LaTeX Einführung

Sandro Speth, Marcial Gaißert
Universität Stuttgart

Sommersemester 2017

1 Einleitung

In dieser PDF werden grundlegende LaTeX-Beispiele gezeigt, die es euch erleichtern sollten die Abgaben in verschiedenen Modulen zu tätigen. Um LaTeX kompilieren zu können, benötigt ihr einen LaTeX-Compiler, der euch dann PDFs erstellen kann. Außerdem könnt ihr auch unterschiedliche Pakete einbinden (z.B.: um Links anklickbar zu machen, Bäume zeichnen zu können, ...) oder Templates angeben, die verwendet werden sollen. Zusätzlich gibt es viele Hilfsprogramme, um z.B. eine Literaturliste erstellen zu lassen. Das kann man sich zwar von Hand zusammensuchen, es empfiehlt sich aber hierfür ein LaTeX-Paketverwaltungssystem, wie MiKTeX[1] (für Windows) oder TeX Live[2], zu verwenden.

Editoren existieren für LaTeX viele verschiedene. Unter Windows empfiehlt sich das TeXnicCenter[3] in Kombination mit SumatraPDF[4], um PDFs gleichzeitig bearbeiten und betrachten zu können und so quasi live zu sehen, wo sich etwas ändert. Wie man das einrichtet steht u.a. hier[5].

Unter Linux kann man je nach eigener Präferenz einen guten Texteditor (Gedit, Emacs, Vim, ...) in Zusammenhang mit einem PDF-Viewer oder eine der verschiedenen LaTeX-IDEs verwenden:

Texstudio Texstudio ist eine Fork von Texmaker mit zusätzlichen Features wie Grammatiküberprüfung oder automatischer mehrfacher Kompilierung. Mit Vorschau.

Gummi GTK+-basiert, mit kontinuierlicher Vorschau (wird bei Änderungen im Dokument automatisch aktualisiert).

Kile KDE-basiert, ohne eigenen PDF-Betrachter.

2 Erste Schritte

In diesem Kapitel wollen wir uns anschauen, was man grundsätzlich in einem LaTeX-Dokument benötigt. Da wir mit LaTeX sozusagen unsere PDF programmieren, benötigen wir wie auch in Java eine Umgebung, in der wir schreiben, die sogenannte Präambel.

[1] http://miktex.org/
[2] http://www.tug.org/texlive/
[3] http://www.texniccenter.org/
[4] http://sumatrapdfreader.org/free-pdf-reader-de.html
[5] http://goo.gl/buqyCQ

2.1 Präambel

Zu Beginn eines jeden LaTeX-Dokuments muss die Dokumentenklasse deklariert werden. Hierfür existiert der Befehl `\documentclass[optionale Parameter]{Dokumentenklasse}`. Unter den optionalen Parametern können beispielsweise die Schriftgröße und Seitengröße (z.B. A4) festgelegt werden. Bei der Dokumentenklasse gibt es einige Standardklassen. Für die Abgaben in DSA eignen sich jedoch besonders die Dokumentenklassen `article` und `report`.

Anschließend werden, ähnlich zu den Java Imports, die verwendeten Pakete eingefügt. Der Befehl hierfür lautet `\usepackage[optionen]{Paketname}`. Viele Funktionen, wie z.B. mathematische Formeln oder eine farbiges PDF Ausgabe erfordern entsprechende Pakete.
Um eine deutsche Silbentrennung, Rechtschreibung und andere wichtige Bestandteile der deutschen Sprache zu erhalten, müssen wir das Paket `\usepackage[ngerman]{babel}` einbinden.
Anschließend sollten wir das Ausgabefont festlegen, das LaTeXverwenden soll. Für unsere Zwecke reicht hierfür das Paket `\usepackage[T1]{fontenc}` völlig aus.
Zusätzlich zum Ausgabefont muss noch die Eingabekodierung festgelegt werden. Dafür bietet sich das Paket `\usepackage[utf8]{inputenc}` an. Selbstverständlich gibt es noch viele weitere Pakete, die ihr verwenden könnt. Im Laufe dieser LaTeX-Einführung wird immer erwähnt, welche Pakete für die jeweiligen Funktionalitäten eingebunden werden müssen.

Abschließend müssen wir noch festlegen, an welcher Stelle das eigentliche Dokument beginnen und enden soll. Den Start des Dokuments geben wir mit dem Befehl `\begin{document}` an. Entsprechend dazu beenden wir das Dokument mit dem Befehl `\end{document}`. Zwischen den beiden Befehlen können wir nun unser eigentliches Dokument schreiben. In Listing 1 ist eine Beispiel Präambel zu sehen.

