

Definition 1.6 (Äquivalenz von Grammatiken). Seien $G_1 = (V_1, \Sigma, \mathcal{P}_1, S_1)$, $G_2 = (V_2, \Sigma, \mathcal{P}_2, S_2)$ zwei Grammatiken mit demselben Terminalalphabet Σ . G_1 und G_2 heißen *äquivalent*, falls $\mathcal{L}(G_1) = \mathcal{L}(G_2)$. ■

Beispiel 1.7 (Grammatik für Identifier). Die Grammatik G_{Idf} aus Abbildung 4 (Seite 8) ist äquivalent zu der Grammatik G'_{Idf} mit folgendem Produktionssystem.

$$\begin{aligned} S' &\rightarrow aA \mid bA \mid \dots \mid zA \\ A &\rightarrow \varepsilon \mid aA \mid bA \mid \dots \mid zA \mid 0A \mid \dots \mid 9A \end{aligned}$$

Dabei ist S' das Startsymbol von G'_{Idf} . ■

Grammatiken und die zugehörigen Sprachklassen werden wie in nachfolgender Definition klassifiziert. Diese Klassifizierung geht auf Chomsky (1956) zurück und wird daher als Chomsky Hierarchie bezeichnet.

Definition 1.8 (Grammatik-/Sprachtypen der Chomsky Hierarchie). Sei $G = (V, \Sigma, \mathcal{P}, S)$ eine Grammatik. G heißt

- *kontextsensitiv* oder *vom Typ 1*, wenn für alle Regeln $x \rightarrow y$ gilt: $|x| \leq |y|$.
- *kontextfrei* oder *vom Typ 2*, wenn alle Regeln die Form $A \rightarrow y$ haben, wobei A eine Variable und y ein beliebiges (eventuell leeres) Wort über $V \cup \Sigma$ ist.
- *regulär* oder *vom Typ 3*, wenn alle Regeln von der Form

$$A \rightarrow \varepsilon \quad \text{oder} \quad A \rightarrow a \quad \text{oder} \quad A \rightarrow aB$$

sind, wobei A, B Variablen (d.h., $A, B \in V$) und $a \in \Sigma$ ein Terminalzeichen sind.

Werden keine Einschränkungen gemacht, so spricht man auch von einer Grammatik *vom Typ 0*. Für Grammatiken, für die das leere Wort ableitbar ist, werden wir in Definition 1.12 einen Sonderfall angeben, der weitere Grammatiken zur Klasse der kontextsensitiven Grammatiken (Typ 1) zulässt. Im Folgenden verwenden wir häufig die Abkürzung CFG für die englische Bezeichnung "contextfree grammar".

Eine Sprache vom Typ i ist eine Sprache L , für die es eine Grammatik vom Typ i mit $\mathcal{L}(G) = L$ gibt. Dabei ist $i \in \{0, 1, 2, 3\}$. Sprachen vom Typ 1 werden kontextsensitiv genannt. Sprachen vom Typ 2 heißen kontextfrei, während Sprachen vom Typ 3 regulär genannt werden. ■

Unmittelbar aus der Definition folgt, dass jede Grammatik vom Typ i ($i = 1, 2, 3$) zugleich eine Grammatik vom Typ 0 ist. Weiter ist klar, dass jede reguläre Grammatik zugleich kontextfrei ist.

Beispiel 1.9 (Grammatiktypen). Wir betrachten Grammatiken mit dem Terminalalphabet $\Sigma = \{a, b, c\}$ und der Variablenmenge $V = \{S, A, B\}$. Folgende Grammatik ist vom Typ 0, aber nicht vom Typ 1, 2 oder 3.

$$S \rightarrow AS \mid ccSb, \quad AS \rightarrow Sbb, \quad cSb \rightarrow c, \quad cS \rightarrow a,$$

Folgende Grammatik ist kontextsensitiv, aber nicht kontextfrei.

$$S \rightarrow aAb, \quad aA \rightarrow aAbc, \quad Abc \rightarrow aacc$$

Ein Beispiel für eine kontextfreie Grammatik, die nicht regulär ist, ist

$$S \rightarrow AB, \quad A \rightarrow aABb \mid \varepsilon \mid bc, \quad B \rightarrow A.$$

