u_0 Az \to_r u_0 uXv_0 z$$

mit

- $w = u_0 u$, $A \to uXv_0 \in P$ und $y = k : z$

Diese Beziehungen bleiben ganz offensichtlich erhalten, wenn der Punkt um eine Stelle nach rechts verschoben wird.
Es bleibt der Fall eines spontanen Übergangs zu behandeln. Hier gehen wir aus von einem Zustand $[A \to u \cdot Bv_0, y] \in \delta(q_0, w)$ mit $B \in N$, für den die Induktionsvoraussetzung gilt, d.h.:

$$S \to_r^* u_0 Az \to_r u_0 uBv_0 z$$

mit

- $w = u_0 u$, $A \to uBv_0 \in P$ und $y = k : z$

Die Behauptung ist dann für einen Zustand $[B \to \cdot v_1, y_1]$ mit $y_1 \in FIRST_k(v_0 y)$ zu zeigen. Nach Definition von $FIRST_k$ gibt es eine Ableitung in G der Form

$$v_0 y \to_r^* y_2 \text{ mit } y_2 \in T^*, y_1 = k : y_2$$

Wegen $y = k : z$ gilt dann auch

$$v_0 z \rightarrow_r^* z_1 \text{ mit } y_1 \in T^*, y_1 = k : z_1$$

Die nach Induktionsvoraussetzung existierende Ableitung $S \rightarrow_r^* u_0 u B v_0 z$ läßt sich somit zu einer Ableitung $S \rightarrow_r^* u_0 u B z_1$ fortsetzen. Zusammengefasst erhalten wir:

$$S \rightarrow_r^* u_0 u B z_1 \rightarrow_r u_0 u v_1 z_1$$

mit

- $w = u_0 u$, $B \rightarrow v_1 \in P$, $y_1 = k : z_1$

Zum Beweis der umgekehrten Implikation gehen wir von einer Herleitung

$$S \rightarrow_r^* u_0 A z \rightarrow_r u_0 u v z$$

aus und wollen $[A \rightarrow u \cdot v, y] \in \delta(q_0, w)$ zeigen. Das geschieht durch Induktion über die Länge der Herleitung und über $|u|$. Der Induktionsanfang, $u_0 = z = \varepsilon, A = S$, ist wieder einfach. Auch der Fall, daß die Länge der Herleitung unveränder bleibt und nur das mit u bezeichnete Teilwort um einen Buchstaben länger wird, ist trivial. Es bleibt also die Situation

$$S \rightarrow_r^* u_0 A z \rightarrow_r u_0 u B v_0 z \rightarrow_r u_0 u v_1 v_0 z$$

zu betrachten mit $z, v_0 \in T^*$ und $B \rightarrow v_1 \in P$. Nach Induktionsvoraussetzung gilt

$$[A \rightarrow u \cdot B v_0, y] \in \delta(q_0, w)$$

für $w = u_0 u$ und $y = k : z$. Zu zeigen ist

$$[B \rightarrow \cdot v_1, y_1] \in \delta(q_0, w)$$

für $y_1 = k : v_0 z$. Mit anderen Worten, es ist zu seigen, daß in dem Automaten $M_{G,k}$ ein spontaner Übergang von $[A \rightarrow u \cdot B v_0, y]$ nach $[B \rightarrow \cdot v_1, y_1]$ möglich ist. Betrachtet man die Definition 66 , dann ist dafür nur noch $y_1 \in FIRST_k(v_0 y)$ zu zeigen. Die Wörter v_0,y,z sind im vorliegenden Fall alle terminale Wörter, so daß $FIRST_k(v_0 y) = \{k : v_0 y\}$ gilt und wir sind fertig.

∎

Die nichtdeterministische LR(0)-Maschine für die Beispielgrammatik G_{ab} (siehe Abb.3.2) ist in Abbildung 3.5 zu sehen.

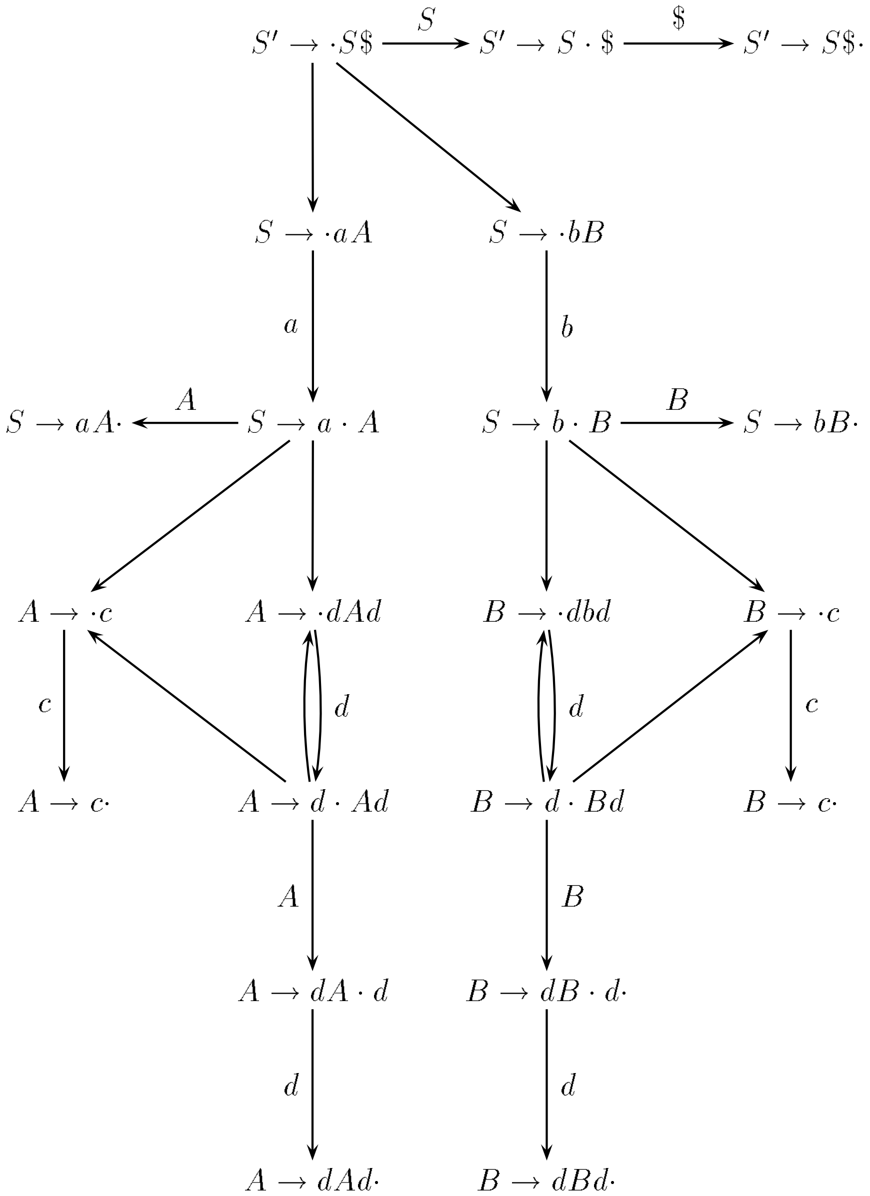


Abbildung 3.5: Der nichtdeterministische LR(0)-Automat für G_{ab}

Definition 67 (Die deterministische LR(k)-Maschine)
Sei $G = (N, T, P, S)$ eine kontextfreie Grammatik, dann ist die deterministische LR(k)-Maschine für G, bezeichnet mit $D_{G,k}$, der zu $M_{G,k}$ äquivalente deterministische endliche Automat.

Die Konstruktion von $D_{G,k}$ aus $M_{G,k}$ erfolgt mit dem Verfahren, das im Beweis von Satz 1 beschrieben wurde. Die deterministische LR(0)-Maschine für G_{ab} ist in Abbildung 3.6 zu sehen.