Listing 1: Präambel eines LaTeX-Dokuments

```
1 \documentclass[12pt]{article}
2 \usepackage[ngerman]{babel}
3 \usepackage[latin1]{inputenc}
4 \usepackage[T1]{fontenc}
5 \begin{document}
6 % Text
7 \end{document}
```

2.2 Strukturierung

Um eine bessere Strukturierung im Dokument zu erhalten, unterteilt man dieses häufig in Kapitel (Abschnitte) und Unterkapitel (Unterabschnitte). In LaTeX-Dokumenten ist dies ebenfalls möglich. Hier lassen sich Abschnitte mit dem Befehl `\section{Überschrift}` erzeugen. Möchte man in den Ebenen darunter Unterabschnitte erzeugen, so lässt sich dies mit dem Befehl `\subsection{Überschrift}` bzw. `\subsubsection{Überschrift}` bewerkstelligen. Zusätzlich zu den Abschnitten existieren in LaTeX auch noch Absätze, die mit dem Befehl `\paragraph{Bezeichnung}` erzeugt werden können. Auch für Absätze kann man äquivalent zu den Abschnitten in den Ebenen tiefer gehen. Fügt man den Befehlen ein Stern in den Namen hinzu, so erhält man eine Gliederung ohne Nummerierung. Ein Beispiel hierfür ist `\section*{Überschrift}`.

2.3 Aufzählung und Nummerierung

Oftmals wollen wir Elemente einfach auflisten. Dies ermöglicht uns in LaTeX die `itemize`-Umgebung, die mit den Befehlen `\begin{itemize}` und `\end{itemize}` begonnen und beendet werden kann. Die einzelnen Punkte lassen sich dann mit Hilfe des Befehls `\item` auflisten.

Wenn wir die Elemente nicht nur aufzählen, sondern auch nummerieren wollen, so müssen wir hierfür anstatt der `itemize`-Umgebung ein `enumerate` nutzen. Diese können wir äquivalent zur `itemize`-Umgebung erzeugen. Beide Aufzählungen lassen sich auch beliebig verschachteln.

2.4 Bilder

Wir haben bereits einige grundlegende Bestandteile von LaTeX gesehen. Nun wollen wir uns genauer anschauen, wie wir Bilder in unser Dokument einfügen können. Hierfür existiert die `figure`-Umgebung, die wie gewohnt mit `\begin` und `\end` beginnt und endet. Innerhalb der Umgebung können wir die Bilder einbinden, sowie Labels und Bildunterschriften setzen. Um Bilder einbinden zu können, müssen wir allerdings erst das Paket `graphicx` mit dem gewohnten Befehl in der Präambel importieren.

Im Listing 2 ist ein Codebeispiel für ein Bild zu sehen. Die optionalen Parameter in den eckigen Klammern geben an, wo sich das Bild am besten befinden sollte. Wird keine Option mitgegeben, so versucht LaTeXdas Bild an einer geeigneten Stelle unterzubringen. In diesem Beispiel wurde die Option `[!h]` verwendet, um dafür zu sorgen, dass das Bild möglichst an der im Text eingefügten Stelle erscheint. Weitere Parameter sind zum Beispiel `t` für „top“oder `b` für „bottom“.

Listing 2: Beispielhaftes einbinden eines Bildes

```
1 \begin{figure}[!h]
2     \centering
3         \includegraphics[width=0.50\textwidth]{figures/Bild.pdf}
4     \caption{Ein Beispielhaftes Bild}
5     \label{fig:Bild}
6 \end{figure}
```

Der Befehl `\centering` sorgt dafür, dass das Bild zentriert in der PDF erscheint. Als Optionen lassen sich dem `\includegraphics`-Befehl die Größenverhältnisse des Bildes mitgeben. In unserem Beispiel wurde das Bild auf 50% der eigentlichen Größe skaliert. Mit einer `\caption{Text}` kann ein Bilduntertitel gewählt werden und mit `\label{Labelname}` kann das Bild mit einem Label versehen werden, dass du im Text wiederum mit `\ref{Labelname}` referenzieren kannst.

2.5 Die Mathe-Umgebung

Um in LaTeXmathematische Formeln oder Symbole schreiben zu können, müssen wir diese in eine sogenannte Mathe-Umgebung einfügen. Hier gibt es viele verschiedene Arten, von denen ich gleich einige vorstellen möchte. Zuerst müssen wir aber die Pakete `amsmath`, `amssymb`, `amstext` und `amsthm` einbinden. Was die einzelnen Pakete genau alles bewirken, wird an dieser Stelle nicht weiter erläutert, da die Dokumentation der Pakete im Internet sehr gut ist. Es besteht kann dennoch vorkommen, dass du für spezielle mathematische Elemente ein weiteres, hier nicht aufgeführtes, Paket einbinden musst.

Betrachten wir nun die Mathe-Umgebungen genauer. Einfache einzeilige Formeln können einfach mit zwei $ umschlossen werden. So kann beispielsweise inline einfach $f(x) = x^2$ mit `$f(x) = x^2$` geschrieben werden.

