Technische Universität München
Fakultät für Informatik
Lehrstuhl für Effiziente Algorithmen
Christian Ivicevic

Sommersemester 2015
Studentisches Tutorium
Übungsblatt 2
27. April 2015

Theoretische Informatik

– Automaten 2 (DFA, NFA), Chomsky-Hierarchie –

Reguläre Sprachen und Automaten

In der ersten Woche haben wir das Konzept von (deterministisch endlichen) Automaten kennengelernt um Wörter einer Sprache akzeptieren zu können. Bei der Konstruktion eines DFA haben wir bereits festgestellt, dass man insbesondere bei der Beschreibung der Zustandsübergangsfunktion δ darauf achten, dass diese sowohl total als auch eindeutig ist. In Prosa bedeutet dies, dass aus jedem Zustand genau eine Kante für jedes Symbol aus dem Eingabealphabet ausgeht. Will man sich die Konstruktion solcher Automaten vereinfachen so kann man einen neuen Automatentypen einführen bei dem solche strikte Regeln nicht gelten - in der dritten Woche werden wir insbesondere kennenlernen, dass beide Typen tatsächlich gleichmächtig sind.

Definition (Nichtdeterministischer endlicher Automat)**.** *Ein* nichtdeterministischer endlicher Automat *(engl. nondeterministic finite automaton oder NFA)* $N = (Q, \Sigma, \delta, S, F)$ *ist ein Tupel bestehend aus*

- *einer endlichen* Zustandsmenge Q*,*
- *einem endlichen* Eingabealphabet Σ*,*
- *einer* Zustandsübergangsrelation[1] $\delta : Q \times \Sigma \to \mathcal{P}(Q)$*,*
- *einer Menge an* Startzuständen $S \subset Q$*,*
- *und einer endlichen Menge an* Endzuständen $F \subset Q$*.*

Die von N akzeptierte (erkannte) Sprache *ist*

$$L(N) = \{w \in \Sigma^* \mid \hat{\delta}(S, w) \cap F \neq \emptyset\}$$

wobei die Erweiterung $\hat{\delta} : Q \times \Sigma^* \to Q$ *rekursiv definiert ist durch*

$$\begin{aligned} \hat{\delta}(Q', \varepsilon) &= Q', \\ \hat{\delta}(Q', aw) &= \hat{\delta}\left(\bigcup_{q \in Q'} \delta(q, a), w\right) \qquad \forall Q' \subset Q, a \in \Sigma, w \in \Sigma^*. \end{aligned}$$

[1] Es muss keine Funktion sein, daher weder total noch eindeutig!

Beispiel 1. Während bei DFAs jeder Knoten für jedes Zeichen aus dem Alphabet genau eine ausgehende Kante haben muss, da δ total und eindeutig ist, ist dies bei NFAs nicht der Fall. Hier kann man intuitiv ohne Beachtung jeglicher Regeln arbeiten, was meist zu äquivalenten Automaten führt, die deutlich kompakter sind. Greifen wir Beispiel von letzter Woche[2] erneut auf, so lässt sich die Sprache mit dem NFA aus Abbildung 1 akzeptieren. Vereinfacht gesagt, kann ein NFA raten, wie oft er beispielweise die Schleife bei q_1 nimmt bis er endlich den letzten Übergang nach q_3 macht und somit ein Wort akzeptiert.[3]