Lemma 51 *Für die deterministische LR(k)-Maschine $D_{G,k} = (Q, V, q_0, \Delta)$ zur ktf. Grammatik $G = (N, T, S, P)$ ist für alle $w \in V^*$, so daß $\Delta(q_0, w)$ definiert ist, $\Delta(q_0, w)$ die Menge aller gültigen LR(k)-Elemente für w.*

Beweis:
Folgt unmittelbar aus der Definition gültiger LR(k)-Elemente (Def.65), aus Lemma 50 und dem Zusammenhang $\bar{\delta}(s_0, w) = \Delta(q_0, w)$ zwischen der nichtdeterministischen LR(k)-Maschine, $M_{G,k} = (S, V, s_0, \delta)$ und der deterministischen $D_{G,k} = (Q, V, q_0, \Delta, F)$ für G. Siehe dazu Satz 1.

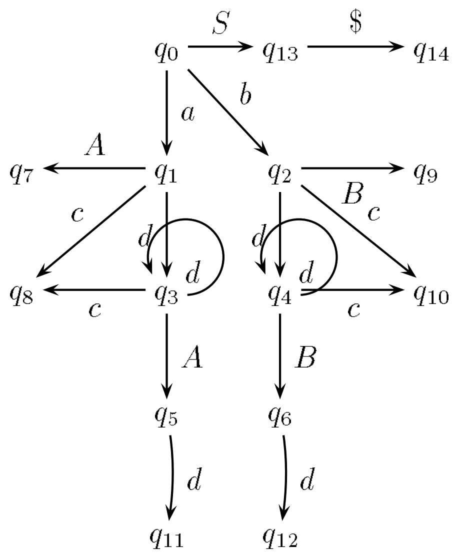
■

3.2.1 Übungsaufgaben

Übungsaufgabe 3.2.1 *Sei w ein lebensfähiger Präfix für die kontextfreie Grammatik G.*
Zeigen Sie, daß dann auch jedes Anfangsstück von w ein lebensfähiger Präfix für G ist.

3.3 LR(k)-Parser

Definition 68 (Der kanonische LR(k)-Parser) *$G = (N, T, P, S)$ sei ktf Grammatik, $D_{G,k} = (Q, V, q_0, \Delta, F)$ die zugehörige deterministische LR(k)-Maschine.*
Der kanonische LR(k)-Parser für G ist der durch die folgenden Zuordnungen festgelegte LR-Automat $K = (Z, \delta_1, \delta_2)$



$$
\begin{array}{lcllcl}
q_0 & = & \{S' \to \cdot S\$, S \to \cdot aA, S \to \cdot bB\} & & & \\
q_1 & = & \{S \to a \cdot A, A \to \cdot c, A \to \cdot dAd\} & q_2 & = & \{A \to dA \cdot d\} \\
q_3 & = & \{A \to d \cdot Ad, A \to \cdot c, A \to \cdot dAd\} & q_4 & = & \{S \to aA \cdot\} \\
q_5 & = & \{S \to b \cdot B, B \to \cdot c, B \to \cdot dBd\} & q_6 & = & \{B \to dB \cdot d\} \\
q_7 & = & \{B \to d \cdot Bd, B \to \cdot c, B \to \cdot dBd\} & q_9 & = & \{S \to bB \cdot\} \\
q_8 & = & \{A \to c \cdot\} & q_{10} & = & \{B \to c \cdot\} \\
q_{11} & = & \{A \to dAd \cdot\} & q_{12} & = & \{B \to dBd \cdot\} \\
q_{13} & = & \{S' \to S \cdot \$\} & q_{14} & = & \{S' \to S\$ \cdot\}
\end{array}
$$

Abbildung 3.6: Der deterministische LR(0)-Automat für G_{ab}

q	a	b	c	d	$	A	B	S
0	s1	s2						13
1			s8	s3		7		
2			s10	s4			9	
3			s8	s3		5		
4			s10	s4			6	
5				s11				
6				s12				
7	r1	r1	r1	r1	r1			
8	r3	r3	r3	r3	r3			
9	r2	r2	r2	r2	r2			
10	r5	r5	r5	r5	r5			
11	r4	r4	r4	r4	r4			
12	r6	r6	r6	r6	r6			
13					acc			

Abbildung 3.7: LR(0)-Parsertabelle für G_{ab}

- *Die Zustände von K sind die Zustände von M_k*
- $\delta_1(q, c) = (\underline{s}hift, \Delta(q, c))$ *für* $q \in Q$, $c \in T$ *falls* $\Delta(q, c)$ *definiert ist.*
- $\delta_2(q, A) = \Delta(q, A)$ *für* $q \in Q$, $A \in N$ *falls* $\Delta(q, A)$ *definiert ist.*
- $\delta_1(q, y) = p$ *falls eine Element* $[A \to u\cdot, y]$ *in* q *liegt und* $p = A \to u$.

Diese Definition ist hauptsächlich für $k = 0$ und $k = 1$ interessant. Für die Grammatik G_{ab} erhält man aus der deterministischen LR(0)-Maschine, siehe Abbildung 3.6 die in Abbildung 3.7 gezeigte LR(0)-Parsertabelle.

Satz 52

Sei $G = (N, T, P, S)$ *eine kontextfreie Grammatik,* $k \geq 0$, *und* $K = (Z, \delta_1, \delta_2)$ *der zugehörige LR(k)-Parser, dann gilt*

$$\begin{gathered} K \textit{ akzeptiert ein Wort } w \in T^* \\ gdw \\ \textit{in } G \textit{ gilt } S \rightarrow_r^* w \end{gathered}$$

Beweis:
1.Teil: K akzeptiere w
Sei

$$\begin{array}{ll} e_0 & w_0 \\ e_1 & w_1 \\ \dots & \dots \\ \dots & \dots \\ e_m & w_m \end{array}$$

eine akzeptierende Berechnungsfolge für K. Dabei steht e_j für den jeweiligen Kellerinhalt und w_j für den noch nicht gelesenen Teil des Eingabewortes. Insbesondere gilt $w_0 = w$, $w_m = \varepsilon$, $e_0 = q_0$ und $e_m = Sq_k$.
Das Wort e_i' entstehe aus e_i indem alle Namen für Zustände weggelassen werden, anders gesagt enthält e_i' nur die Zeichen aus e_i, die auch in $N \cup T$ liegen. Sei

$$u_0 = e_0, u_1 = e_{j(1)}, \dots, u_i = e_{j(i)}, \dots u_k = e_{j(k)} = e_m$$

die Teilfolge aus $e_0, \dots, e_m$, die abgesehen von dem ersten Element nur solche Indixes $j(i)$ berücksichtigt, für die der Übergang von $e_{j(i)-1}$ nach $e_{j(i)}$ durch eine Reduktionsaktion von K entsteht. Schließlich sei $v_i = u_i w_{j(i)}$.
Man sieht jetzt leicht, daß

$$v_k = S, v_{k-1}, \dots, v_0 = w$$

eine Ableitung von w in G ist.

2.Teil: in G gelte $S \rightarrow_r^* w$
Es ist also eine Folge von Wörtern $S = w_0, w_1, \dots, w_n = w$ gegeben mit $w_i \rightarrow_r w_{i+1}$. Genauer haben wir

$$\begin{array}{lcll} w_i & = & u_i A_i z_i & \text{mit } z_i \in T^* \\ w_{i+1} & = & u_i v_i z_i & \text{mit } A_i \rightarrow v_i \in P \end{array}$$

Es ist wichtig zu bemerken, daß die Benutzung von Rechtsableitungen zur Folge hat:

$$z_{i-1} \text{ ist ein Endstück von } z_i$$

für alle $1 \leq i \leq n$.

Wir wollen zeigen, daß eine Ableitung des kanonischen LR(k)-Parsers der folgenden Form möglich ist.