Für mehrzeilige Matheumgebungen wollen wir uns die `align`-Umgebung genauer anschauen. Wir können eine `align`-Umgebung mit `\begin{align}` starten und mit `\end{align}` beenden. Die einzelnen Zeilen können dann mit `\newline` oder \\ gebrochen werden und erhalten jeweils eine fortlaufende Nummer. In Listing 3 ist der Code für die folgenden Gleichungen zu sehen.

$$x^2 + y^2 = \frac{1}{2} \tag{1}$$

$$y = \sqrt{\frac{1}{2} - x^2} \tag{2}$$

Listing 3: LaTeX-Code des `align`-Beispiels

```
1 \begin{align}
2 x^2 + y^2 = \frac{1}{2}\\
3 y = \sqrt{\frac{1}{2} - x^2}
4 \end{align}
```

Möchtest du keine Nummerierung hinter den Zeilen, so kannst du anstatt `align` einfach `align*` verwenden, also mit einem Stern angefügt.

Die einzelnen Symbole und mathematischen Konstrukte, wie das oben genutzte `\frac{}{}`, kannst du dir einfach im TeXnicCenter zusammenklicken oder im Internet nachlesen.

3 Listings

Im Folgenden wollen wir uns nun damit beschäftigen, wie man einfach Quellcode in das LaTeX-Dokument einfügen kann. Hierfür eignet sich besonders die `listings`-Umgebung, die man wie gewohnt mit `\usepackage{listings}` importieren muss. Innerhalb einer `listings`-Umgebung werden eigentliche LaTeX-Befehle ignoriert und so dargestellt, wie sie geschrieben wurden. Wie gewohnt beginnt man ein Listing mit dem Befehl `\begin{listings}` und beendet es mit `\end{listings}`. Über optionale Parameter lassen sich dann noch diverse Spezialisierungen festlegen, wie beispielsweise ein Syntaxhighlighting oder ein Rahmen um das Listing.

Achtung: Bitte beachte, dass in meist Quellcode Abgaben per PDF nicht erlaubt sind, sondern direkt als Java-Dateien abgegeben werden müssen!

4 Tikzpicture

Da für Abgaben in Modulen, wie DSA, häufig Graphen, Listen oder Bäume gezeichnet werden müssen, wird an dieser Stelle kurz das Paket Tikzpicture vorgestellt. Mit diesem Paket lassen sich diese Datenstrukturen schön und einfach zeichnen. Das Paket lässt sich einfach mit `\usepackage{tikz}` einbinden.

Schauen wir uns genauer an, wie wir mit Tikzpicture zum Beispiel einen Binärbaum zeichnen können. Hierfür betrachten wir Listing 4 genauer, welcher den Binärbaum aus Abbildung 1 erzeugt.

Listing 4: LaTeX-Code eines Binärbaumes, der mit Hilfe von Tikzpicture erzeugt wurde

```
\begin{center}
\node[circle,draw] (a) {A}
        child {node [circle,draw] (b) {B}
                child {node [circle,draw] (c) {C}}
                child {node [circle, draw] (d) {D}}
        }
        child {node [circle, draw] (e) {E}
                child {node [circle, draw] (f) {F}
                        child[missing] {node {}}
                        child {node [circle, draw] (i) {J}}
                }
                child {node [circle, draw] (g) {G}
                        child {node [circle, draw] (h) {H}}
                        child[missing] {node {}}
                }
        };
\end{tikzpicture}
\end{center}
```