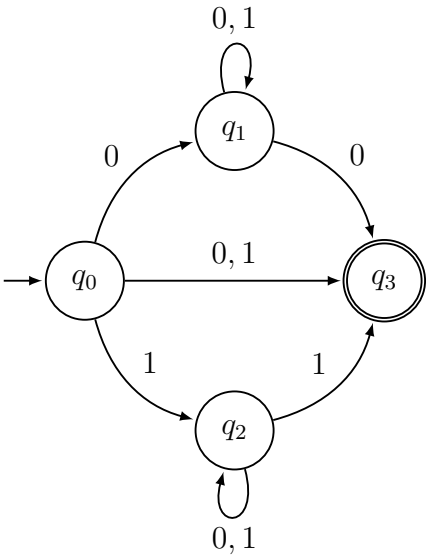

Abbildung 1: Automat zu Beispiel 1

Beispiel 2. Besonders einfach ist es Automaten zu konstruieren, die Wörter $w = uv$ akzeptieren, die auf einem festgelegten Wort v enden müssen. Betrachten wir die Sprache $S = \{w \in \Sigma^* \mid w \text{ endet auf } 110\}$ über dem Alphabet $\Sigma = \{0, 1\}$ so können wir in nur wenigen Sekunden den passenden NFA konstruieren, der in Abbildung 2 zu sehen ist.

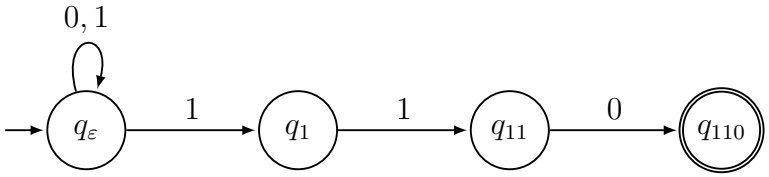

Abbildung 2: NFA N mit $L(N) = S$

Hierbei können wir bei einem Zustand q_s ablesen, dass s die letzten Zeichen waren, die eingelesen wurden. Der Automat bleibt anfangs in q_ε und rät dann richtig, wann er nun in

[2]Dabei ging es um die Sprache, deren Wörter alle gleichzeitig mit 0 beginnen und enden bzw gleichzeitig mit 1 beginnen und enden.

[3]In Wirklichkeit wird bei der Simulation eines NFA jeder nur bisher mögliche Weg, den der Automat durchlaufen hätte können, abgespeichert um abschließend zu prüfen, ob einer der Wege in einen akzeptierenden Zustand geführt hat.

die folgenden Zustände gehen kann, sofern das vorgegebene Wort tatsächlich akzeptiert werden soll. Es ist besonders wichtig zu verstehen, dass wir den Zuständen einen Sinn bzw. eine Semantik geben - der Automat damit aber nichts anfangen kann, da jeder DFA und jeder NFA stets *gedächtnislos* ist, was bedeutet, dass er die Info s aus den Zuständen q_s nicht interpretieren kann wie wir. Möchte man nun einen DFA M mit $L(M) = S$ konstruieren, so kann man mithilfe des *Myhill-Verfahren*[4] aus N einen DFA herleiten oder man ist kreativ und baut M from scratch wie in Abbildung 3 zu sehen ist.

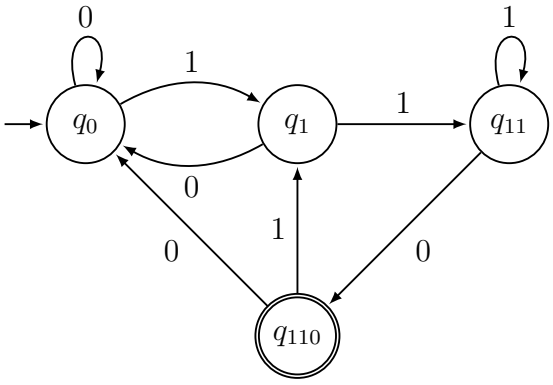

Abbildung 3: DFA M mit $L(M) = S$

Beispiel 3. In diesem Beispiel werden wir kennenlernen, dass jede Sprache, die sich mit endlich vielen Informationen darstellen lässt, regulär ist. Betrachten wir nun das Alphabet $\Sigma = \{0, 1, 2, 3\}$ und die Sprache

$$M = \{w \in \Sigma^* \mid w \text{ besitzt keine führende } 0 \wedge \#w \bmod 3 = 2\}.$$