Keller	Eingabe
ε	$w_n = u_{n-1}v_{n-1}z_{n-1}$
...	...
...	...
$u_{n-1}v_{n-1}$	z_{n-1}
$u_{n-1}A_{n-1}$	z_{n-1}
...	...
...	...
$u_{n-2}v_{n-2}$	z_{n-2}
$u_{n-2}A_{n-2}$	z_{n-2}
...	...
...	...
$w_1 = v_0$	ε
$w_0 = S$	ε

Die ausgelassenen Zeilen sollen durch shift-Aktionen aufgefüllt werden. Die erste Frage ist: geht das überhaupt? Kann man durch shift-Aktionen z.B. von

$$u_{n-1}A_{n-1} \quad | \; z_{n-1}$$

nach

$$u_{n-2}v_{n-2} \quad | \; z_{n-2}$$

kommen. Nach Definition gilt

$$u_{n-1}A_{n-1}z_{n-1} = u_{n-2}v_{n-2}z_{n-2}$$

und da z_{n-2} ein Endstück von z_{n-1} ist kommt A_{n-1} irgendwo in $u_{n-2}v_{n-2}$ vor und unsere Frage findet eine positive Antwort. Jetzt müssen wir die für oben angegebene Ableitung benutzten Zustände des kanonischen LR(k)-Parsers rekonstruieren. Aber das ist einfach. Jedes Kellerwort u ist offensichtlich ein lebensfähiger Präfix für G, wird also von dem Automaten $D_{G,k}$ akzeptiert. Der aktuelle Zustand in jeder Zeile sei $\Delta(q_0, u)$, der Zustand von $D_{G,k}$, der nach dem Einlesen von u erreicht wird. Man sieht jetzt leicht, daß die Transformationsregeln für K ein der obigen Ableitung eingehalten werden. Ist z.B. das Kellerwort $u_{n-1}v_{n-1}$ mit dem aktuellen Zustand q und wurde beim Übergang in die nächste Zeile eine Reduktion mit $A_{n-1} \to v_{n-1}$ vorgenommen, so ist $[A_{n-1} \to v_{n-1}., k : z_{n-1}]$ ein Element von q.

▪

Definition 69 (LR(k)-Grammatiken)
Eine Grammatik G heißt eine **LR(k)-Grammatik**, *wenn der kanonische LR(k)-Parser deterministisch ist, d.h. in jedem Feld der LR(k)-Parsertabelle für G steht höchstens ein Eintrag.*

Offensichtlich sind G_{AEXP} und G_{ab} Beispiele für LR(0)-Grammatiken.

Definition 70 (Konflikte)
Sei $D_{G,k}$ der deterministische LR(k)-Automat zur Grammatik G.
Ein Zustand q von $D_{G,k}$ enthält einen **reduce-reduce-Konflikt**, *wenn es zwei verschiedene Produktionen $A \to v$, $B \to u$ und ein Wort $y \in k : T^*$ gibt, so daß*

$$[A \to v\cdot, y] \in q \text{ und } [B \to u\cdot, y] \in q$$

Ein Zustand q von $D_{G,k}$ enthält einen **shift-reduce-Konflikt**, *wenn es zwei Produktionen $A \to vcw$, $B \to u$ mit $c \in T$ und Wörter $y, z \in k : T^*$ mit $1 : y = c$ gibt, so daß*

$$[A \to v \cdot cw, z] \in q \text{ und } [B \to u\cdot, y] \in q$$

Durch Auflösen der Definitionen läßt sich mühelos das folgende Lemma zeigen:

Lemma 53
Eine Grammatik G ist genau dann eine LR(k)-Grammatik, wenn ihr deterministischer LR(k)-Automat weder einen reduce-reduce-Konflikt noch einen shift-reduce-Konflikt enthält.

$$
\begin{array}{llcllllcl}
1 & S & \rightarrow & aA & \quad & 2 & S & \rightarrow & bB \\
3 & A & \rightarrow & \varepsilon & & 4 & A & \rightarrow & cAd \\
5 & B & \rightarrow & \varepsilon & & 6 & B & \rightarrow & cBd
\end{array}
$$

$$\mathcal{L}(G_{ab\varepsilon}) \;=\; r\{ac^n d^n \mid n \geq 0\} \cup \{bc^n d^n \mid n \geq 0\}$$

Abbildung 3.8: Die Beispielgrammatik $G_{ab\varepsilon}$

q	a	b	c	d	$	A	B	S
1	s3	s4						2
2					acc			
3			s7			5		
4			s8				6	
5					r1			
6					r2			
7			s13	r3		9		
8			s14	r5			10	
9				s11				
10				s12				
11					r4			
12					r6			
13			s13			15		
14			s14				16	
15				s17				
16				s18				
17				r4				
18				r6				

Abbildung 3.9: Kanonische LR(1)-Parsertabelle für $G_{ab\varepsilon}$

Satz 54
Ist G eine LR(k)-Grammatik, dann ist G auch eine LR(k')-Grammatik für jedes $k' \geq k$.

Beweis : Siehe z.B. [Sippu & Soisalon-Soininen, 1988b], Theorem 6.40, Seite 53.

Satz 55
Für jedes $k \geq 1$ gibt es zu jeder LR(k)-Grammatik eine äquivalente LR(1)-Grammatik.

Beweis: Siehe [Sippu & Soisalon-Soininen, 1988b], Abschnitt 6.7.

3.3.1 Übungsaufgaben

Übungsaufgabe 3.3.1 *Zeichnen Sie das Transitionsdiagramm des nichtdeterministischen LR(0)-Automaten, $M_{G_{AEXP},0}$, für die Grammatik G_{AEXP} aus Abbildung 3.1.*

3.4 LALR(k)-Parser

Definition 71 (Kern eines LR(k)-Elements)
Ist $[A \to u \cdot w, y]$ ein LR(k)-Element, so nennen wir die erste Komponente $A \to u \cdot w$, den **Kern** *(core) des LR(k)-Elements. Der Kern einer Menge Q von LR(k)-Elementen ist die Menge der Kerne ihrer Elemente, d.h. $Kern(Q) = \{Kern(q) \mid q \in Q\}$*

Lemma 56 *Sei $D_{G,k} = (Q, V \cup \{\$\}, \Delta)$ der deterministische LR(k)-Automat für die kontextfreie Grammatik G, und $q_1, q_2 \in Q$ zwei Zustände. Dann gilt für alle $X \in V \cup \{\$\}$:*

$$\textit{wenn } kern(q_1) = kern(q_2) \textit{ dann } kern(\Delta(q_1, X)) = kern(\Delta(q_2, X))$$

Beweis:
Ist $M_{G,k} = (S, V \cup \{\$\}, \delta)$ der nichtdeterministische LR(k)-Automat für G, dann genügt es wegen

$$\begin{aligned} \Delta(q, X) &= \bar{\delta}(q, X) \\ &= \varepsilon\text{-}cl(\{s \in S \mid \exists s_1 \in \bar{\delta}(q, \varepsilon) \wedge s \in \delta(s_1, X)\}) \end{aligned}$$

1. für $s_1, s_2 \in S$ mit $kern(s_1) = kern(s_2)$ auch $kern(\delta(s_1, X)) = kern(\delta(s_2, X))$ und

2. für $Q_1, Q_2 \subseteq S$ mit $kern(Q_1) = kern(Q_2)$ auch $kern(\varepsilon\text{-}cl(Q_1)) = kern(\varepsilon\text{-}cl(Q_1))$

zu zeigen. Diese beiden Behauptungen lassen sich aber leicht aus den beteiligten Definitionen herleiten.

