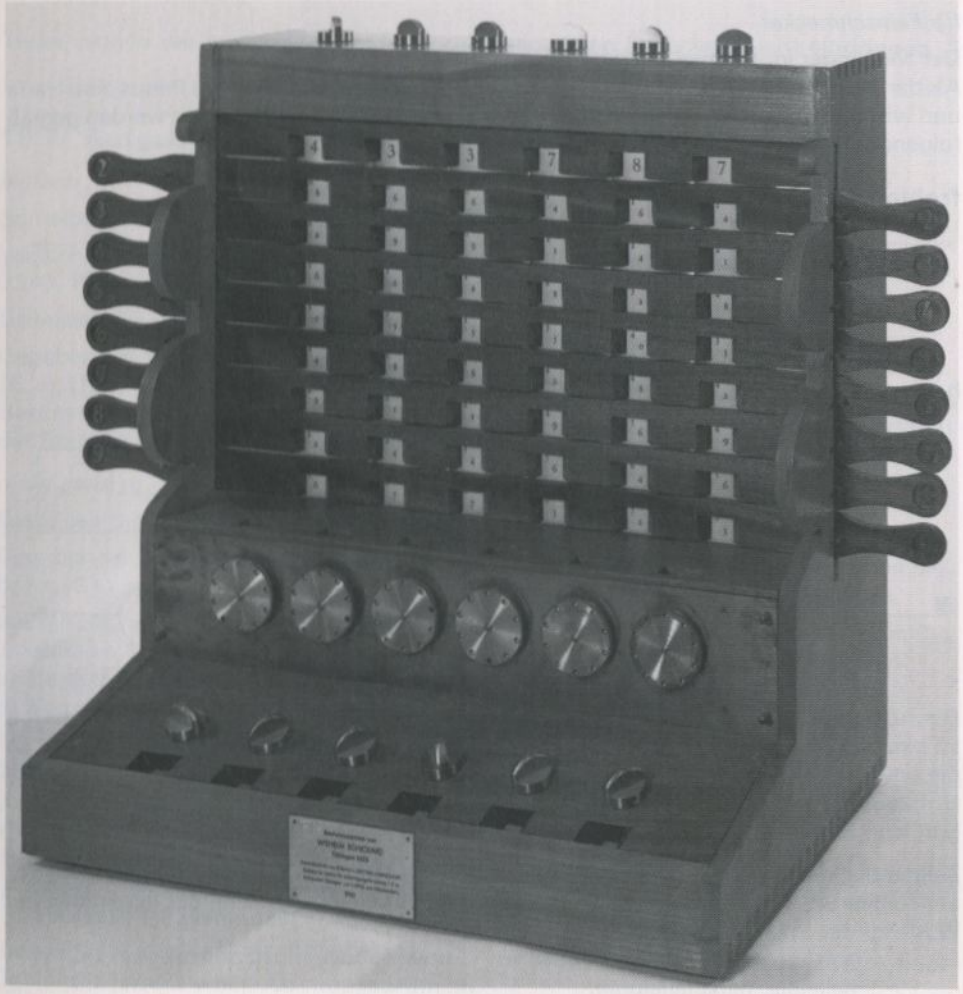
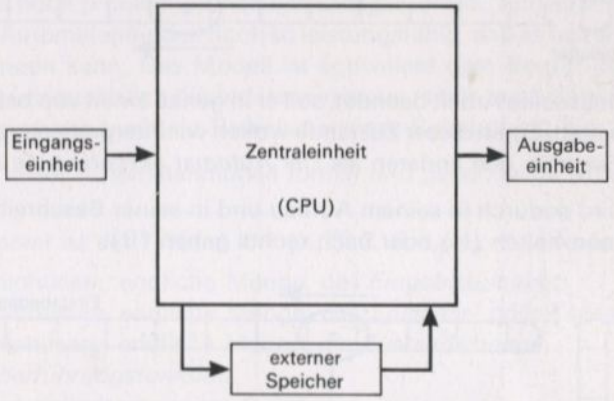
11. Von Formalen Sprachen zu Automatenmodellen



Das Kapitel setzt Kenntnisse aus Kapitel 10 voraus. Es dient als Überleitung von Kapitel 10 zu Kapitel 8 oder 12. Es sollen Automaten konstruiert werden, die das gleiche leisten wie Grammatiken. Dazu machen wir einige Vorüberlegungen über Aufbau und Arbeitsweise von Automaten. Sie werden konkretisiert am Beispiel des Kellerautomaten. Der Übergang zu den Endlichen Automaten (Kapitel 8) und den Turingmaschinen (Kapitel 12) wird angedeutet.

