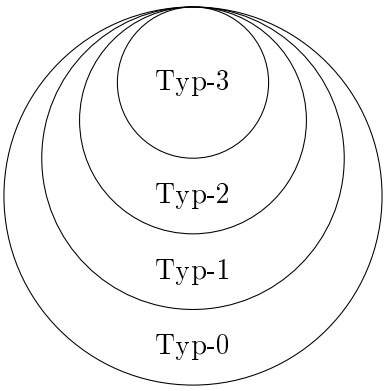
Proseminar TI: Kellerautomaten

Vortrag: 10.11.2015

Von Sebastian Oltmanns und Dorian Wachsmann. Dozent: Wolfgang Mulzer.

1 Motivation



Wir kennen bereits die Chomsky-Hierarchie. Sie klassifiziert formale Grammatiken und damit auch formale Sprachen. Am besten kennen wir bisher die stärkste Klassifizierung dieser Hierarchie und somit die kleinste Art von Grammatiken: die Typ-3-Grammatiken, auch reguläre Grammatiken genannt. Die Menge der Sprachen, die von regulären Grammatiken erzeugt werden, ist die Menge der regulären Sprachen. Die Menge der regulären Sprachen ist genau die Menge der Sprachen, welche von der Menge der Nichtdeterministischen Endlichen Automaten (bzw. Deterministischen Endlichen Automaten) beschrieben wird. Da wir uns viel mit DEA's und NEA's beschäftigt haben, wissen wir auch viel über die Sprachklasse der Typ-3-Grammatiken.

Nun wollen wir etwas über den Rand dieser Sprachklasse hinausblicken. Die Menge der Typ-2-Grammatiken, die kontextfreien Grammatiken, beschreibt die Sprachklasse der kontextfreien Sprachen. Diese Sprachklasse beinhaltet die Sprachklasse der Typ-3-Grammatiken: $Typ_3 \subset Typ_2$. Nun stellt sich die Frage eines Maschinenmodells, mit dem entscheidbar ist, ob eine Sprache die Bedingungen an eine kontextfreie Sprache erfüllt, analog zum Modell der NEA's für die regulären Sprachen. Ein solches Modell kennen wir noch nicht, und so soll diese Ausarbeitung davon handeln, ein Modell zu finden, welches die Menge der kontextfreien Sprachen akzeptiert.

2 Einführung

Wir betrachten zunächst die Regeln einer Typ-3-Grammatik. Eine Grammatik $G = (N, \Sigma, P, S)$ ist eine Typ-3-Grammatik, wenn alle Produktionen $p \in P$ eine der folgenden Formen (innerhalb einer Grammatik nur links- oder rechtsregulär) haben:

$$A \to aB \text{ (rechtsregulär)} \tag{1}$$
$$A \to Ba \text{ (linksregulär)} \tag{2}$$
$$A \to a \tag{3}$$
$$A \to \epsilon, \text{ wobei} \tag{4}$$
$$A, B \in N; a \in \Sigma \tag{5}$$

Schauen wir uns ebenfalls noch die Regeln einer Typ-2-Grammatik an. Eine Grammatik $G = (N, \Sigma, P, S)$ ist eine Typ-2-Grammatik, wenn alle Produktionen $p \in P$ die folgende Form haben: $A \to \gamma$, wobei $A \in N; \gamma \in (\Sigma \cup N)^*$. Wir erkennen hier bereits deutliche Unterschiede zwischen den Grammatiken. Während die Produktionen der Typ-2-Grammatiken lediglich auf der linken Seite eingeschränkt sind und dort nur genau ein Nichtterminalsymbol haben dürfen und die rechte Seite eine beliebige Kombination aus Nichtterminal- und Terminalsymbolen sein kann, sind die Produktionen der Typ-3-Grammatik auch auf der rechten Seite stark eingeschränkt: genau ein Terminal- und maximal ein Nichtterminalsymbol oder das leere Wort dürfen auf der rechten Seite stehen.