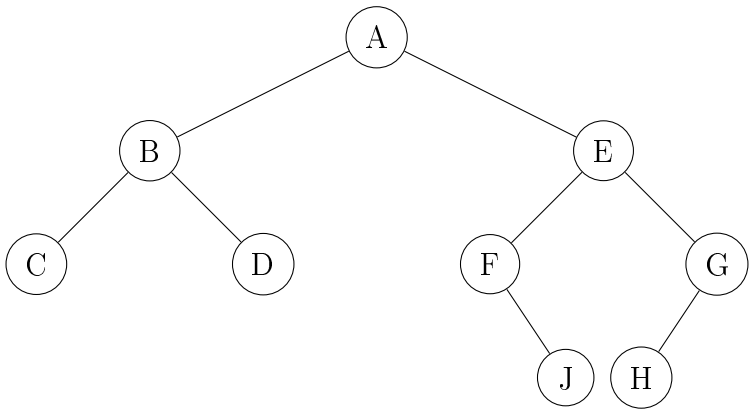

Abbildung 1: Binärbaum, der mit Tikzpicture erzeugt wurde

Das Ganze lässt sich am besten in Hierarchien vorstellen, ähnlich zu inneren Klassen in Java. Wir beginnen die Wurzel mit `\node[...] (Label) {Name}`. Die Parameter erzeugen hierbei den Kreis für den jeweiligen Knoten. Dem Knoten selbst geben wir mit den normalen Klammern ein Label, auf das wir später referenzieren können. Der Inhalt wird in den geschweiften Klammern angegeben.
Die Kinder der Wurzel können wir nun mit `child {node[...] (Label) {Name} ... }` angeben. Für Geschwisterknoten müssen wir nichts weiter beachten und können die jeweiligen Knoten einfach untereinander schreiben. Besitzt eines der Kinder jedoch weitere Kinder, so gehen wir eine Hierarchie tiefer. Hinter dem Namen des neuen Vaterknotens erzeugen wir nun weitere Kinder, bevor wir den aktuellen Knoten mit der geschweiften Klammer beenden. So sind beliebig tiefe Hierarchien möglich.

4.1 Automaten

Mit Tikzpicture lassen sich auch komfortabel Endliche Automaten erstellen. Dieses Feature muss aber zunächst durch `\usetikzlibrary{automata,positioning}` aktiviert werden. Das `positioning` erlaubt es, Positionen relativ mittels `left above` und ähnlichen Schlüsselwörtern anzugeben (statt zum Beispiel über Koordinaten). Dann kann folgendermaßen ein Automat erstellt werden:

Listing 5: LaTeX-Code eines endlichen Automaten

```
\begin{tikzpicture}[initial text={}]
        \node[state,initial] (q0) {$q_0$};
        \node[state] (q1) [below right=of q0] {$q_1$};
        \node[state] (q2) [above right=of q0] {$q_2$};
        \node[state,accepting] (q3) [below right=of q2] {$q_3$};
        \path[->] (q0) edge node[below left]{$a$} (q1);
        \path[->] (q0) edge node[above left]{$b$} (q2);
        \path[->,loop above] (q2) edge node{$b$} (q2);
        \path[->] (q2) edge node[left]{$a$} (q1);
        \path[->] (q1) edge node[below right]{$a$} (q3);
        \path[->,loop below] (q1) edge node{$b$} (q1);
        \path[->,loop right] (q3) edge node{$a,b$} (q3);
\end{tikzpicture}
```

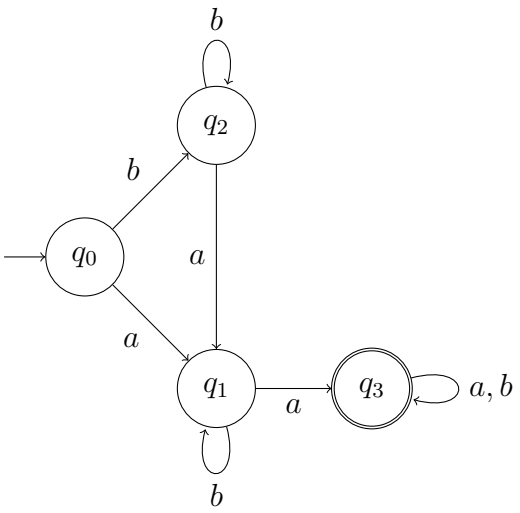

Abbildung 2: Endlicher Automat, der mit Tikzpicture erzeugt wurde

5 Wichtige Symbole für die theoretoische Informatik

Nur in Mathematik-Umgebung:

$\emptyset$	`\emptyset`	Leere Menge bzw. leere Sprache
$\cap$	`\cap` bzw. `\bigcap`	Schnitt bzw. Schnitt vieler Mengen (Funktioniert wie Summe, s. u.)
$\cup$	`\cup` bzw. `\bigcup`	Vereinigung
$\sum_{i=0}^{n}$	`\sum_{i=0}^{n}`	Summe von 0 bis n
$\prod_{i=0}^{n}$	`\prod_{i=0}^{n}`	Produkt
ε	`\varepsilon`	Leeres Wort
$\vdash$	`\vdash`	Konfigurationsübergang
$\mathcal{A}$	`\mathcal{A}`	"Geschwungene" Schrift für Belegungen und ähnliches
$\vee$	`\lor` oder `\vee`	Oder
$\wedge$	`land` oder `\wedge`	Und
$\forall$	`\forall`	Allquantor "für alle"
$\exists$	`\exists`	Existenzquantor "es gibt"

Berechenbarkeit und Komplexität Für BuK interessant ist zudem das Paket `complexity`[6], das Kürzel für Komplexitätsklassen definiert (zum Beispiel `\DSPACE`).

Tipp für weitere Zeichen: Detexify[7] erlaubt es Zeichen zu malen und sich ähnlich aussehende LaTeX-Zeichen anzeigen zu lassen.

[6] `https://www.ctan.org/tex-archive/macros/latex/contrib/complexity`
[7] `http://detexify.kirelabs.org/`