In diesem Kapitel wollen wir Automatenmodelle konstruieren, die feststellen, ob ein Wort zu einer bestimmten Grammatik gehört oder nicht. Wir orientieren uns dabei an einem einfachen Computermodell:

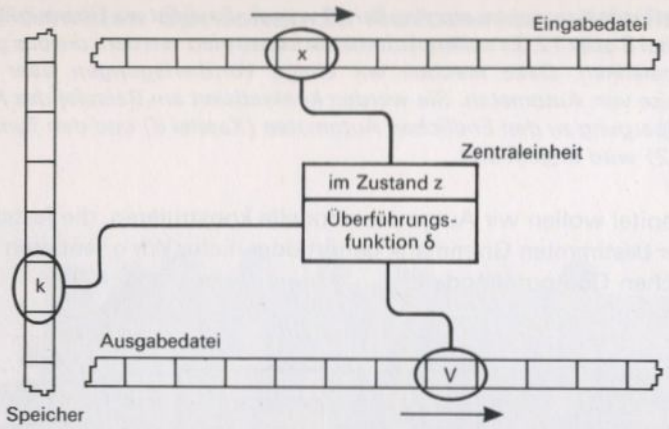


Die *Eingabe* wird wie etwa bei der Programmiersprache Pascal als *sequentielle Datei* verstanden, d. h. sie kann nur zeichenweise von vorne nach hinten gelesen werden. Gegebenenfalls muß ein Abschluß entsprechend dem *eof* zur Verfügung stehen.

Da die Regeln von Grammatiken keinen Einschränkungen unterliegen, scheint es in diesem Stadium der Überlegungen nicht sinnvoll, die Arbeit des Modells nach erfolgter Eingabe abzubrechen. Auch könnte es sich als sinnvoll erweisen, die Eingabe mehrfach zu lesen.

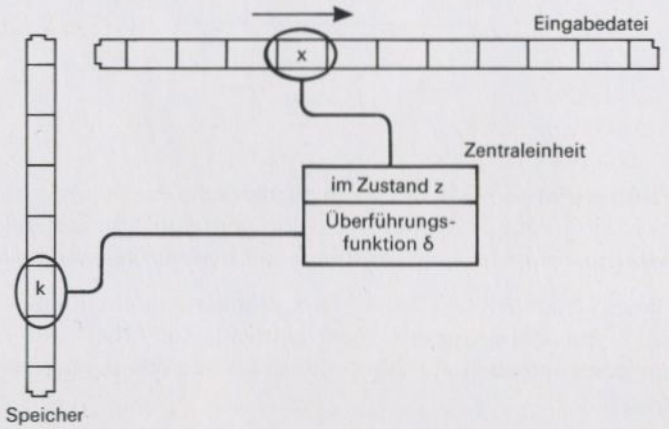
Die Zentraleinheit wird als endliche *Zustandsmenge* mit einer gegebenen *Überführungsfunktion* festgelegt. Die Überführungsfunktion bestimmt die Arbeitsweise des Automaten, indem sie jedem Eingabezeichen, Speicherzeichen und Zustand einen neuen Zustand und neue Worte in Speicher und Ausgabe zuordnet. Außerdem legt sie die Lage des Lese-/Schreibkopfes in Eingabe, Speicher und Ausgabe fest. Sie muß *in sich* die Regeln der Grammatik enthalten. Als *Ausgabe* bietet sich wieder eine *sequentielle Datei* an (Abb. S. 180). Über die Organisation des Speichers ist damit noch nichts ausgesagt.

Der Automat soll eine Entscheidung treffen und uns diese am Ende durch besondere Zustände anzeigen. Wir können uns deshalb eine Ausgabe sparen. Wir führen zwei spezielle Zustände ein.