∎

Die Anzahl der Zustände eines LR(1)-Parsers nimmt noch sehr rasch mit der Größe der Grammatik zu, wobei man zusätzlich beobachtet, daß sich manche Zustände nur geringfügig voneinander unterscheiden. Diese Beobachtung führte zur Einführung der LALR-Parser, die heute in der Praxis hauptsächlich benutzt werden.

$$
\begin{array}{llll}
0 & S' & \rightarrow & S\$ \\
1 & S & \rightarrow & CC \\
2 & C & \rightarrow & cC \qquad 3 \quad C \rightarrow d
\end{array}
$$

Abbildung 3.10: Die Beispielgrammatik G_{CC}

Definition 72
Sei $D_{G,k}$ der deterministische LR(k)-Automat für die Grammatik G.
Für $q \in Q$ setzen wir

$$\bar{q} = \bigcup \{q_1 \in Q \mid kern(q) = kern(q_1)\}$$

Dann konstruiert man den LA-Automaten für G, bezeichnet mit $LA_{G,k} = (\bar{Q}, V \cup \{\$\}, \bar{\Delta})$ wie folgt:

$$
\begin{array}{lcl}
\bar{Q} & = & \{\bar{q} \mid q \in Q\} \\
\bar{\Delta}(\bar{q}, X) & = & \Delta(q, X)
\end{array}
$$

Damit diese Definition Sinn macht muß man wissen, daß aus $\bar{q_1} = \bar{q_2}$ auch $\Delta(q_1, X) = \Delta(q_1, X)$ gilt. Aber das folgt leicht aus dem vorangegangen Lemma 56 .

Wir betrachten zur Illustration die Beispielgrammatik G_{CC} aus [Aho & Ullmann, 1977] in Abbildung 3.4. Abbildung 3.4 zeigt eine LR(1)-Grammatik, die keine LALR(1)-Grammatik ist, ebenfalls aus [Aho & Ullmann, 1977] entnommen.

3.5 Bison

Zu den in der Unix-welt bekanntesten Compilergeneratoren, die aus kontextfreien Grammatiken LALR(1)-Parser erzeugen gehören YACC und Bison. Eine Eingabedatei für das Bisonprogramm hat die in Abbildung 3.14 gezeigte Form und wird eine Bison Grammatikdatei genannt. Dabei sind '`%%`', '`%{`' und '`%}`' Interpunktionszeichen, die in jeder Bison Grammatikdatei die verschiedenen Abschnitte voneinander trennen.

q	c	d	$	S	C
0	s3	s4		1	2
1			acc		
2	s6	s7			5
3	s3	s4			8
4	r3	r3			
5			r1		
6	s6	s7			9
7			r3		
8	r2	r2			
9			r2		

Abbildung 3.11: Kanonische LR(1)-Parsertabelle für G_{CC}

q	c	d	$	S	C
0	s36	s47		1	2
1			acc		
2	s36	s47			5
36	s36	s47			89
47	r3	r3			
5			r1		
89	r2	r2	r2		

Abbildung 3.12: LALR(1)-Parsertabelle für G_{CC}

$$
\begin{array}{llcl}
0 & S' & \to & S\$ \\
1-4 & S & \to & aAd \mid bBd \mid aBe \mid bAe \\
5 & A & \to & c
\end{array}
\qquad
\begin{array}{llcl}
6 & B & \to & c
\end{array}
$$

$$\mathcal{L}(G_4) = \{acd, bcd, ace, bce\}$$

Abbildung 3.13: Die Beispielgrammatik G_4

```
%{
C DECLARATIONS
%}

BISON DECLARATIONS

%%
GRAMMAR RULES
%%

ADDITIONAL C CODE
```

Abbildung 3.14: Format von Eingabedateien in Bison

Die C Deklarationen können Typen und Variablen definieren, die in dem Aktionsteil benutzt werden. Als ein einfaches Beispiel für eine Bison Grammatikdatei ist in Abbildung 3.15 die Bisonversion der Grammatik G_{CC} aus Abbildung 3.4 gezeigt. Nichtterminalzeichen werden mit kleinen Buchstaben geschrieben, Terminalzeichen in Anführungszeichen gesetzt. Anstelle des Pfeils, den wir bisher in der Darstellungen von Produktionsregeln benutzt haben, steht hier nach dem Nichtterminalzeichen auf der linken Seite ein Doppelpunkt :. In diesem Beispiel gibt es keine Deklarationen und keine Aktionsteile. Als Ausgabe produziert das Bison Programm C Kode für einen Parser, der uns hier nicht besonders interessiert. Läßt man Bison mit der *verbose* Option laufen, so erhält man zusätzlich einen Ausdruck der LALR(1)-

```
%%
s:     c c
;
c:     'c' c
     | 'd'
;
%%
```

Abbildung 3.15: Bison Grammatikdatei für G_{CC}

Parsertabelle. Für die Eingabegrammatik G_{CC} ist das Ergebnis in Abbildung 3.16 zu sehen. Am Anfang werden noch einmal alle Produktionsregeln wiederholt mit einer Nummerierung, auf die auch später zurückgegriffen wird. Danach folgt eine Querverweistabelle, wo welche Terminal- und Nichtterminalzeichen auftreten, gefolgt von der Liste der Zustände. Die Zustände entsprechen den Zuständen des LALR(1)-Automaten aus Abb. 3.12, wie folgt:

Abb. 3.16	Abb. 3.12
0	0
1	36
2	47
3	2
4	89
5	5
6	1

Es sind jeweils die punktierten Produktionen, die den Kern eines LALR(1)-Zustandes bilden, angegeben. Allerdings sind die punktierten Produktionen, die aus dem Abschluß unter spontanen Übergängen hervorgehen, weggelassen. Im Zustand 1 (state 1) in Abbildung 3.16 wären z.B. noch die punktierten Produktionen `c -> . 'c' c` und `c -> .'d'` zu ergänzen. Da diese Produktionen eindeutig und ohne großen Aufwand ergänzt werden können ist es wegen der damit gewonnen Übersichtlichkeit durchaus sinnvoll sie wegzulassen. Die Vorausschaubuchstaben werden nur im Konfliktfall angezeigt.

Dem Zustand 0 ist keine Menge von punktierten Produktionen zugeordnet. Wir hatten in der vorangegangen Darstellung durch Einführung eines neuen Startzeichens S' eine uniforme Beschreibung auch für den Zustand 0 benutzt. Es geht, wie man sieht, auch anders.

Dem Standardbeispiel, das auch dem Bisonprogramm als Demonstrationsbeispiel beigegeben ist, das Beispiel eines Taschenrechners mit umgekehrter Polnischer Notation als Eingabesyntax, ist in Abbildung 3.5 in der Form zu sehen, wie wir die Menge der erlaubten Eingabewörter für das Taschenrechnerprogramm bisher notiert hätten. Für das Nichtterminalzeichen N (für *Numeric*) müßten noch weitere Grammatikregeln hinzugenommen werden, die sicherstellen, daß aus N alle erlaubten Eingabezahlen produziert werden können. Das ist hier nicht geschehen, da auch in Bison diese Aufgabe einem vorgeschalteten Programm, einem *lexical analyser*, delegiert wird. Die Grammatik aus Abbildung 3.5 führt zu der in Abbildung 3.18 gezeigten Grammatikdatei, die als Eingabe für Bison dient. Die Nichtterminalzeichen E,I,L,N sind jetzt mit `exp`, `input`, `line` und `Num` bezeichnet. Wobei durch die Großschreibung und die token-Deklaration angezeigt ist, daß `NUM` ein Fall für den *lexical analyser* ist. Das Zeichen für das Zeilenende, bisher \ ist jetzt `\n`. Zum ersten Mal sind in diesem Beispiel Aktionsangaben zu sehen, hinter jeder Produktion in geschweiften Klammern angegeben. Diese Aktionsteile sind C-Programme. Die darin auftretenden Variablen ordnen den Nichtterminalzeichen der Produktion Werte, in diesem Fall numerische Werte zu. Dabei enthält die Variable `$$` den Wert für das Nichtterminalzeichen auf der linken Seite, `$1`, `$2` und so weiter die Werte für das erste , zweite usw auf der rechten Seite auftretende Nichtterminalzeichen. Während des LR-parsings werden ja Regeln von rechts nach links angewandt. Die Werte von `$1` und `$2` sind also bei Regelanwendung bekannt und der der Regel zugeordnete Aktionsteil berechnet daraus den Wert für `$$`. Schließlich enthält die Produktionsregel für eine nichtleere Einagbezeile als Aktionsteil eine Druckanweisung, die das Endergebniss ausdruckt. Die Bisonausgabe ist in den Abbildungen 3.19 bis 3.21 zu sehen.

```
rule 1    s -> c c
rule 2    c -> 'c' c
rule 3    c -> 'd'
Terminals, with rules where they appear
$ (-1)   'c' (99) 2  'd' (100) 3    error (256)
Nonterminals, with rules where they appear
s (5)    on left: 1
c (6)    on left: 2 3, on right: 1 2
state 0
    'c'         shift, and go to state 1
    'd'         shift, and go to state 2
    s           go to state 6
    c           go to state 3
state 1
    c  ->  'c' . c   (rule 2)
    'c'         shift, and go to state 1
    'd'         shift, and go to state 2
    c           go to state 4
state 2
    c  ->  'd' .   (rule 3)
    $default    reduce using rule 3 (c)
state 3
    s  ->  c . c   (rule 1)
    'c'         shift, and go to state 1
    'd'         shift, and go to state 2
    c           go to state 5
state 4
    c  ->  'c' c .   (rule 2)
    $default    reduce using rule 2 (c)
state 5
    s  ->  c c .   (rule 1)
    $default    reduce using rule 1 (s)
state 6         $        go to state 7
state 7         $        go to state 8
state 8         $default accept
```

