Kellerautomaten

In der ersten Vorlesung haben wir den endlichen Automaten kennengelernt. Mit diesem werden wir uns in der zweiten Vorlesung noch etwas eingängiger beschäftigen und bspw. Ansätze zur Konstruktion solcher Automaten (zu einer gegebenen Sprache) kennenlernen. Wir werden dann aber auch merken, dass es Sprachen gibt, für die es uns schwer fällt, einen endlichen Automaten zu konstruieren. Z.B. ist es recht einfach zu $L_1 := \{a^n b^m \mid n, m \in \mathbb{N}\}$ einen DFA zu konstruieren, aber für $L_2 := \{a^n b^m \mid n, m \in \mathbb{N} \text{ und } n \leq m\}$ ist dies sehr viel schwieriger (obwohl die ja gar nicht so viel anders aussieht). Intuitiv ist das Problem, dass ein DFA nur *endlich viele Informationen* speichern kann (nämlich in seinen endlich vielen Zuständen). Um L_1 zu akzeptieren, muss man sich im Grunde genommen nur den "Wechsel" von den as zu den bs merken. Nachdem dieser passiert ist, darf nämlich nur noch das b gelesen werden. Für L_2 müssten wir hingegen einen Zustand haben, in dem wir uns z.B. merken, dass wir 5 mal das a gelesen haben. Nachfolgend müssten wir dann ja mindestens 5 mal das b lesen und dafür brauchen wir die Information, dass es vorher 5 a waren. Ebenso müssten wir uns aber auch merken können, dass es in anderen Fällen 6, 7 oder 8 a waren usw. Dies werden unendlich viele Informationen, die nicht in endlich vielen Zuständen gespeichert werden können.

Obwohl die Sprache L_2 oben recht künstlich wirkt, gibt es viele ähnliche Sprachen, bei denen man die praktische Bedeutung schneller sieht. Bspw. hat man bei der Sprache, die alle korrekt geklammerten Ausdrücke beschreibt (hier soll ()(()) enthalten sein, (() hingegen nicht) ein ähnliches Problem: Man muss irgendwie öffnende Klammern zählen, damit man weiß, wie viele schließende kommen dürfen (zumindest u.a., hier gibt es sogar noch weitere Probleme). Solche Probleme sind bspw. für die Syntaxerkennung in Programmiersprachen wichtig und genau dort werden Automaten unter anderem auch eingesetzt.

Wenn der DFA nun diese Sprachen aber nicht akzeptieren kann, stellt sich die Frage, ob man das Automatenmodell vielleicht so erweitern kann, dass eines entsteht, mit dem man solche Sprachen akzeptieren kann? Mehrere Erweiterungen sind hier denkbar. Man könnte z.B. damit experimentieren, mehrere Leseköpfe zu haben, Zähler für Buchstaben zu haben, allgemeine Zähler zu haben usw. Das Modell, das wir hier einführen wollen, und das eine bekannte Datenstruktur nutzt, ist das eines *Kellerautomaten*. Bei diesem wird der endliche Automat um einen zusätzlichen *Keller* (oder Stack) erweitert.

Abbildung 1 zeigt eine Skizze eines solchen Automaten. Neben dem Eingabeband (oben) gibt es nun noch einen Keller (zur rechten Seite). Auf diesem