Wenn der Automat seine Arbeit beendet, soll er in genau einen von beiden gehen und aufhören zu arbeiten. Einen dieser Zustände wollen wir interpretieren als *Der Automat akzeptiert die Eingabe*, den anderen als *Der Automat akzeptiert die Eingabe nicht*.

Der Automat wird dadurch in seinem Aufbau und in seiner Beschreibung einfacher, der Lesekopf kann halten (H) oder nach rechts gehen (R):



Die Arbeitsweise und die Organisation des Speichers sind immer noch völlig offen. Weil es für die Praxis bedeutsam ist, wollen wir uns in einem ersten Versuch an die LR (1)-Analyse[1] (siehe Kapitel 10) anlehnen.

Die Arbeitsweise könnte dann folgendermaßen aussehen:

Der Automat liest Zeichen aus der Eingabe und kopiert sie in den Speicher. Sobald er feststellt, daß eine Regel anwendbar ist, wendet er sie an. Das heißt, er löscht ein oder mehrere Zeichen im Speicher und schreibt andere dafür hin. Dies wird gegebenenfalls wiederholt.

[1] *L* steht für Lesen des Wortes von *L*inks nach rechts, *R* für *R*echtsanalyse

Ist die ganze Eingabe bearbeitet und steht im Speicher nur noch das Startzeichen der Grammatik, dann liegt ein Wort der Sprache vor, und der Automat geht in den Endzustand *a*(*kzeptiert*). Kann der Automat irgendwann keine Regel mehr anwenden, geht er in den Endzustand *n*(*icht akzeptiert*). Er braucht dann die Eingabe nicht ganz zu bearbeiten.

Schauen wir uns den Speicher an: Durch die besondere Art der LR (1)-Analyse muß der Automat nur die letzten oder nur das letzte gespeicherte Zeichen wissen. Dieses muß er löschen und zum vorhergehenden zurückgehen können. Außerdem muß er weitere Zeichen anfügen können. Ein Speicher dieser Art heißt *Keller*, *Stapel* oder *stack*.[2]
Obwohl wir nur von einer kleinen Teilmenge der kontextfreien Sprachen ausgegangen sind, die zudem noch problemlos, nämlich sackgassenfrei, zu bearbeiten sind, ist das so entworfene Automatenmodell doch so leistungsfähig, daß es fast alle kontextfreien Sprachen erkennen kann. Das Modell ist äquivalent dem Begriff der kontextfreien Sprache, wenn wir zusätzlich *Nichtdeterminismus* (siehe auch Kapitel 16) zulassen. Dies würde genau wie auch ein Beweis für diese Behauptung hier zu weit führen.

Wir definieren unser Automatenmodell formal und geben seine Arbeitsweise an:

Ein *Kellerautomat* ist ein 7-Tupel $K = (X, S, Z, \delta, s_0, z_0, Z_E)$, wobei gilt:
- X ist eine nichtleere, endliche Menge, das *Eingabealphabet*,
- S ist eine nichtleere, endliche Menge, das *Speicher-* oder *Kelleralphabet*,
- Z ist eine nichtleere, endliche Menge, die *Zustandsmenge*,
- δ ist die *Überführungsfunktion*,
- $s_0 \in S$ ist das *Kellerleerzeichen*,
- $z_0 \in Z$ ist der *Anfangszustand*,
- $Z_E \subseteq Z$ ist die *Endzustandsmenge*.

Die Überführungsfunktion δ sieht kompliziert aus, weil sie die komplexe Arbeitsweise des Automaten wiedergeben muß.