Abbildung 3.16: Bisonausgabe für G_{CC}

0	$I \to \varepsilon$			1	$I \to IL$		
2	$L \to N$			3	$L \to E\backslash$		
4	$E \to num$			5	$E \to EE+$		
6	$E \to EE-$			7	$E \to EE*$		
8	$E \to EE/$			9	$E \to EE\hat{}$		
10	$E \to E\,n$						

Abbildung 3.17: Die Beispielgrammatik G_{RPCALC}

```
%{
#define YYSTYPE double
#include <math.h>
%}
 %token NUM
 %%
input:  /* empty */
       | input line
;
 line:    '\n'
       | exp '\n' { printf ("\t%.10g\n", $1);}
;
 exp:     NUM          { $$ = $1;       }
       | exp exp '+'   { $$ = $1 + $2;  }
       | exp exp '-'   { $$ = $1 - $2;  }
       | exp exp '*'   { $$ = $1 * $2;  }
       | exp exp '/'   { $$ = $1 / $2;  }
     /* Exponentiation */
       | exp exp '^'   { $$ = pow ($1, $2);}
     /* Unary minus    */
       | exp 'n'       { $$ = -$1; }
;
%%
```

Abbildung 3.18: Bison-Grammatikdatei für G_{RPCALC}

```
Grammar
rule 1    input ->              /* empty */
rule 2    input -> input line
rule 3    line -> '\n'
rule 4    line -> exp '\n'
rule 5    exp -> NUM
rule 6    exp -> exp exp '+'
rule 7    exp -> exp exp '-'
rule 8    exp -> exp exp '*'
rule 9    exp -> exp exp '/'
rule 10   exp -> exp exp '^'
rule 11   exp -> exp 'n'
Terminals, with rules where they appear
$ (-1)
'\n' (10) 3 4
'*' (42) 8
'+' (43) 6
'-' (45) 7
'/' (47) 9
'^' (94) 10
'n' (110) 11
error (256)
NUM (258) 5
Nonterminals, with rules where they appear
input (11)
    on left: 1 2, on right: 2
line (12)
    on left: 3 4, on right: 2
exp (13)
    on left: 5 6 7 8 9 10 11, on right: 4 6 7 8 9 10 11
```

Abbildung 3.19: Kopf der Bison-Ausgabe für G_{RPCALC}

```
state 0
    $default    reduce using rule 1 (input)
    input       go to state 1
state 1
    input  ->  input . line   (rule 2)
    $           go to state 14
    NUM         shift, and go to state 2
    '\n'        shift, and go to state 3
    line        go to state 4
    exp         go to state 5
state 2
    exp  ->  NUM .   (rule 5)
    $default    reduce using rule 5 (exp)
state 3
    line  ->  '\n' .   (rule 3)
    $default    reduce using rule 3 (line)
state 4
    input  ->  input line .   (rule 2)
    $default    reduce using rule 2 (input)
state 5
    line  ->  exp . '\n'   (rule 4)
    exp  ->  exp . exp '+'   (rule 6)
    exp  ->  exp . exp '-'   (rule 7)
    exp  ->  exp . exp '*'   (rule 8)
    exp  ->  exp . exp '/'   (rule 9)
    exp  ->  exp . exp '^'   (rule 10)
    exp  ->  exp . 'n'   (rule 11)
    NUM         shift, and go to state 2
    '\n'        shift, and go to state 6
    'n'         shift, and go to state 7
    exp         go to state 8
```

Abbildung 3.20: Bison-Ausgabe für G_{RPCALC}: LALR(1)-Tabelle

```
state 6
    line  ->  exp '\n' .   (rule 4)
    $default    reduce using rule 4 (line)
state 7
    exp  ->  exp 'n' .   (rule 11)
    $default    reduce using rule 11 (exp)
state 8
    exp  ->  exp . exp '+'   (rule 6)
    exp  ->  exp exp . '+'   (rule 6)
    exp  ->  exp . exp '-'   (rule 7)
    exp  ->  exp exp . '-'   (rule 7)
    exp  ->  exp . exp '*'   (rule 8)
    exp  ->  exp exp . '*'   (rule 8)
    exp  ->  exp . exp '/'   (rule 9)
    exp  ->  exp exp . '/'   (rule 9)
    exp  ->  exp . exp '^'   (rule 10)
    exp  ->  exp exp . '^'   (rule 10)
    exp  ->  exp . 'n'   (rule 11)
    NUM         shift, and go to state 2
    '+'         shift, and go to state 9
    '-'         shift, and go to state 10
    '*'         shift, and go to state 11
    '/'         shift, and go to state 12
    '^'         shift, and go to state 13
    'n'         shift, and go to state 7
    exp         go to state 8
```

Abbildung 3.21: Bison-Ausgabe für G_{RPCALC}: LALR(1)-Tabelle, Forts.1

```
state 9
    exp  ->  exp exp '+' .   (rule 6)
    $default    reduce using rule 6 (exp)
state 10
    exp  ->  exp exp '-' .   (rule 7)
    $default    reduce using rule 7 (exp)
state 11
    exp  ->  exp exp '*' .   (rule 8)
    $default    reduce using rule 8 (exp)
state 12
    exp  ->  exp exp '/' .   (rule 9)
    $default    reduce using rule 9 (exp)
state 13
    exp  ->  exp exp '^' .   (rule 10)
    $default    reduce using rule 10 (exp)
state 14
    $           go to state 15
state 15
    $default    accept
```

Abbildung 3.22: Bison-Ausgabe für G_{RPCALC}: LALR(1)-Tabelle,Forts.2

Kapitel 4

Lösungen

Dieses Kapitel enthält Lösungen zu einigen ausgewählten Übungsaufgaben.

Zu Abschnitt endliche Automaten

Lösung zu Aufgabe 1.1.2:

Sei $A_1 = (S, V_T, s_0, \delta_1, F)$ der gegebene nichtdeterministische endliche Automat mit spontanen Übergängen. Neben $\delta_1(s, a) \subseteq S$ für jeden Buchstaben $a \in V_T$ ist auch $\delta_1(s, \varepsilon)$ definiert und mindestens für ein $s \in S$ von der leeren Menge verschieden. Der Übergang zu einem nicht deterministischen endlichen Automaten ohne spontane Übergänge erfolgt in zwei Stufen. Zur Definition des Automaten, der nach der ersten Transformation erreicht wird brauchen wir die folgende Definition.