Keller ist zu Beginn das spezielle Symbol $\perp$, das sogenannte *Kellerbodensymbol*. Eine Kantenbeschriftung wie bspw. $a, \perp \mid A\perp$ an der Kante von z_0 nach z_0 hat folgende Bedeutung: Befindet sich der Automat im Zustand z_0, liest das a vom Eingabeband und das Symbol $\perp$ vom Keller, dann wechselt er in den Zustand z_0 und der Lesekopf auf dem Eingabeband wird ein Feld weiter nach rechts bewegt (wie beim endlichen Automaten). Auf dem Keller wird nun zusätzlich das $\perp$ gelöscht und durch $A\perp$ ersetzt. Wobei der erste Buchstabe dieses Wortes (also das A) nun ganz oben auf dem Keller liegt. In diesem Fall ist das $\perp$ also eigentlich gar nicht vom Keller gelöscht worden. Beim Übergang von z_0 nach z_1 mit der Kantenbeschriftung $b, A \mid \lambda$ würde folgendes passieren: Wenn der Automat im Zustand z_0 ist, vom Eingabeband ein b gelesen wird und oben auf dem Keller das A liegt und gelesen wird, dann wird in den Zustand z_1 gewechselt, der Lesekopf auf dem Eingabeband wieder ein Feld nach rechts bewegt und auf dem Keller das A gelöscht und λ (also nichts!) geschrieben. Dadurch ist also im Endeffekt das A gelöscht worden. Wer möchte kann sich überlegen, dass der abgebildete Kellerautomat die Sprache $\{a^n b^n \mid n \in \mathbb{N}\}$ akzeptiert: Für jedes a, das vom Eingabeband gelesen wird, wird nämlich zunächst in z_0 ein A auf den Keller geschrieben. Nach lesen von i as befinden sich nun also i As auf dem Keller. Es werden dann durch den Übergang nach z_1 und dann in z_1 die As gelöscht und für jedes genau ein b gelesen. Es wurden also zunächst i as und dann i bs gelesen. Der Übergang von z_1 nach z_2 ist nun nur dann möglich, wenn auf dem Keller wieder nur das $\perp$ ist (also insb. alle A entfernt wurden). Ist das Wort nun zu Ende gelesen, akzeptiert der Automat in z_2 *mit leerem Keller*. Diese Akzeptanzbedingung ist auch neu im Vergleich zum endlichen Automaten. Die Akzeptanz wird so definiert, dass das Wort zu Ende gelesen sein muss *und dann* der Keller leer sein muss (inkl. des Kellerbodensymbols; auch dieses muss entfernt worden sein). Es ist auch möglich die Akzeptanz so zu definieren, dass man mit Endzuständen arbeitet. Ein Wort wird dann akzeptiert, wenn, wie beim DFA, das Wort zu Ende gelesen ist und der Automat dann in einem Endzustand ist. Der Kellerinhalt zu diesem Zeitpunkt ist dann nicht relevant.

An dem Beispiel aus Abbildung 1 kann man gut die erlaubten Übergänge illustrieren. Ein PDA kann ein Symbol vom Eingabeband lesen, muss aber nicht. Beim Übergang von z_1 zu z_2 wird bspw. nichts vom Eingabeband gelesen. Gleich-

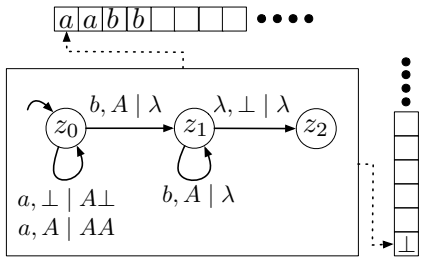

Abbildung 1: Skizze eines Kellerautomaten

zeitig *muss* er stets *genau ein* Symbol vom Keller lesen. Dieses wird gelöscht und durch die Zeichenkette hinter dem | ersetzt. Sei X das zu löschende Symbol und w die Zeichenkette, die geschrieben werden soll. Die Zeichenkette w darf aus beliebige vielen Symbolen bestehen. Besteht w aus keinem Symbol, ist also das leere Wort λ, so wird nur das gelesene Symbol X gelöscht. Besteht w aus einem oder mehreren Symbolen, so wird das gelesene Symbol X gelöscht und durch dieses Zeichenkette ersetzt. Dabei steht der erste Buchstabe von w nun ganz oben auf dem Keller. Der Lesekopf für den Keller befindet sich nun ganz oben auf dem Keller und zeigt auf das oberste Symbol im Keller. Man erkennt an dieser Arbeitsweise auch, dass, wenn die Zeichenkette w mit dem gerade gelöschten Symbol X endet, X faktisch nicht entfernt worden ist.