Wir wollen sie nur verbal erläutern:

Der Automat liest ein Zeichen der Eingabe oder das Eingabeende eof. Weiter liest er ein Zeichen des Speichers. In Abhängigkeit dieser Zeichen und seines aktuellen Zustands reagiert er. Für Endzustände braucht δ nicht definiert zu sein, da der Automat in einem Endzustand seine Arbeit einstellt. In der Eingabedatei kann er stehenbleiben (H), oder er geht zum nächsten Zeichen (R). Steht er auf dem Eingabeende, muß er stehenbleiben. Das Zeichen im Speicher wird gelöscht und ein neues Wort hingeschrieben. Der Lese-/Schreibkopf des Speichers steht danach auf dem letzten Zeichen dieses Wortes. Ist das neue Wort das leere Wort, so bedeutet das, daß das letzte Zeichen des Speichers gelöscht wurde, und der Lese-/Schreibkopf steht auf dem letzten Zeichen vor dem gelöschten. Ist der Speicher leer, d.h. ist das gelöschte Zeichen das Kellerleerzeichen, so muß dieses wieder geschrieben werden. Außerdem kann der Automat in einen anderen Zustand gehen.

[2] stack (engl.) = Stapel

Die Sprache a^nb^n

Wir wollen uns nun an einem Beispiel die Arbeitsweise des Automaten ansehen. Wir konstruieren einen Automaten, der gerade die Sprache $\{a^nb^n \mid n \in \mathbb{N}\}$ akzeptiert. Die Idee dazu ist einfach: Zuerst werden alle a's gelesen und in den Speicher kopiert. Bei jedem folgenden b wird dann ein a aus dem Speicher gelöscht. Wird das letzte a genau dann gelöscht, wenn das letzte b gelesen wurde, so hat das Eingabewort die gewünschte Gestalt und wird akzeptiert. In allen anderen Fällen wird es abgelehnt.

Wir geben die Überführungsfunktion durch drei Tabellen an. Dies reicht hier aus, da der Automat außer den beiden Endzuständen nur einen Zustand hat.

1. Bewegung auf der Eingabedatei

x \ S	s_0	a
a	R	R
b	R	R
eof	H	H

2. Neue Worte im Keller

x \ S	s_0	a
a	s_0a	aa
b	s_0	ϵ
eof	s_0	ϵ

3. Neuer Zustand

x \ S	s_0	a
a	z_0	z_0
b	n	z_0
eof	a	n

Obwohl wir nur einen echten Zustand haben, wird der Automat recht umfangreich. Wir möchten auf ein Problem dieses Automatenmodells hinweisen: Ersetzt man in der ersten Tabelle unseres Beispiels das R durch H, so gerät der Automat in eine Endlosschleife. Hier ist das einfach zu erkennen.

In Kapitel 15 greifen wir das Problem von Endlosschleifen nochmals auf, allerdings in vertrauter Umgebung, nämlich bei Programmiersprachen.

Sind alle Regeln der Grammatik sehr einfach, genauer linkslinear oder abschließend, so kann der Automat vereinfacht werden. Diese Regeln sind so einfach, daß wir auf den Speicher sogar verzichten können. Wir haben dann das Modell des *Endlichen Automaten* vor uns, das in Kapitel 8 ausführlich besprochen wird.

Sind einzelne Regeln der Grammatik komplexer, genauer kontextsensitiv oder sogar vom Typ 0, so reicht die einfache Speicherorganisation wie besprochen nicht aus. Wir müssen dann zulassen, daß unter dem Lese-/Schreibkopf einzelne Zeichen gelöscht oder auch eingefügt werden können. Außerdem muß sich der Lese-/Schreibkopf im Speicher bewegen können. Als Struktur des Speichers bietet sich dann eine *doppelt verkettete, lineare Liste* an. Üblicherweise faßt man dann Eingabe und Speicher zu einem Band zusammen und vereinfacht noch etwas die Arbeitsweise. Wir haben dann das Modell der *Turingmaschine* vor uns, das in Kapitel 12 vorgestellt wird.

Die beiden zuletzt angesprochenen Modelle des Endlichen Automaten und der Turingmaschine sind gegenüber dem Kellerautomaten wesentlich einfacher in ihrer Arbeitsweise zu beschreiben und zu verstehen. Wir verzichten deshalb hier auf eine Vertiefung des Modells des Kellerautomaten und verweisen dazu auf die Fachliteratur.

Nach diesem Ausblick müssen Sie sich entscheiden, ob Sie weiter mit der Theorie arbeiten wollen, etwa mit den Kapiteln 12 oder 15. Sie können auch zu Assembler (Kapitel 1 bzw. 2) oder zur Technik (Kapitel 6) wechseln.