Definition 73 *Zu jeder Teilmenge $X \subseteq S$ definieren wir den spontanen Abnschluß von X, $cl(X)$, als die kleinste Teilmenge von Zuständen, die X enthält und unter spontanten Übergängen abgeschlossen ist. Formal:*

$$\begin{array}{lll} cl(X,0) & = & X \\ cl(X,n+1) & = & \{s' \mid \text{ es gibt } s_1 \in cl(X,n) \text{ mit } s' \in \delta_1(s_1,\varepsilon)\} \\ cl(X) & = & \bigcup_{n\leq 0} cl(X,n) \end{array}$$

A_2 stimmt mit A_1 überein, bis auf die Übergangsfunktion δ_2, die wie folgt definiert ist:

$$\begin{array}{lcll}
\delta_2(s_0,\varepsilon) & = & cl(\delta_1(s_0,\varepsilon)) & \\
\delta_2(s_0,a) & = & \delta_1(s_0,a) & \text{für alle } a \in V_T \\
\delta_2(s,\varepsilon) & = & \emptyset & \text{für } s \neq s_0 \\
\delta_2(s,a) & = & cl(\delta_1(s,a)) & \text{für } a \in V_T \text{ und } s \neq s_0
\end{array}$$

Außerdem sei $F_2 = F_1 \cup \{s_0\}$ falls $cl(\{s_0\} \cap F_1 \neq \emptyset$,sonst gelte $F_2 = F_1$.

Man zeigt leicht

$$L(A_1) = L(A_2)$$

Der nicht deterministische endliche Automat A_3 unterscheidet sich von A_2 durch

- die Definition von δ_3

$$\begin{array}{lcll}
\delta_3(s,a) & = & \delta_2(s,a) & \text{für } s \neq s_0,\ a \in V_T \\
\delta_3(s,\varepsilon) & = & \emptyset & \text{für alle } s \in S \\
\delta_3(s_0,a) & = & \bigcup_{s\in\delta_2(s_0,\varepsilon)} \delta_2(s,a) \ \cup \ \delta_2(s_0,a) &
\end{array}$$

- $F_3 = F_2 \cup \{s_0\}$ falls $cl(\{s_0\} \cap F_2 \neq \emptyset$, sonst gelte $F_3 = F_2$.

Zu Abschnitt Abschlußeigenschaften

Lösung zu Aufgabe 1.5.1:

Angenommen L wäre kontextfrei. Die Sprache $M = \{a^k b^n c^m \mid k,n,m \geq 0\}$ ist regulär. Nach Satz 11 wäre dann auch $L \cap M$ regulär. Aber $L \cap M = \{a^n b^n c^n \mid n \geq 0\}$ und wir wissen schon, daß diese Sprache nicht kontextfrei ist.

Zu Abschnitt minimale endliche Automaten

Lösung zu Aufgabe 2.1.2:

1. Für zwei Wörter $w_1, w_2 \in V^*$ gilt $w_1 R_L w_2$ genau dann, wenn $w_1 = w_2$ ist. Denn für $u = w_1^r$ gilt nach Definition von L sicher $w_1 u \in L$, aber $w_2 u$ kann nur dann in L liegen, wenn $w_2 = u^r = w_1^r = w_1$ gilt.

2. Für alle natürlichen Zahlen $n \neq m$ gilt nicht $a^n R_L a^m$, denn es gilt $a^n b^n \in L$ aber $a^m b^n \notin L$. Es wird nicht behauptet, daß $\{[a^n]_{R_L} \mid n \geq 0\}$ alle Kongruenzklassen von R_L sind, aber das Argument genügt um $indez(L) = \infty$ zu zeigen.

3. Man zeigt:

   (a) $\varepsilon R_L w$ gdw $w = a^n$ für $n \geq 0$
   Offensichtlich gilt $\varepsilon R_L a^n$ für alle $n \geq 0$. Aus $\varepsilon R_L w$ folgt zunächst, daß b in w nicht vorkommen kann; den $\varepsilon a = a$ liegt in L aber wa könnte dann nicht in L liegen. Mit demgleichen Argument überlegt man sich, daß c nicht in w vorkommen kann.

   (b) $bR_L w$ gdw $w = a^n b^m$ mit $n \geq 0$, $m > 0$

   (c) $cR_L w$ gdw $w = a^n b^m c^k$ mit $n, m \geq 0$, $k > 0$

   (d) $baR_L w$ gdw $w \notin L$
   Die Sprache L hat die Eigenschaft, daß für jedes $w \in V^*$ gilt:

   $$w \notin L \Rightarrow wu \notin L \text{ für alle } u \in V^*$$

   Andererseits gilt für jedes $w \in L$ auch $wc \in L$. Als ist $V^* \setminus L$ eine Kongruenzklasse bezgl. R_L.

   Daraus folgt, daß $[\varepsilon]$, $[b]$, $[c]$, $[ba]$ alle Kongruenzklassen von R_L sind, also $index(L) = 4$.

Lösung zu Aufgabe 2.1.5:

Zu zeigen ist für je zwei Zustände $q, r \in S$

$$qRr \Rightarrow q \approx_A r$$

Wir zeigen die äquivalente Behauptung für alle $q, r \in S$

$$q \not\approx_A r \Rightarrow \neg qRr$$

Da definitionsgemäß

$$q \approx_A r \Leftrightarrow \text{ für alle } w \in V^* \text{ gilt } (\delta(q,w) \in F \leftrightarrow \delta(r,w) \in F)$$

gilt, können wir die zu zeigende Behauptung umformulieren zu

$$\text{wenn } \neg(\delta(q,w) \in F \leftrightarrow \delta(r,w) \in F) \text{ für ein } w \in V^* \Rightarrow qRr$$

Diese Behauptung läßt sich nun durch Induktion nach Länge von w zweigen. Für $w = \varepsilon$ folgt die Behauptung aus der Voraussetzung, daß R finale von nichtfinalen Zuständen trennt. Gelte die Behauptung für das Wort w. Wir wollen sie für aw zeigen. Voraussetzung ist also

$$\neg(\delta(q,aw) \in F \leftrightarrow \delta(r,aw) \in F)$$

Daraus folgt

$$\neg(\delta(\delta(q,a),w) \in F \leftrightarrow \delta(\delta(r,a),w) \in F)$$

Nach Induktionsvoraussetzung erhalten wir

$$\neg(\delta(q,a)R\delta(r,a))$$

Dann muß aber auch

$$\neg qRr$$

gelten, denn aus qRr würde wegen der Kongruenzeigenschaft von R der Widerspruch $\delta(q,a)R\delta(r,a)$ folgen.

∎

Zu Abschnitt Endliche Automaten mit Ausgabe

Lösung zu Aufgabe 2.2.1:

Wir zeigen, daß für je zwei Wörter $x, y \in V_e^*$ mit $f(x)R_L f(y)$ und $f_x = f_y$ gilt $xR_{f^{-1}(L)}y$. Da R_L nach Voraussetzung nur endlich viele Äquivalenzklassen besitzt und nach Lemma 26 auch nur endlich viele verschiedene Ableitungen von f existieren folgt daraus, daß $R_{f^{-1}(L)}$ nur endlich viele Äquivalenzklassen haben kann. Es gilt

$$\begin{array}{llll} xR_{f^{-1}(L)}y & gdw & xw \in f^{-1}(L) \leftrightarrow yw \in f^{-1}(L) & \text{für alle } w \in V_e^* \\ & gdw & f(xw) \in L \leftrightarrow f(yw) \in L & \\ & gdw & f(x)f_x(w) \in L \leftrightarrow f(y)f_y(w) \in L & \text{Def. der Ableitung} \\ & gdw & f(x)v \in L \leftrightarrow f(y)v \in L & \text{nach Vor. gilt} \\ & & & f_x(w) = f_y(w) = v \\ & gdw & true & \end{array}$$

Zu Abschnitt Definierbarkeit

Lösung zu Aufgabe 2.3.1:

zu 1.

$$\Phi_<(x,y) = \exists X(\neg X(x) \wedge X(y) \wedge \forall u \forall z(X(u) \wedge z = u+1 \rightarrow X(z)))$$

zu 2. Wir geben die Formel für $m = 3$ an. Daraus wird das Muster für den allgemeinen Fall deutlich.

$$\begin{array}{lll} \Phi_{0 \ mod \ 3}(x) & = & \exists X_0 \exists X_1 \exists X_2(\\ & & \bigwedge_{0 \le i < j \le 2} disjunkt_{i,j} \ \wedge \\ & & \forall y(X_0(y) \vee X_1(y) \vee X_2(y)) \ \wedge \\ & & \forall y(\neg \exists z(y = z+1) \rightarrow X_0(y)) \ \wedge \\ & & \forall y \forall z(z = y+1 \rightarrow \\ & & (X_0(y) \rightarrow X_1(z)) \wedge \\ & & (X_1(y) \rightarrow X_2(z)) \wedge \\ & & (X_2(y) \rightarrow X_0(z)))) \end{array}$$

wobei $disjunkt_{i,j} = \neg \exists y(X_i(y) \wedge X_j(y))$.