So wie bisher beschrieben, ist das Modell *nichtdeterministisch* (auch wenn dies bei dem Automaten aus Abbildung 1 nicht auftritt). Nichtdeterminismus ist hier teilweise etwas schwieriger zu erkennen als bei endlichen Automaten. Hat man z.B. eine Kante aus z_1 heraus, die ein a von der Eingabe liest und ein A vom Keller und gleichzeitig eine Kante aus z_1 heraus, die λ von der Eingabe "liest" und ein A vom Keller, so hat man in der Situation, dass man in z_1 ist, im Keller oben das A steht und auf dem Eingabeband gerade a gelesen werden kann, beide Kanten zur Auswahl. Man kann das a von der Eingabe also lesen oder nicht. Ansonsten treten aber auch die "normalen" nichtdeterministischen Fälle auf, wenn es zwei Möglichkeiten gibt, was man aus dem Zustand z_1 heraus bei Lesen eines a von der Eingabe und eines A vom Keller tun kann.

Es ist möglich ein deterministisches Modell einzuführen und dann für beide die Akzeptanzbedingung mit leerem Keller und mit Endzustand zu betrachten. Es stellt sich dann heraus, dass für das nichtdeterministische Modell beide Akzeptanzbedingungen äquivalent sind. Bei dem deterministischem Modell unterscheiden sie sich aber. Außerdem ist, anders als bei endlichen Automaten, das nichtdeterministische Modell echt mächtiger als das deterministische, d.h. es gibt Sprachen, die mit einem nichtdeterministischem Kellerautomaten akzeptiert werden können, nicht aber mit einem deterministischem. Wir gehen hierauf in der Vorlesung noch etwas näher ein.

Das Pumping Lemma

Im ersten Absatz haben wir *intuitiv* argumentiert, warum es wohl keinen DFA für $L_2 = \{a^n b^m \mid n, m \in \mathbb{N} \text{ und } n \leq m\}$ gibt. Dies war aber noch keine hinreichende Begründung. Es könnte ja sein, dass dies doch über eine clevere Konstruktion geht. (Vielleicht muss man sich doch weniger merken, als man zuerst dachte. Die Intuition führt einen da manchmal eben doch in die Irre.) Um stichhaltig zu beweisen, dass eine Sprache nicht regulär ist, gibt es verschiedene Techniken. Eine davon ist die Nutzung des Pumping Lemmas. Das Pumping Lemma macht eine Aussage über alle regulären Sprachen. Es (und insb. seine Anwendungen) machen erfahrungsgemäß Probleme. Wir formulieren es hier einmal mathematisch, skizzieren danach die Anwendung und diskutieren es dann ausführlich in der Vorlesung (und den Übungen). Wichtig zu verinnerlichen ist, dass das Pumping Lemma eine "wenn"-Aussage über reguläre Sprachen macht:

Wenn eine Sprache regulär ist, *dann* gelten bestimmte Dinge. Man kann das Lemma daher *nicht* benutzen, um zu zeigen, dass eine Sprache regulär ist (die Aussage ist nicht, "wenn xyz gilt, dann ist die Sprache regulär"), aber um zu zeigen, dass eine Sprache es *nicht* ist! Zeigt man nämlich dass eine Sprache den "dann"-Teil nicht erfüllt, dann kann sie nicht regulär sein (denn dies würde ja gerade erzwingen, dass der "dann"-Teil erfüllt ist). Wir geben nun das Pumping Lemma an.

Lemma 1. *Sei $L \in REG$ eine reguläre Sprache. Dann gibt es ein $n \in \mathbb{N}$, so dass jedes Wort $z \in L$ mit $|z| \geq n$ in die Form $z = uvw$ zerlegt werden kann, wobei*

1. $|uv| \leq n$
2. $|v| \geq 1$
3. $uv^iw \in L$ *für jedes* $i \in \mathbb{N}$

Äquivalent zur letzten Aussage ist die Aussage $\{u\}\{v\}^*\{w\} \subseteq L$.

Das Lemma sagt also, dass, *wenn* eine Sprache regulär ist, dass *dann* eine natürliche Zahl existiert, so dass Wörter, die länger als diese Zahl sind, in Teilworte u, v und w zerlegt werden können, die dann drei bestimmte Eigenschaften erfüllen.