Die Grammatik mit den Regeln

$$S \rightarrow b, \quad A \rightarrow \varepsilon \mid a \mid aA \mid aS$$

ist regulär. Die Grammatik G_{Idf} aus Abbildung 4 auf Seite 8 ist zwar nicht regulär; jedoch ist sie äquivalent zu der regulären Grammatik G'_{Idf} aus Beispiel 1.7. Daher ist die Sprache der “Identifizier” regulär. ■

Eine informelle Erklärung für die Begriffe “kontextsensitiv” und “kontextfrei” ist wie folgt. Oftmals haben kontextsensitive Regeln die Form $uAw \rightarrow uvw$. Diese Regel erlaubt es, Variable A im Kontext von u und w durch v zu ersetzen. Man beachte jedoch, dass die Definition kontextsensitiver Grammatiken auch völlig andere Regeln zulässt, die dieser intuitiven Erklärung nicht standhalten. So ist z.B. auch die Regel $aAB \rightarrow cAd$ legitim in Grammatiken vom Typ 1. Mit der Regel $aAB \rightarrow cAd$ darf das Wort aAB durch cAd ersetzt werden, was einer Änderung des Kontexts (statt einer kontextabhängigen Änderung) entspricht. Der Begriff “kontextfrei” erklärt sich daraus, dass in Grammatiken vom Typ 2 die Nichtterminale völlig unabhängig vom Kontext, in dem sie auftauchen, ersetzt werden dürfen.

Beispiel 1.10 (CFG für arithmetische Ausdrücke). Arithmetische Ausdrücke über den natürlichen Zahlen werden aus Konstanten $k \in \mathbb{N}$, Variablen vom Typ Integer und den Basisoperatoren $+$ (Addition) und $*$ (Multiplikation) gebildet. Zur Vereinfachung gehen wir hier von einer festen endlichen Variablenmenge $\{x_1, \dots, x_n\}$ aus. Die Syntax vollständig geklammerter arithmetischer Ausdrücke kann durch eine kontextfreie Grammatik G_{aA} beschrieben werden. Das Alphabet der Terminalzeichen ist

$$\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\} \cup \{x_1, \dots, x_n\} \cup \{+, *, (,)\}.$$

Wir verwenden drei Nichtterminale: das Startsymbol S und Nichtterminale A, B zur Erzeugung der Konstanten, dargestellt als Dezimalzahlen ohne führende Nullen. Die Regeln der Grammatik G_{aA} sind:

$$\begin{aligned} S &\rightarrow A \mid x_1 \mid \dots \mid x_n \mid (S + S) \mid (S * S) \\ A &\rightarrow 0 \mid 1B \mid 2B \mid \dots \mid 9B \\ B &\rightarrow \varepsilon \mid 0B \mid 1B \mid \dots \mid 9B \end{aligned}$$

Beispielsweise ist

$$\begin{aligned} S &\Rightarrow (S + S) \Rightarrow ((S * S) + S) \\ &\Rightarrow ((x_1 * S) + S) \\ &\Rightarrow ((x_1 * x_2) + S) \\ &\Rightarrow ((x_1 * x_2) + x_3) \end{aligned}$$

eine Ableitung der Formel $\alpha = ((x_1 * x_2) + x_3)$ in G_{aA} . Aus G_{aA} sind nur vollständig geklammerte Ausdrücke herleitbar, nicht aber z.B. Ausdrücke wie $x_1 * x_2 + x_3$ (in der Bedeutung von $((x_1 * x_2) + x_3)$) oder $x_1 + x_2 + x_3$ (was aufgrund des Assoziativgesetzes für die Addition eine sinnvolle Schreibweise ist). Für das Entfernen überflüssiger Klammern kann die oben angegebene Grammatik G_{aA} um Regeln vom Typ 0 erweitert werden, etwa $(S + (S + S)) \rightarrow S + S + S$. Statt bereits gesetzte Klammern zu entfernen, können korrekt (eventuell sparsam) geklammerte arithmetische Ausdrücke auch durch eine kontextfreie Grammatik erzeugt werden. Die CFG $G_{aA'}$ verwendet 6 Nichtterminale T, U, A, B, C und das Startsymbol S und folgende Regeln:

$$\begin{array}{lcl} S \rightarrow T & | & S+T \\ T \rightarrow U & | & S*T \\ U \rightarrow (S) & | & A \quad | \quad C \end{array} \quad \begin{array}{lcl} A \rightarrow 0 & | & 1B \quad | \quad 2B \quad | \quad \dots \quad | \quad 9B \\ B \rightarrow \varepsilon & | & 0B \quad | \quad 1B \quad | \quad \dots \quad | \quad 9B \\ C \rightarrow x_1 & | & x_2 \quad | \quad \dots \quad | \quad x_n \end{array}$$

Klammerlose Summen $\alpha_1 + \alpha_2$ sind aus S herleitbar, während klammerlose aus T hergeleitet werden können. Das Nichtterminal U dient zum Setzen von Klammern und der Generierung der Terme: Konstanten (dargestellt als Dezimalzahlen ohne führende Nullen) können mittels der Nichtterminale A und B erzeugt werden. Das Nichtterminal C dient der Erzeugung der Variablen x_1, \dots, x_n . ■

Beispiel 1.11 (Typ 1 Grammatik). Die Sprache $L = \{a^n b^n c^n : n \in \mathbb{N}, n \geq 1\}$ wird durch die kontextsensitive Grammatik G mit folgenden Regeln erzeugt:

$$\begin{array}{lcl} S \rightarrow aSBC & | & aBC \\ CB \rightarrow BC & & \end{array} \quad \begin{array}{lcl} aB \rightarrow ab & & bC \rightarrow bc \\ bB \rightarrow bb & & cC \rightarrow cc \end{array}$$

Dabei sind S, B, C Nichtterminale und S das Startsymbol. Beispielsweise ist

$$\begin{aligned} S &\Rightarrow aSBC &\Rightarrow aaBCBC \\ &\Rightarrow aaBBCC &\Rightarrow aabBCC \\ &\Rightarrow aabbCC &\Rightarrow aabbcC \\ &\Rightarrow aabbcc \end{aligned}$$

eine Herleitung für das Wort $aabbcc = a^2 b^2 c^2$. Wir skizzieren nun, wie der Nachweis, dass $\mathcal{L}(G) = L$ gilt, erbracht werden kann. Zum Beweis der Inklusion " \supseteq " kann durch Induktion nach n gezeigt, dass alle Wörter $a^n b^n c^n$, $n \geq 1$, aus S abgeleitet werden können. (Exemplarisch wurde der Fall $n = 2$ oben betrachtet.) Etwas schwieriger ist der Nachweis der Inklusion " \subseteq ". Zunächst betrachten wir ein Wort $w \in \{S, B, C, a, b, c\}^*$ gebildet aus Nichtterminalen und Terminalen. Ist $\sigma \subseteq \{a, b, c, B, C\}$, so schreiben wir $\#(\sigma, w)$ für die Anzahl an Vorkommen des Symbols σ in w zu bezeichnen. Durch Inspektion aller Regeln stellt man fest, dass für alle $w \in \{S, B, C, a, b, c\}^*$ mit $S \Rightarrow^* w$ folgende beiden Aussagen (1) und (2) gelten:

- (1) $\#(a, w) = \#(b, w) = \#(c, w)$
- (2) Ist $w = uav$, dann ist u von der Form a^n für ein $n \geq 0$. Ist $w = ubv$, so enthält u keines der Symbole c, S oder C . Besteht w nur aus Terminalzeichen, also $w \in \mathcal{L}(G) \subseteq \{a, b, c\}^*$, so hat w die Gestalt $a^n b^m c^k$, wobei $n, m, k \geq 1$.