Schauen wir uns nun die beispielhafte Sprache $L = \{a^n b^n | n \geq 0\}$ an. Wir versuchen, eine Typ-2-Grammatik dafür zu bauen. Wir erstellen $G_{Typ-2} = (N, \Sigma, P, S)$ mit $N = \{S_0, A, B\}$, $\Sigma = \{a, b\}$, $P = \{\{S_0 \to \epsilon\}, \{S_0 \to$

$AS_0B\}, \{A \to a\}, \{B \to b\}\}$ und $S = \{S_0\}$. Diese Grammatik G_{Typ-2} erfüllt die Bedingungen an eine Typ-2-Grammatik und akzeptiert die Sprache L, da für jedes Wort $w_n = a^n b^n$ einfach n-mal die Produktionsregel $S_0 \to AS_0B$ angewendet wird, danach alle A mit der Regel $A \to a$ durch a (analog B mit Regel $B \to b$ zu b) ersetzt werden und das in der Mitte stehende S_0 mit der Regel $S_0 \to \epsilon$ zu ϵ ersetzt wird. L ist also eine kontextfreie Sprache, das ist durch die kontextfreie Grammatik G_{Typ-2}, welche L erzeugt, begründet.

Versuchen wir nun, eine Typ-3-Grammatik für diese Sprache L zu bauen. Hier merken wir, dass wir dies nicht ohne Probleme schaffen. Das Problem ist, dass wir die Produktion mit der Struktur $S_0 \to AS_0B$ in einer Typ-3-Grammatik nicht beliebig nachbauen können. Übertragen wir dieses Problem auf das Maschinenmodell der Typ-3-Grammatik, merken wir, dass ein NEA/DEA nur endlich viele Zustände haben kann und daher zum Zeitpunkt, an dem er das erste b liest, nicht mehr wissen kann, wieviele a's er zuvor gelesen hat.

Die Sprache L ist also eine kontextfreie Sprache und keine reguläre Sprache. Wir haben bereits eine Typ-2-Grammatik für L gefunden, aber welche Maschine können wir nutzen, um L zu erkennen? Wir merken, dass ein NEA mit einer Art Speicher, in dem er sich merken kann, wieviele a's er gelesen hat, diese Sprache akzeptieren kann. Wir führen also ein neues Maschinenmodell ein, welches durch ein 7-Tupel definiert wird:

$$M := (Z, \Sigma, \Gamma, \delta, z_0, \#, F)$$

, wobei Z die endliche Menge der Zustände ist, Σ das Eingabealphabet, δ die Überführungsfunktion, $z_0 \in Z$ der Startzustand und $F \subseteq Z$ die Menge der akzeptierten Endzustände ist. Soweit ist eigentlich alles von NEA's bekannt. Hinzu kommen noch Γ, das Alphabet des Speichers, und $\# \in \Gamma$, das Zeichen, welches anfangs alleine im Speicher steht. Außerdem ist die Überführungsfunktion δ anders als bei NEA's: $\delta : Z \times (\Sigma \cup \{\epsilon\}) \times \Gamma \to \mathcal{P}_e(Z \times \Gamma^*)$, wobei $\mathcal{P}_e(Z \times \Gamma^*)$ für die endlichen Teilmengen von $\mathcal{P}(Z \times \Gamma^*)$ steht.

3 Kellerautomaten

Dieses Modell M entspricht also grob einem NEA mit einem zusätzlichen Kellerspeicher (auch: Stack, Stapelspeicher). Noch einmal zur Übersicht:

$$\begin{aligned}
M &:= (Z, \Sigma, \Gamma, \delta, z_0, \#, F) \\
Z &: \text{ endliche Menge von Zuständen} \\
\Sigma &: \text{ Eingabealphabet} \\
\Gamma &: \text{ Speicher-/Kelleralphabet} \\
\delta &: \text{ Übergangsfunktion, } \delta : Z \times (\Sigma \cup \{\epsilon\}) \times \Gamma \to \mathcal{P}_e(Z \times \Gamma^*) \\
z_0 &: \text{ Startzustand, } z_0 \in Z \\
\# &: \text{ unterstes Kellersymbol, } \# \in \Gamma \\
F &: \text{ Menge der akzeptierten Endzustände, } F \subseteq Z
\end{aligned}$$