Um einen Widerspruch zu erzeugen, geht man wie folgt vor: Man nimmt an, eine Sprache wäre regulär. Dann müsste die Aussage des Pumping Lemmas gelten und also eine Zahl n existieren, so dass für Worte ab der Länge n Dinge gelten. Man muss nun *ein* Wort z angeben, das sowohl in L ist und mindestens die Länge n hat und dann *für jede Zerlegung* von z in uvw zeigen, dass mindestens eine der drei Eigenschaften nicht erfüllt ist. Dann ist man "schon" fertig. Man macht dies üblicherweise so, dass man von einer Zerlegung ausgeht, die die ersten beiden Eigenschaften erfüllt und zeigt dann, dass jede solche Zerlegung im Widerspruch zur dritten Eigenschaft ist. Damit hat man auch gezeigt, dass keine Zerlegung alle drei Eigenschaften erfüllt, hat aber aufgrund der Annahme der ersten und zweiten Eigenschaft mehr Eigenschaften, die man beim Beweis benutzen kann (z.B. kann man dann ja benutzen, dass $|v| \geq 1$ gilt).

Wir gehen hierauf in der Vorlesung noch genauer ein, zeigen Anwendungen des Pumping Lemmas und beweisen das Pumping Lemma.

Selbsttest (Lösung auf Seite 6)

1. Wann akzeptiert ein PDA ein Eingabewort mit leerem Keller?
2. Wann akzeptiert ein PDA ein Eingabewort mit Endzustand?
3. Wie viele Symbole kann der PDA vom Keller lesen?
4. Wie viele Symbole kann der PDA auf den Keller schreiben?
5. Was passiert bei einem Zustandsübergang eines PDAs?
6. Wozu kann das Pumping Lemma benutzt werden?

Die mathematische Seite

Wir definieren den nichtdeterministischen Kellerautomaten, die Konfiguration, die Zustandsübergänge und die akzeptierte Sprache.

Definition 2. *Ein* nichtdeterministischer Kellerautomat *(kurz PDA) ist ein* 7*-Tupel* $A = (Z, \Sigma, \Gamma, \delta, Z_{start}, Z_{end}, \bot)$ *mit*

- *Der endlichen Menge von* Zuständen Z*.*
- *Dem endlichen Alphabet* Σ *von* Eingabesymbolen*.*
- *Dem endlichen Alphabet* Γ *von* Kellersymbolen*.*
- *Der* Überführungsfunktion $\delta : Z \times (\Sigma \cup \{\lambda\}) \times \Gamma \to 2^{Z \times \Gamma^*}$*.*
- *Der Menge von* Startzuständen $Z_{start} \subseteq Z$*.*
- *Der Menge der* Endzustände $Z_{end} \subseteq Z$*.*
- *Dem* Kellerbodensymbol $\bot \in \Gamma$*.*

Statt der Überführungsfunktion δ *kann auch mit einer Relation* $K \subseteq Z \times (\Sigma \cup \{\lambda\}) \times \Gamma \times \Gamma^* \times Z$ *gearbeitet werden.*

Ein PDA beginnt im Startzustand z_0 und mit dem Kellerinhalt $\bot$. Ist der Automat im Zustand z, liest vom Eingabeband das a und vom Keller das X (dies muss ganz oben auf dem Keller sein), so kann ein Element $(w, z') \in \delta(z, a, X)$ gewählt werden. Es wird dann in den Zustand z' gewechselt, das X vom Keller gelöscht und durch w ersetzt (wobei das erste Symbol von w nun ganz oben auf dem Keller ist) und der Lesekopf auf dem Eingabeband ein Feld nach rechts bewegt.