Aus (1) folgt, dass jedes Wort $w \in \mathcal{L}(G) \subseteq \{a, b, c\}^*$ ebenso viele a 's wie b 's und c 's enthält. Zusammen mit Aussage (2) folgt, dass jedes Wort $w \in \mathcal{L}(G)$ die Gestalt $a^n b^n c^n$ für ein $n \geq 1$ hat. ■

Die ε -Sonderfall für Typ 1 und ε -Freiheit. Intuitiv erwartet man, dass Sprachen vom Typ i zugleich Sprachen vom Typ $i-1$ sind. Für die Fälle $i = 1$ oder $i = 3$ ist diese Aussage klar. Mit der Forderung $|x| \leq |y|$ für alle Produktionen $x \rightarrow y$ einer kontextsensitiven Grammatik, kann das leere Wort niemals in der durch eine kontextsensitive Grammatik definierten Sprache $\mathcal{L}(G)$ liegen. Insofern ist jede kontextfreie Sprache, die ε enthält, nicht kontextsensitiv im Sinne von Definition 1.8 (Seite 13). Im Folgenden lockern wir die Definition kontextsensitiver Grammatiken/Sprachen etwas auf und berücksichtigen den Spezialfall von Sprachen, die das leere Wort enthalten.

Definition 1.12 (ε -Sonderfall für kontextsensitive Grammatiken). Eine Grammatik $G = (V, \Sigma, \mathcal{P}, S)$ mit $\varepsilon \in \mathcal{L}(G)$ heißt *kontextsensitiv* oder *vom Typ 1*, falls gilt:

- (1) $S \rightarrow \varepsilon$
- (2) Für alle Regeln $x \rightarrow y$ in $\mathcal{P} \setminus \{S \rightarrow \varepsilon\}$ ist $|x| \leq |y|$ und S kommt in y nicht vor.

Die zugehörige Sprachen $\mathcal{L}(G)$ wird als Sprache vom Typ 1 oder kontextsensitive Sprache bezeichnet. ■

Z.B. ist die Grammatik mit dem Startsymbol S und den Regeln

$$S \rightarrow \varepsilon \mid A b a, \quad A b \rightarrow b b$$

kontextsensitiv; nicht aber die Grammatik $S \rightarrow \varepsilon \mid S b a$.

Offenbar ist eine Sprache L genau dann kontextsensitiv, wenn die Sprache $L \setminus \{\varepsilon\}$ kontextsensitiv im Sinne von Definitionen 1.8 ist, also wenn es eine Grammatik G gibt, so dass $\mathcal{L}(G) = L \setminus \{\varepsilon\}$ und so dass $|x| \leq |y|$ für jede Regel $x \rightarrow y$ in G .

Wir zeigen nun, dass sich jede kontextfreie Grammatik in eine kontextsensitive Grammatik transformieren lässt. Problematisch sind die in CFG zulässigen ε -Regeln, d.h., Regeln der Form $A \rightarrow \varepsilon$. Im Folgenden zeigen wir, dass diese ε -Regeln in kontextfreien Grammatiken weitgehend entfernt werden können. Hierzu geben wir ein Verfahren an, das eine gegebene CFG G durch eine äquivalente kontextfreie Grammatik ersetzt, welche entweder gar keine ε -Regeln enthält oder gemäß des oben genannten ε -Sonderfalls kontextsensitiv ist. Man spricht dann auch von ε -Freiheit:

Definition 1.13 (ε -Freiheit kontextfreier Grammatiken). Sei $G = (V, \Sigma, \mathcal{P}, S)$ eine CFG. G heißt *ε -frei*, falls gilt:

- (1) Aus $A \rightarrow \varepsilon$ folgt $A = S$.
- (2) Falls $S \rightarrow \varepsilon$, dann gibt es *keine* Regel $A \rightarrow y$, so dass S in y vorkommt. ■

Bis auf die Regel $S \rightarrow \varepsilon$ sind also keine ε -Regeln in ε -freien CFG zugelassen. Falls $S \rightarrow \varepsilon$, dann stellt Bedingung (2) sicher, dass diese nur zur Herleitung des leeren Worts angewandt, aber in keiner anderen Ableitung eingesetzt werden kann. Daher sind ε -freie CFG kontextsensitiv.