Lösung zu Aufgabe 2.3.2:

Hinweis: Benutzen Sie den Kompaktheitssatz der Prädikatenlogik erster Stufe.

Lösung zu Aufgabe 2.3.4:

Wir führen den Nachweis für den Spezialfall $p = 2$ und $q = 3$ und überlassen

es dem Leser darin das allgemeine Muster zu erkennen.
Zunächst zerlegen wir $K_{2,3}$ in

$$K_{2,3} = K_{2,3,1} \cap K_{2,3,2}$$

wobei
$K_{2,3,i} = \{w \in V_{2.3}^* \mid u_i$ enthält genau eine $1\}$. Weiter brauchen wir
$B_1 = \{\langle a, 1, c_2, \ldots, c_5\rangle \mid a \in V, c_j \in \{0,1\}\}$
$K_{2,3,1} = \{w \in V_{2.3}^* \mid$ in w kommt genau ein Buchstabe $b \in B_1$ vor$\}$
also läßt sich $K_{2,3,1}$ als regulärer Ausdruck schreiben:
$K_{2,3,1} = V_{-1}^* B_1 V_{-1}^*$ für $V_{-1} = V_{2,3} \setminus B_1$

■

Lösung zu Aufgabe 2.3.5:

Die Projektion $\pi : V_{p,q} \to V^*$ ordnet jedem Buchstaben $\langle a, c_1, \ldots, c_{p+q}\rangle \in V_{p,q}$ den Buchstaben $a \in V$ zu, und ist damit ein Homomorphismus. Die Behauptung folgt aus der Abgeschlossenheit der Klasse der regulären Sprachen unter Homomorphismen.
Die Zahlen p, q müssen so gewählt werden, daß für jeden in Φ vorkommenden Quantor $\exists x_i$ bzw $\exists X_j$ $i \leq p$ bzw $j \leq q$ gilt.

Lösung zu Aufgabe 2.3.6:

Der Beweis wird durch Induktion über die Länge des Wortes u geführt. Der Induktionsanfang $u = \varepsilon$ ist trivial. Sei also $u = au_1$ für ein $a \in V$.

$$\begin{array}{rcll}
\delta(x, uw) & = & \delta(x, au_1w) & \\
 & = & \delta(x, u_1w) \cdot \delta(x, a) & \text{Def. von } \delta \\
 & = & \delta(x, w) \cdot \delta(x, u_1) \cdot \delta(x, a) & \text{Ind.vor.} \\
 & = & \delta(x, w) \cdot \delta(x, au_1) & \text{Assoz. von } \cdot \text{ und Def. von } \delta \\
 & = & \delta(x, w) \cdot \delta(x, u) &
\end{array}$$

Lösung zu Aufgabe 2.3.7:

Zu (1) Sei $(H, \cdot)$ eine Untergruppe von $(G, \cdot)$ a das neutralre Element von H und h ein beliebiges Element aus H. Dann gilt natürlich $h^{k+1} = h^k$. Außerdem gibt es wegen der Gruppeneigenschaft $g \in H$ mit $g \cdot h = a$. Multipliziert man die Gleichung $h^{k+1} = h^k$ auf beiden Seiten mit g^k so erhält man $h = a$.

Also besitzt H genau ein Element.

Lösung zu Aufgabe 2.3.8:

Angenommen es gibt eine Formel $sum(u, v, w) \in \mathcal{L}_1^2$, so daß für alle Strukturen $\mathcal{M}$, deren Universum $U = U_{\mathcal{M}}$ ein Anfangsstück von $I\!N$ ist, und für alle Elemente $a, b, c \in U$ gilt

$$\mathcal{M} \models sum[a, b, c] \text{ gdw } a + b = c.$$

Dann ist die folgende Formel äquivalent zu einer Formel aus $\mathcal{L}_1^2$:

$$\begin{array}{rcl} \Phi & = & \exists x \exists z_1 \exists z_2 (sum(z_1, x, z_2) \wedge z_1 + 1 = x) \wedge \\ & & \forall y (y < x \rightarrow Q_a(y)) \wedge \\ & & \forall y (x \leq y \leq z_2 \rightarrow Q_b(y)) \wedge \\ & & \forall y (y \leq z_2) \end{array}$$

$$L(\Phi) = \{a^n b^n \mid n \geq 0\}$$

Da $\{a^n b^n \mid n \geq 0\}$ keine reguläre Sprache ist erhalten wir damit einen Widerspruch zu Satz 34.

Zu Abschnitt unendliche Wörter

Lösung zu Aufgabe 2.4.1:

Sei $A = (S_A, V, s_0^A, \delta_A, F_A)$ ein endlicher Automat mit $L(A) = K$ und $B = (S_B, V, s_0^B, \delta_B, F_B)$ ein Büchi Automat mit $L^\omega(B) = L$. Wir nehmen ohne Beschränkung der Allgemeinheit an, daß $S_A \cap S_B = \emptyset$ gilt. Der Automat $C = (S_C, V, s_0^C, \delta_C, F_C)$ werde wie folgt konstruiert:

$$\begin{array}{rcll} S_C & = & S_C \cup S_B & \\ s_0^C & = & s_0^A & \\ \delta_C(q, a) & = & \delta_B(q, a) & \text{falls } q \in S_B \\ \delta_C(q, a) & = & \delta_A(q, a) & \text{falls } q \in S_A \text{ und } q \notin F_A \\ \delta_C(q, a) & = & \delta_A(q, a) \cup \delta_B(s_0^B, a) & \text{falls } q \in F_A \\ F_C & = & F_B & \end{array}$$

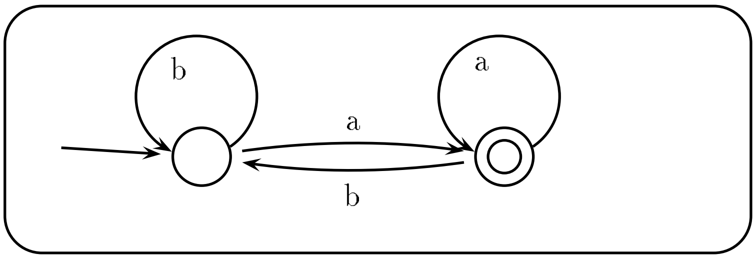
Lösung zu Aufgabe 2.4.2:

Die von dem abgebildeten Automaten akzeptierte ω-Sprache ist

$$L = \{w \in V^\omega \mid \text{für alle } n \text{ mit } w(n) = a \text{ gibt es } m > n \text{ mit } w(m) = b\}$$

Lösung zu Aufgabe 2.4.3:

A_1



$L^\omega(A_1) = b^* a\{b^+ a, a\}^\omega$

Zu Abschnitt Präfixe

Lösung zu Aufgabe 3.2.1:

Nach Lemma 50 ist w ein lebensfähiger Präfix für G, wenn $M_{G,0}$ das Wort w akzeptiert. Da in $M_{G,0}$ aber jeder Zustand ein Finalzustand ist akzeptiert $M_{G,0}$ auch jedes Anfangsstück von w.