Definition 3. *Eine* Konfiguration *eines PDA* A *ist ein Element* $(z, w, v) \in Z \times \Sigma^* \times \Gamma^*$*, mit der Bedeutung, dass* A *im Zustand* z *ist, das* w *noch auf dem Eingabeband zu lesen ist und der aktuelle Kellerinhalt* v *ist.*

Die Überführungsrelation $\vdash \subseteq (Z \times \Sigma^* \times \Gamma^*) \times (Z \times \Sigma^* \times \Gamma^*)$ *ist definiert durch*

$$(z, xu, yw) \vdash (z', u, y'w) \text{ gdw. } (z', y') \in \delta(z, x, y).$$

Die von einem PDA mit leerem Keller akzeptierte Sprache *ist die Menge*

$$L_\lambda(A) := \{w \in \Sigma^* \mid (z_0, w, \bot) \vdash^* (z, \lambda, \lambda)\}$$

und die von einem PDA mit Endzustand akzeptierte Sprache *ist die Menge*

$$L_Z(A) := \{w \in \Sigma^* \mid (z_0, w, \bot) \vdash^* (z_e, \lambda, v), z_e \in Z_{end}, v \in \Gamma^*\}.$$

Man beachte, dass wir bei $L_\lambda(A)$ die Endzustände gar nicht benutzen und dass bei $L_Z(A)$ der Kellerinhalt v am Ende egal ist. Statt $L_\lambda(A)$ schreiben wir auch (insb. im Skript) $N(A)$ und statt $L_Z(A)$ auch $L(A)$.

Lösungen zum Selbsttest auf Seite 4

1. **F:** Wann akzeptiert ein PDA ein Eingabewort mit leerem Keller?

   A: Der PDA muss das Wort zu Ende lesen *und dann* muss der Keller leer sein (der Keller muss komplett leer sein, d.h. auch das Kellerbodensymbol muss entfernt worden sein). Es ist dann egal, in welchem Zustand der Automat ist.

2. **F:** Wann akzeptiert ein PDA ein Eingabewort mit Endzustand?

   A: Wie bei einem DFA muss das Wort zu Ende gelesen werden und genau dann ein Endzustand erreicht sein. Es ist dann egal, was gerade auf dem Keller steht. Wir werden diese Akzeptanzbedingung aber i.A. nicht benutzen. Sie ist für nichtdeterministische PDAs äquivalent zur Akzeptanz mit leerem Keller. Bei deterministischen PDAs unterscheiden sich die beiden Akzeptanzbedingungen.

3. **F:** Wie viele Symbole kann der PDA vom Keller lesen?

   A: Bei jedem Übergang liest der PDA genau ein Symbol vom Keller. Es ist weder möglich, mehr als ein Symbol vom Keller zu lesen, noch gar kein Symbol vom Keller zu lesen. Letzteres heißt insb. auch, dass, wenn der Keller leer ist, kein Zustandsübergang mehr möglich ist. Man kann 'nichts vom Keller lesen' aber dadurch simulieren, dass man das Symbol, das man vom Keller liest, auch wieder zurück schreibt. Dies geht aber nur, wenn noch etwas im Keller ist.

4. **F:** Wie viele Symbole kann der PDA auf den Keller schreiben?

   A: Beliebig viele. Er kann das Symbol, das gerade vom Keller gelesen wurde, löschen. Er kann dieses Symbol löschen und durch ein oder mehrere Symbole ersetzen. Und er kann dieses Symbol stehen lassen und noch ein oder mehrere Symbole hinzufügen.

5. **F:** Was passiert bei einem Zustandsübergang eines PDAs?

   A: Wird der Kantenübergang (z, a, X, w, z') ausgeführt, dann ist der Automat im Zustand z, liest vom Eingabeband das Symbol a und vom Keller das Symbol X (oberstes Symbol auf dem Keller). Wie beim DFA geht nun der Lesekopf auf dem Eingabeband ein Feld weiter nach recht und der Automat wechselt in den Zustand z'. Ferner wird das X vom Keller gelöscht und durch w ersetzt (endet w aber mit X, so ist das X faktisch auf dem Keller geblieben). Das erste Symbol von w ist nun ganz oben auf dem Stack.

6. **F:** Wozu kann das Pumping Lemma benutzt werden?

   A: Dazu zu beweisen, dass eine Sprache *nicht* regulär ist. Insbesondere gilt: Will man zeigen, dass eine Sprache regulär ist, dann hilft einem das Pumping Lemma dabei nicht!