Für Kellerautomaten M beschreiben wir eine „Momentaufnahme“ des Kellerautomaten durch eine Konfiguration $k = (z, a, A)$, wobei $z \in Z$ der aktuelle Zustand, $a \in \Sigma^*$ die noch zu lesende Eingabe und $A \in \Gamma^*$ der aktuelle Speicherinhalt ist. Außerdem definieren wir $\rightsquigarrow_M$ als ein- oder mehrfaches Anwenden der Übergangsfunktion. $(z, a, A) \rightsquigarrow_M (z', a', A')$ heißt, dass die Übergangsfunktion so definiert ist, dass der Kellerautomat M im Zustand z mit a als restlicher Eingabe und A als Speicherinhalt durch ein- oder mehrfaches Anwenden der Übergangsfunktion in den Zustand z' wechseln kann, wobei dann a' die noch zu lesende Eingabe ist und A' der Speicherinhalt.

3.1 Das Kelleralphabet, das Zeichen # und die Endzustände F

Das Modell des Kellerautomaten benötigt offensichtlicherweise eine weitere Eigenschaft im Gegensatz zum NEA/DEA: Das Kelleralphabet Γ. Dieses bestimmt, welche Elemente aus dem Speicher gelesen und hineingeschrieben werden dürfen. Das Zeichen $\# \in \Gamma$ ist das Zeichen, welches anfangs im sonst leeren Speicher steht. Es gibt die Möglichkeit, einen Kellerautomaten $M' := (Z, \Sigma, \Gamma, \delta, z_0, \#)$ zu definieren, bei dem es keine Menge der akzeptierten Endzustände gibt. Hier wird ein Wort als akzeptiert erachtet, wenn der Kellerspeicher nach dem Einlesen des Wortes leer ist. Falls es eine Menge der akzeptierten Endzustände

gibt, wird diese Form des Akzeptierens nicht gewählt sondern wie bei NEA/DEA's gesagt, dass ein Wort akzeptiert ist, wenn sich der Automat nach dem Einlesen des Wortes in einem akzeptierten Endzustand befindet. Dementsprechend definieren wir zwei Arten von Sprachen über Kellerautomaten:

Für einen Kellerautomaten $M = (Z, \Sigma, \Gamma, \delta, z_0, \#, F)$ sei $L(M)$ die durch einen Endzustand akzeptierte Sprache und definiert als:

$$\{w|(z_0, w, \#) \rightsquigarrow_M (f, \epsilon, \gamma), f \in F, \gamma \in \Gamma\}$$

Für M sei $N(M)$ die durch den leeren Keller akzeptierte Sprache und definiert als:

$$\{w|(z_0, w, \#) \rightsquigarrow_M (z, \epsilon, \epsilon), z \in Z\}$$

Betrachtet man nur $N(M)$, schreibt man $M = (Z, \Sigma, \Gamma, \delta, z_0, \#, \emptyset)$ oder $M = (Z, \Sigma, \Gamma, \delta, z_0, \#)$.

Im Folgenden werden wir noch die Idee für den Beweis beschreiben, dass diese beiden Formen des Akzeptieren bei Kellerautomaten äquivalent sind.

3.2 Die Übergangsfunktion δ

Ein Tripel $T = (z, a, A)$ mit $z \in Z, a \in (\Sigma \cup \{\epsilon\}), A \in \Gamma$ bildet eine gültige Eingabe für die Übergangsfunktion δ, und $\delta(T)$ liefert eine Menge von verschieden großen Tupeln. Diese Tupel sind nach dem Muster $(z', B_1, \ldots, B_k)$ aufgebaut, wobei $z' \in Z, B_1, \ldots, B_k \in \Gamma$.

$$(z', B_1, \ldots, B_K) \in \delta(z, a, A)$$

bedeutet also: Wenn sich der Kellerautomat in Zustand z befindet, das Eingabezeichen a liest und das oberste Kellerzeichen A ist, kann er im nächsten Schritt in den Zustand z' wechseln, A aus dem Keller löschen und alle $B_1, \ldots, B_k$ in den Keller schreiben (Reihenfolge beachten: zuerst wird B_k, dann B_{k-1}, usw. in den Speicher geschrieben, sodass B_1 das neue oberste Element ist, LIFO-Prinzip bei Stacks!).