Aus der Schulzeit ist bekannt, dass ausschließlich die letzte Ziffer ausschlaggebend ist bei der Modulorechnung mit 3. Da wir nur 3 verschiedene Ergebnisse haben und diese Menge endlich ist, können wir zügig einen Automaten konstruieren, der M akzeptiert, auch wenn er gedächtnislos ist. Im ersten Schritt (siehe Abbildung 4) sorgen wir mit einem *Fangzustand* dafür, dass Zahlen, die mit 0 beginnen keinesfalls akzeptiert werden und jeder weitere Eingabe im Fangzustand verbleibt. Da wir drei mögliche Ergebnisse

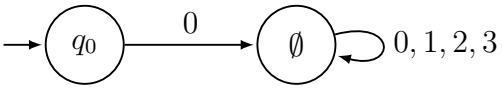

Abbildung 4: Schritt 1 bei der Konstruktion eines Automaten für M

bei der vorgegebenen Modulorechnung haben, benötigen wir einen Zustand pro Ergebnis. Hierzu führen wir die Zustände $0, 1, 2$ ein und halten den bisherigen Automaten fest, der ein einziges Zeichen einlesen kann (siehe Abbildung 5). Nun überlegen wir uns die einzelnen Zustandsübergänge. Lesen wir eine 1 so erhöht sich das Ergebnis jeweils um 1 (mit anschließender mod 3-Rechnung). Somit gelangen wir von 0 nach 1, 1 nach 2 und 2 nach 0. Würden wir eine 2 lesen, so erhöht sich das Ergebnis analog um 2 mit den

[4]Wird in der dritten Übungswoche vorgestellt.

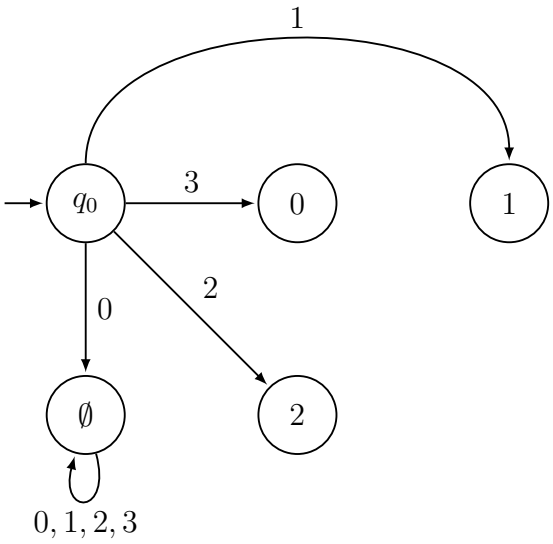

Abbildung 5: Schritt 2 bei der Konstruktion eines Automaten für M

Übergängen 0 nach 2, 1 nach 0 und 2 nach 1. Nun müssen noch die Schlingen für 0 und 3 eingefügt werden, da sich bei diesen Eingaben klarerweise die Ergebnisse nicht ändern. Weiterhin müssen wir den Zustand 2 als Endzustand markieren, da alle Eingaben, die hier landen die gegebene Formel erfüllen. Der fertige Automat ist in Abbildung 6 zu sehen.