Lösung zu Aufgabe 2.4.4

Man setze

$$\begin{array}{lcl} S_M & = & S_A \\ s_0^M & = & s_0^A \\ \delta_M & = & \delta_A \\ \mathcal{F} & = & \{X \subseteq S_M \mid X \cap F \neq \emptyset\} \end{array}$$

$$\begin{aligned} w \in L^\omega(A) &\Leftrightarrow \text{es gibt eine Berechnungsfolge } \sigma \text{ mit } In(\sigma) \cap F \neq \emptyset \\ &\Leftrightarrow \text{es gibt eine Berechnungsfolge } \sigma \text{ mit } In(\sigma) \in \mathcal{F} \\ &\Leftrightarrow w \in L^\omega(M) \end{aligned}$$

Zu Abschnitt LR(k)-Parser

Lösung zu Aufgabe 3.3.1:

Siehe Abbildung 4.1.

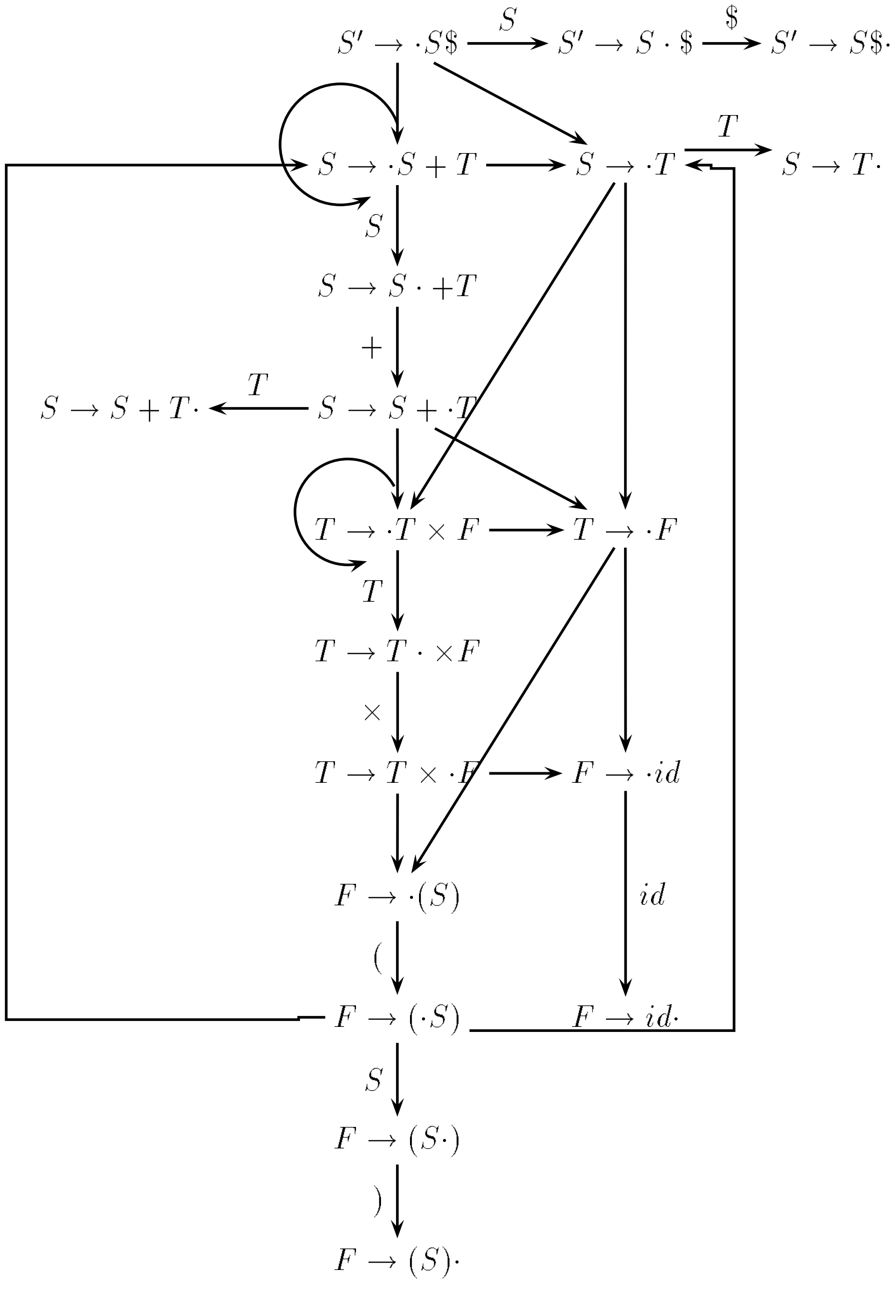


Abbildung 4.1: Der nichtdeterministische LR(0)-Automat für G_{AEXP}

Literaturverzeichnis

[Aho & Ullmann, 1977] Alfred V. Aho & Jeffrey D. Ullmann. *Principles of Compiler Design.* Addison-Weseley, 1977.

[Carroll & Long, 1989] John Carroll & Darrell Long. *Theory of Finite Automata with an introduction to formal languages.* Prentice Hall, 1989.

[Felscher, 1978] Walter Felscher. *Naive Mengen und abstrakte Zahlen I.* B.I.Wissenschaftsverlag, 1978.

[Hopcroft & Ullmann, 1979] J. E. Hopcroft & J. D. Ullmann. *Introduction to Automata Theory.* Addison-Weseley, 1979.

[Immermann, 1988] Neil Immermann. Nondeterministic space is closed under complementation. *SIAM J. Computing*, 17:935–938, 1988.

[McNaughton & Pappert, 1971] R. McNaughton & S. Pappert. *MIT Press.* Counter-free Automata, 1971.

[Moll *et al.*, 1988] R. N. Moll, M. A. Arbib, & A. J. Kfoury. *An Introduction to Formal Language Theory.* Text and Monographs in Computer Science. Springer, 1988.

[Monk, 1976] J. Donald Monk. *Mathematical Logic.* Springer-Verlag, 1976.

[Pin, 1994] Jean-Eric Pin. Logic on words. *Bulletin of the EATCS*, 54:145–165, 1994.

[Ramsey, 1929] F. P. Ramsey. On a problem of formal logic. *Proc. of London Math.Soc.*, 30:264–286, 1929.

[Salomaa, 1978] Arto K. Salomaa. *Formale Sprachen.* Springer, 1978.

[Schützenberger, 1965] M. P. Schützenberger. On finite monoids having only trivial subgroups. *Information and Control*, 8:190–194, 1965.

[Sippu & Soisalon-Soininen, 1988a] Seppo Sippu & Eljas Soisalon-Soininen. *Parsing Theory. Vol. I: Languages and Parsing.* Number 15 in EATCS Monographs on Theoretical Computer Science. Springer, 1988.

[Sippu & Soisalon-Soininen, 1988b] Seppo Sippu & Eljas Soisalon-Soininen. *Parsing Theory. Vol. II: LR(k) and LL(k) Parsing.* Number 20 in EATCS Monographs on Theoretical Computer Science. Springer, 1988.

[Stallman, 1988] Richard Stallman. GNU Emacs Manual, sixth edition, emacs version 18. Technical report, Free Software Foundation, 675 Mass Ave, Cambridge, MA 02139, 1988.

[Straubing, 1994] Howard Straubing. *Finite Automata, Formal Logic, and Circuit Complexity.* Progress in Theoretical Computer Science. Birkhäuser, 1994.

[Szelepcséngi, 1988] R. Szelepcséngi. The method of forced enumerations for nondeterministic automata. *Acta Informatica*, 26:279–284, 1988.

[Thomas, 1990] Wolfgang Thomas. Automata on infinite objects. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science. Vol. B : Formal Models and Semantics*, pages 135–192. Elsevier, Amsterdam, 1990.

[Zima, 1988] Hans Zima. *Compilerbau, I Analyse*, volume 36 of *Reihe Informatik.* B.I.Wissenschaftsverlag, 1988.