Hierbei ist es möglich, dass A nur gelöscht wird und keine weiteren Zeichen in den Keller geschrieben werden, wenn $k = 0$.

Es ist auch möglich, dass der Keller nicht verändert wird, indem $B_1 = A$ und $k = 1$.

Es kann auch nur ein Element B hinzugefügt werden, ohne dass A entfernt/verändert wird, indem $B_1 = B, B_2 = A$ und $k = 2$.

In der Definition von δ ist zugelassen, dass $a = \epsilon$, sprich, dass kein Symbol vom Wort gelesen wird, und der Automat trotzdem seinen Zustand, den Speicher oder beides ändert. Diese Übergänge werden „spontane Übergänge“ genannt.

3.3 Deterministische und Nichtdeterministische Kellerautmaten

NEA und DEA unterscheiden sich durch die verschiedene Definition der Zustandsübergangsfunktion(/-relation). Hier gilt: bei einem DEA gibt es zu einer Konfiguration genau einen Übergang durch die Übergangsfunktion. Das heißt, wenn ein DEA in einem Zustand ist und ein bestimmtes Zeichen liest, dann liefert die Übergangsfunktion genau eine Folgekonfiguration. Bei einem NEA hingegen kann die Übergangsrelation mehrere mögliche Folgekonfigurationen liefern.

Die gleiche Unterscheidung trifft man bei Kellerautomaten. Wenn sich ein deterministischer Kellerautomat in einer bestimmten Konfiguration befindet, liefert die Übergangsfunktion genau eine Folgekonfiguration, in die der Kellerautomat wechseln wird. Bei nichtdeterministischen Kellerautomaten hingegen kann es für eine Konfiguration mehrere Folgekonfigurationen geben.

Anders als bei NEA und DEA sind diese Arten von Kellerautomaten nicht äquivalent.

4 Äquivalenz der Formen des Akzeptieren

Satz: *Für jeden Kellerautomaten M gibt es einen Kellerautomaten M' mit $L(M') = N(M)$.*

Beweisidee: Wir konstruieren aus dem Automaten $M = (Z, \Sigma, \Gamma, \delta, z_0, \#, \emptyset)$ einen Automaten M' mit $L(M') = N(M)$. Wir konstruieren M' so, dass gilt:

- M' ersetzt im ersten Berechnungsschritt das Symbol # durch $\#X_0$, wobei $X_0 \notin \Gamma$ ein neues Kellersymbol ist
- M' simuliert M
- Sobald X_0 das einzige Symbol im Keller ist, wissen wir, dass der Keller von M leer wäre. Also löscht M' das Symbol X_0 und geht in einen neuen Zustand $z_f \notin Z$, welcher akzeptierter Endzustand ist: $F' = \{z_f\}$.

Mit dieser Simulation geht M' genau dann in den Endzustand, wenn der Keller von M leer wäre. Also ist $L(M') = N(M)$.

Satz: *Für jeden Kellerautomaten M gibt es einen Kellerautomaten M' mit $N(M') = L(M)$.*
Beweisidee: Wir konstruieren aus dem Automaten $M = (Z, \Sigma, \Gamma, \delta, z_0, \#, F)$ einen Automaten M' mit $N(M') = L(M)$. Wir konstruieren M' so, dass gilt:

- M' ersetzt im ersten Berechnungsschritt das Symbol # durch $\#X_0$, wobei $X_0 \notin \Gamma$ ein neues Kellersymbol ist
- M' simuliert M
- Wenn M' während der Simulation von M einen Endzustand von M erreicht, kann sich M' nichtdeterministisch dazu entscheiden, seinen gesamten Keller löschen und damit das Wort akzeptieren.