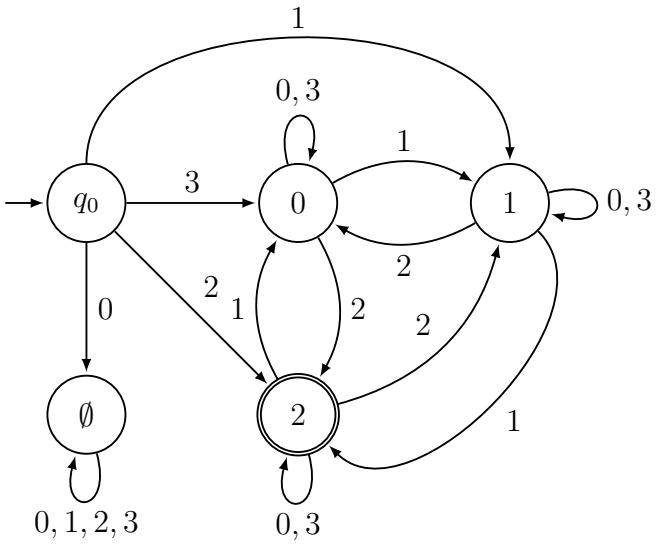

Abbildung 6: Schritt 3 bei der Konstruktion eines Automaten für M

Beispiel 4. Im vorherigen Beispiel haben wir festgestellt, dass wir lediglich drei Informationen mit unserem Automaten darstellen mussten um festzuhalten, ob ein Wort bei der Modulorechnung durch drei als Ergebnis 0, 1 oder 2 liefert. Analog kann man ein Bauchgefühl entwickeln um auch kompliziertere Sprachen genauer analyisieren zu können und begründen zu können, dass diese regulär sind. Betrachten wir die Sprache

$$E = \{a^m b^n \mid (m + n) \bmod 2 = 1\}$$

so kann man feststellen, dass man sich unabhängig von der tatsächlichen Anzahl an a's und b's merken kann ob eine gerade bzw. ungerade Anzahl an a's respektive b's gelesen wurde um dann eine Entscheidung zu treffen, ob ein Wort akzeptiert werden soll. Daher ist es möglich einen Automaten mit genau vier Zuständen zu bauen, der diese Sprache akzeptiert.

Hierarchische Einordnung der bisher behandelten Sprachen

In der Übung haben wir uns bisher mit drei recht kompakten Klassen an Sprachen beschäftigt, die in Abbildung 7 dargestellt sind. Die vorliegende Darstellung werden wir bis zum Ende der Vorlesung noch deutlich erweitern um einen gesamten Überblick über die uns relevanten Probleme zu erhalten.

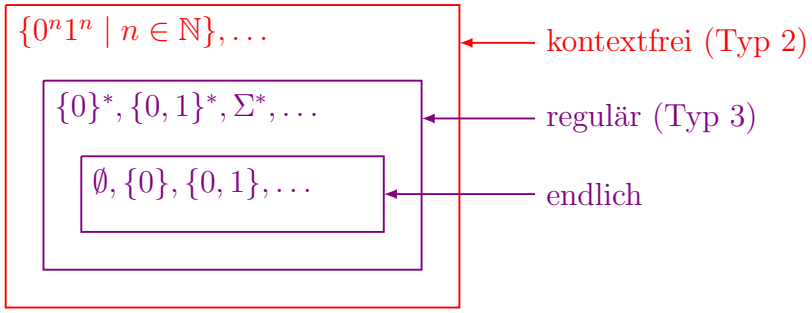

Abbildung 7: Hierarchie der wichtigsten Sprachen und Probleme (Ausschnitt)

Beispiel 5. Wir betrachten eine beliebige endliche Sprache $L = \{w_1, \ldots, w_n\}$ mit $n \in \mathbb{N}$ Wörtern. L ist kontextfrei, da wir eine Grammatik $G = (V, \Sigma, P, S)$ konstruieren können, für die $L(G) = L$ gilt. Hierzu listen wir bei unseren Produktionen lediglich sämtliche Wörter auf und erhalten $P = \{S \to w_1 \mid \ldots \mid w_n\}$.

Beispiel 6. In der letzten Übung wurde verdeutlicht, dass wir im Zusammenhang mit Automaten reguläre Sprachen vorliegen haben - akzeptiert ein DFA oder NFA M eine Sprache $L(M)$, dann ist $L(M)$ regulär. Umgekehrt heißt dies, dass wir beweisen können, dass eine gegebene Sprache regulär ist, sofern wir einen korrekten Automaten konstruieren können, der diese Sprache akzeptiert. Somit können wir beweisen, dass jede endliche Sprache auch regulär ist, ohne auf das Argument aus dem vorherigen Beispiel zurückzugreifen. Hierzu betrachten wir wieder eine Sprache $L = \{u_1 \ldots u_k, v_1 \ldots v_l, w_1 \ldots w_m, \ldots\}$, die unter anderem Wörter u, v, w der Längen k, l, m enthält. An dieser Stelle konstruieren wir einen NFA, der für jedes zu akzeptierende Wort einen Pfad aufweist. Die Skizze[5] eines solchen Automaten ist Abbildung 8 zu entnehmen. Der Zustand q_0' soll den restlichen Teil des Automaten darstellen der sämtliche anderen Wörter in L akzeptiert, die wir aus Platzgründen weggelassen haben. Erneut erkennen wir, wie angenehm der Gebrauch von NFAs ist analog zum Beispiel mit Wörtern, die spezifische Suffixe haben mussten um akzeptiert zu werden.