Wir zeigen nun, dass es zu jeder CFG eine äquivalente ε -freie CFG gibt. Algorithmus 1 skizziert ein Verfahren, das für gegebene CFG $G = (V, \Sigma, \mathcal{P}, S)$ eine äquivalente ε -freie CFG G' erstellt. Für alle Nichtterminale A mit $A \Rightarrow^* \varepsilon$ und für alle Regeln $B \rightarrow v$, in denen A auf der rechten

Algorithmus 1 CFG $G \rightsquigarrow$ äquivalente ε -freie CFG G'

Berechne $V_\varepsilon = \{A \in V : A \Rightarrow^* \varepsilon\}$ (* Markierungsalgorithmus (s.u.) *)Entferne alle ε -Regeln aus G .**WHILE** es gibt eine Regel $B \rightarrow xAy$ mit $A \in V_\varepsilon$, $|xy| \geq 1$ und $B \not\rightarrow xy$ **DO**wähle eine solche Regel und füge $B \rightarrow xy$ als neue Regel ein.**OD****IF** $S \in V_\varepsilon$ **THEN**füge ein neues Startsymbol S' mit den Regeln $S' \rightarrow \varepsilon$ und $S' \rightarrow S$ ein.**FI**

Seite enthalten ist, etwa $v = xAy$, wird eine neue Regel $B \rightarrow xy$ eingefügt, vorausgesetzt, dass diese Regel noch nicht in der Grammatik enthalten ist und dass x und y nicht beide leer sind. Diese neuen Regeln dienen der Simulation von Herleitungsschritten in G , in denen Regeln zur Herleitung von ε aus A eingesetzt werden. Da in den neuen Regeln $B \rightarrow xy$ weitere Nichtterminale, aus denen sich das leere Wort ε herleiten lässt, auf der rechten Seite (also in x oder y) vorkommen können, ist dieser Vorgang solange zu wiederholen bis auf diese Weise keine neuen Regeln mehr eingefügt werden können. Die ursprünglichen ε -Regeln können dann entfernt werden. Falls $\varepsilon \notin \mathcal{L}(G)$, so enthält die resultierende CFG G' keinerlei ε -Regeln und ist zu G äquivalent. Falls ε in der von G erzeugten Sprache liegt, so benötigen wir in G' ein neues Startsymbol S' und fügen die Regel $S' \rightarrow \varepsilon$ sowie die Kettenregel $S' \rightarrow S$ ein. Die Regel $S' \rightarrow S$ dient als erste Regel in Herleitungen aller nicht-leeren Wörter von $\mathcal{L}(G)$, während die Regel $S' \rightarrow \varepsilon$ benötigt wird, um das leere Wort herzuleiten.

Im ersten Schritt von Algorithmus 1 wird ein algorithmisches Verfahren benötigt, das als Eingabe eine CFG G hat und dessen Aufgabe darin besteht, die Menge V_ε aller Nichtterminale A mit $A \Rightarrow^* \varepsilon$ zu berechnen. Dies lässt sich durch einen *Markierungsalgorithmus* realisieren, der sukzessive alle Nichtterminale, aus denen das leere Wort herleitbar ist, markiert.

1. Markiere alle Nichtterminale A mit $A \rightarrow \varepsilon$.
2. Solange es eine Regel $B \rightarrow A_1 \dots A_n$ gibt, so dass A_1, \dots, A_n markierte Nichtterminale sind und B nicht markiert ist, wähle eine solche Regel und markiere B .
3. Gib alle markierten Nichtterminale aus.

Der Markierungsalgorithmus kann auch für den Test “gilt $\varepsilon \in \mathcal{L}(G)$?” eingesetzt werden. Hierzu ist lediglich zu prüfen, ob S nach Ausführung des Markierungsalgorithmus markiert ist. Genau in diesem Fall gilt $S \Rightarrow^* \varepsilon$.

Hiermit ist gezeigt, dass jede kontextfreie Sprache zugleich kontextsensitiv ist. Wir erhalten folgenden Satz:

Satz 1.14 (Chomsky Hierarchie). *Sprachen vom Typ i sind zugleich Sprachen von Typ $i-1$ ($i = 1, 2, 3$).*

2 Reguläre Sprachen

Reguläre Sprachen wurden in Abschnitt 1 mit Hilfe regulärer Grammatiken definiert. In diesem Kapitel stellen wir äquivalente Formalismen für reguläre Sprachen vor (endliche Automaten und reguläre Ausdrücke) und diskutieren einige wichtige Eigenschaften regulärer Sprachen.