5 Kontextfreie Sprachen werden genau von ND-Kellerautomaten erkannt

Satz: *Für jede kontextfreie Grammatik $G = (N, \Sigma, P, S)$ gibt es einen Kellerautomaten M, für den gilt: $N(M) = L(G)$.*
Beweis: Wir geben einen Algorithmus zur Erzeugung eines solchen Kellerautomaten an. Wir konstruieren $M = (Z, \Sigma', \Gamma, \delta, z_0, \#)$ mit:

$$\begin{aligned} Z &= \{q_0\} \\ \Sigma' &= \Sigma \\ \Gamma &= \Sigma' \cup N \cup \{\#\} \\ z_0 &= q_0 \end{aligned}$$

Zur Übergangsfunktion δ:

1. Startvariable von G in Keller schreiben.
2. Wiederhole:
   a) Wir lesen ein Nichtterminal n vom Speicher: wähle nichtdeterministisch Produktion von G, sodass n auf der linken Seite der Produktion ist, entferne das gelesene Nichtterminal, ersetze es durch die rechte Seite der Produktion
   b) Wir lesen ein Terminalsymbol t vom Speicher: überprüfe, ob aktuell eingelesenes Eingabezeichen mit t übereinstimmt. Falls ja, entferne t aus dem Speicher.
   c) Wir lesen # vom Speicher: überprüfe, ob das Eingabewort abgearbeitet wurde. Falls ja, entferne # vom Speicher und akzeptiere damit das Wort

Satz: *Für jeden Kellerautomaten $M = (Z, \Sigma', \Gamma, \delta, z_0, \#)$ gibt es eine kontextfreie Grammatik G, sodass $L(G) = N(M)$.*

Beweis: Wir geben einen Algorithmus zur Erzeugung einer solchen Grammatik an. Wir konstruieren $G = (N, \Sigma, P, S)$ mit:

$$N = \{(q, A, p) | q, p \in Z, A \in \Gamma\} \cup \{S\}$$
$$\Sigma = \Sigma'$$

Zu den Produktionen P:

- $\{S \to (q_0, \#, q) | q \in Z\} \subset P$
- $\{(q, A, q_{m+1}) \to (a(q_1, B_1, q_2)(q_2, B_2, q_3) \dots (q_m, B_m, q_{m+1}))$
  $| A, B_1, \dots, B_m \in \Gamma, a \in (\Sigma' \cup \{\epsilon\}), q_1, \dots, q_{m+1} \in Z\}$
  $\subset P$

Wenn man nun $((q, A, p) \Rightarrow_G^* x) \Leftrightarrow ((q, x, A) \rightsquigarrow_M (p, \epsilon, \epsilon))$ zeigt, dann existiert eine Grammatik G zu dem Kellerautomaten M, sodass gilt: $L(G) = N(M)$.

Quellen

J. E. Hopcroft, R. Motwani, J.D. Ullman. Einführung in die Automatentheorie, Formale Sprachen und Komplexität. Pearson Studium, 3. Auflage, 2011.

U. Schöning. Theoretische Informatik – kurz gefasst. Spektrum Akademischer Verlag, 5. Auflage, 2008.

Michael Sipser. Introduction to the Theory of Computation, Thomson Course Technology, 2. Auflage, 2006.

http://home.uni-leipzig.de/heck/automat/webgrammar.pdf (Zugriff 09.11.2015, 15:00 Uhr)

http://www2.cs.uni-paderborn.de/cs/kindler/Lehre/Skripte/Automaten.pdf (Zugriff 09.11.2015, 15:00 Uhr)

http://www.cl.uni-heidelberg.de/kurs/skripte/fg/html/page027.html (Zugriff 09.11.2015, 15:00 Uhr)

http://www.inf.fu-berlin.de/lehre/WS12/Prosem-ThInf/kostic.pdf (Zugriff 09.11.2015, 15:00 Uhr)

https://wwwmath.uni-muenster.de/logik/Personen/Schulz/BETHWIKI/index.php/Kellerautomat (Zugriff 09.11.2015, 15:00 Uhr)

Folgende Artikel der deutschen Wikipedia: Kellerautomat, Chomsky-Hierarchie, Kontextfreie Sprache (Zugriff 09.11.2015, 15:00 Uhr)