[5] Als Skizze bezeichnen wir Abbildungen bei denen wir der Übersichtlichkeit halber einige Formalismen weglassen.

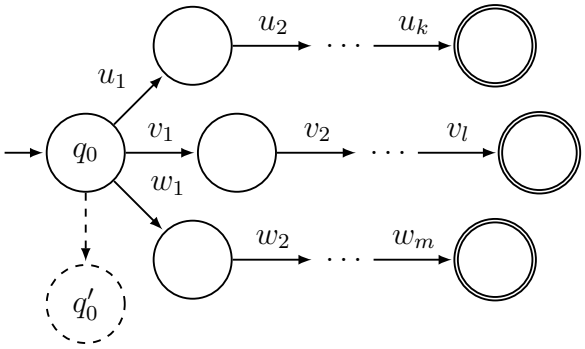

Abbildung 8: Automat zu Beispiel 6

Zuletzt schauen wir uns an, wieso jede reguläre Sprache L auch kontextfrei ist. Wäre L endlich, so folgt die Behauptung aus Beispiel 1. Sofern L nicht endlich ist, existiert ein DFA M mit $L(M) = L$. In der Vorlesung wurde ein Verfahren vorgestellt, welches ermöglicht, aus einem Automat eine Grammatik G abzulesen mit $L(G) = L$.

Satz. *Sei $M = (Q, \Sigma, \delta, q_0, F)$ ein DFA und $G = (Q, \Sigma, P, q_0)$ eine Grammatik mit*

$$P = \{q \to sq' \mid \delta(q, s) = q'\} \cup \{q \to s \mid \delta(q, s) \in F\}.$$

Es gilt $L(G) = L(M)$.

Beispiel 7. Wir betrachten nun den Automaten $M = (\{q_0, q_1, q_2\}, \{a, b\}, \delta, q_0)$, der in Abbildung 9 dargestellt ist. Die zugehörige Grammatik $G = (Q, \Sigma, P, q_0)$ besitzt als Variablen die Zustände Q und das Startsymbol q_0. Um die erste Menge an Produktionen P' zu konstruieren betrachten wir zunächst jeden Bogen, der zwei Zustände $q \in Q$ und $q' \in Q$ mit einem Symbol $s \in \Sigma$ verbindet und fügen die zugehörige Produktion $q \to sq'$ hinzu. In unserem konkreten Beispiel erhalten wir

$$\begin{aligned} P' = \{&q_0 \to aq_1,\ q_0 \to bq_2,\\ &q_1 \to aq_2,\ q_1 \to bq_2,\\ &q_2 \to aq_2,\ q_2 \to bq_2\}. \end{aligned}$$

In einem letzten Schritt müssen wir terminierende Produktionen einfügen. Sofern wir von einem Zustand $q \in Q$ mit einem Symbol $s \in \Sigma$ in einen beliebigen Endzustand, der nicht näher spezifiziert werden muss, kommen, so fügen wir $q \to s$ den bisherigen Produktionen hinzu und erhalten $P = P' \cup \{q_0 \to a\}$.

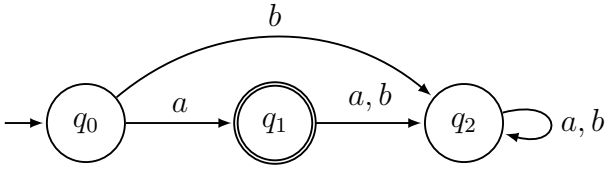

Abbildung 9: Automat zu Beispiel 7