2.1 Endliche Automaten

Ein endlicher Automaten fungiert als *Sprachakzeptor*, dessen Aufgabe darin besteht, das Wortproblem für eine gegebene Sprache L zu lösen, d.h., für gegebenes Eingabewort w zu entscheiden, ob w in L liegt (“der Automat akzeptiert”) oder nicht (“der Automat verwirft”). Im Kontext eines Compilers ist es z.B. die Aufgabe eines Scanners zu entscheiden, ob eine gegebene Zeichenkette w den syntaktischen Regeln eines Identifiers genügt, also ob w zu der formalen Sprache, die durch die syntaktischen Regeln für Identifier gegeben ist, gehört.

Endliche Automaten können als eine sehr eingeschränkte Variante eines Rechnermodells angesehen werden, die mit einem Eingabeband ausgerüstet sind. Die Eingabe eines endlichen Automaten besteht aus einem endlichen Wort, welches zeichenweise auf dem Eingabeband abgelegt ist und auf welches mit Hilfe eines Lesekopfs zugegriffen werden kann. Der Lesekopf kann nur von links nach rechts bewegt werden kann (man spricht deshalb auch von “one-way automata”), um die Eingabezeichen nacheinander zu lesen. Auf das Eingabeband kann nicht geschrieben werden. Das Schema eines endlichen Automaten ist in Abbildung 5 skizziert. Ein endlicher Automat besteht im Wesentlichen nur aus einer endlichen Kontrolle (den Zuständen und der Übergangsfunktion) und einem Eingabeband. Das einzige zur Verfügung stehende Speichermedium sind die Zustände. Da es nur eine feste (von der Eingabe unabhängige) Anzahl von Zuständen gibt, können endliche Automaten zwar in den Zuständen speichern, ob z.B. bereits das Eingabezeichen a gelesen wurde, nicht aber die Anzahl der bereits gelesenen a 's.

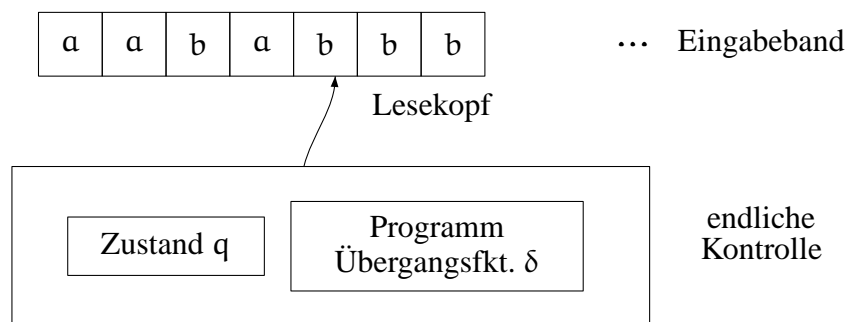


Figure 5: Endlicher Automat (Schema)

2.1.1 Deterministische endliche Automaten

Syntaktisch können endliche Automaten als eine besondere Form von gerichteten Graphen angesehen werden, in denen die Kanten mit Symbolen aus einem Alphabet beschriftet sind und gewisse Zustände als Anfangs- bzw. Endzustände deklariert sind. Wir betrachten zunächst

deterministische endliche Automaten, für die wir die Abkürzung DFA verwenden. Diese steht für die englische Bezeichnung “deterministic finite automaton”. In Abschnitt 2.1.2 werden wir die nichtdeterministische Variante untersuchen.

Definition 2.1 (Deterministischer endlicher Automat (DFA)). Ein DFA ist ein Tupel

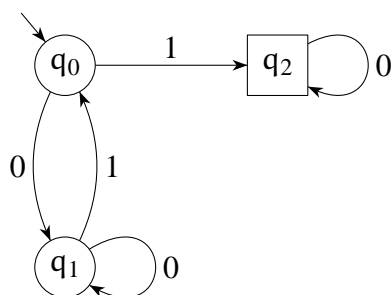
$$\mathcal{M} = (Q, \Sigma, \delta, q_0, F)$$

bestehend aus

- einer endlichen Menge Q von *Zuständen*,
- einem endlichen Alphabet Σ ,
- einem *Anfangszustand* $q_0 \in Q$,
- einer Menge $F \subseteq Q$ von *Endzuständen*
- einer partiellen Funktion $\delta : Q \times \Sigma \rightarrow Q$.

δ wird auch *Übergangsfunktion* genannt. Der Anfangszustand q_0 wird auch als *Startzustand* oder *initialer Zustand* bezeichnet. Die Zustände in F werden auch als *Akzeptanzzustände* oder *finale Zustände* genannt. ■

Wir schreiben auch $q \xrightarrow{a} p$ für $p = \delta(q, a)$ und nennen $q \xrightarrow{a} p$ eine Transition oder einen Übergang. Manchmal sprechen wir auch von einer a -Transitionen oder einem a -Übergang oder auch einer a -Kante. Für die graphische Darstellung eines DFA verwenden wir Kreise für die Zustände $q \in Q \setminus F$ und Quadrate oder Rechtecke für die Endzustände. Der Anfangszustand ist durch einen kleinen Pfeil markiert. Ein Beispiel ist in Abbildung 6 angegeben.



Zustandsraum: $Q = \{q_0, q_1, q_2\}$
 Alphabet: $\Sigma = \{0, 1\}$
 Anfangszustand q_0
 Endzustandsmenge $F = \{q_2\}$

$\delta(q_0, 0) = q_1$	$\delta(q_0, 1) = q_2$
$\delta(q_1, 0) = q_1$	$\delta(q_1, 1) = q_0$
$\delta(q_2, 0) = q_2$	$\delta(q_2, 1) = \perp$

Figure 6: DFA \mathcal{M}

Nun zu dem operationellen Verhalten eines endlichen Automaten. Initial steht das Eingabewort auf dem Eingabeband, der Lesekopf zeigt auf das erste Zeichen des Eingabeworts und der Automat befindet sich in seinem Anfangszustand q_0 . In jedem Schritt findet in Abhängigkeit des aktuellen Zustands $q \in Q$ und des Zeichens a unter dem Lesekopf auf dem Eingabeband ein Wechsel von Zustand q zu Zustand $p = \delta(q, a)$ statt und der Lesekopf wird um eine Position nach rechts verschoben. Ist $\delta(q, a) = \perp$ (undefiniert), so hält die Berechnung des Automaten an. Tritt dieser Fall nicht ein und kann das Eingabewort vollständig “gescannt” werden, dann hält die Berechnung an, nachdem das letzte Zeichen der Eingabe bearbeitet wurde. Eine Berechnung

ist entweder *akzeptierend* (wenn das Eingabewort vollständig gelesen wurde und der Automat einen Endzustand erreicht hat) oder *verwerfend* (sofern kein Endzustand erreicht wurde oder die Eingabe nicht zu Ende gelesen werden konnte).

Wir formalisieren nun das intuitiv erläuterte Akzeptanzverhalten eines DFA. Hierzu führen wir den Begriff eines Laufs für ein Eingabewort in einem DFA ein.

Definition 2.2 (Lauf, akzeptierend, verwerfend). Sei $\mathcal{M} = (Q, \Sigma, \delta, q_0, F)$ ein DFA und $w = a_1 a_2 \dots a_n \in \Sigma^*$ ein Wort der Länge n . Der *Lauf* für w in \mathcal{M} ist diejenige Zustandsfolge $p_0 p_1 \dots p_m$, so dass

- (1) $p_0 = q_0$,
- (2) $q_{i+1} = \delta(q_i, a_{i+1})$ für $i = 1, \dots, m-1$, und
- (3) entweder $m = n$ oder $m < n$ und $\delta(q_m, a_{m+1}) = \perp$.

Falls $m = n$ und $q_n \in F$, dann wird $p_0 p_1 \dots p_n$ *akzeptierender Lauf* für w genannt. In diesem Fall sagen wir, dass w von \mathcal{M} akzeptiert wird. Andernfalls, also wenn entweder $m < n$ oder $m = n$ und $q_n \notin F$, so wird $p_0 p_1 \dots p_m$ *verwerfender Lauf* für w genannt, und w wird von \mathcal{M} verworfen. ■