

# Automatische Sprachübersetzung von $\text{\LaTeX}$ -Dokumenten

Name: Hendrik Theede

Matrikelnummer: 221201256

Abgabedatum: 02.12.2025

Betreuer und Gutachter: Prof. Dr. rer. nat. habil. Clemens H. Cap  
Universität Rostock  
Fakultät für Elektrotechnik und Informatik

# Abstrakt

placeholder

# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>1</b>
<b>2</b>	<b>Problemfälle</b>	<b>3</b>
2.1	Herangehensweise . . . . .	3
2.2	Elemente in einem T <sub>E</sub> X-Quellcode . . . . .	5
2.2.1	Die Präambel . . . . .	5
2.2.2	Kommandos . . . . .	5
2.2.3	Optionen . . . . .	6
<b>3</b>	<b>Stand der Technik</b>	<b>17</b>
3.1	Anforderungen . . . . .	17
3.2	Denkbare Ansätze . . . . .	17
3.3	Existierende Ansätze . . . . .	17
3.3.1	Testverfahren . . . . .	17
3.3.2	Durchführung . . . . .	17
3.3.3	Auswertung . . . . .	17
3.4	Grenzen der Lösungen . . . . .	17
3.5	Takeaways . . . . .	17
<b>4</b>	<b>Eigenständigkeitserklärung</b>	<b>18</b>
	<b>Literatur</b>	<b>19</b>
<b>A</b>	<b>Anhänge</b>	<b>20</b>
A.1	Fontskalierung auf Webseiten . . . . .	20

# 1 Einleitung

Die schnellstmögliche und einfache Erstellung von Dokumenten beliebiger Natur (formlos) wird heutzutage oftmals über Produkte bekannter Anbieter abgewickelt (bspw. Microsoft's Word, PowerPoint, etc., die vergleichbaren „LibreOffice“, sowie Apple-Produkte). Unterliegen Dokumente allerdings strengeren stilistischen Vorgaben (bspw. bei wissenschaftlichen Abhandlungen) entsteht der Vorteil, dass sich diese Vorgaben wie ein Regelsatz behandeln lassen, aus welchem bestimmte, feste Dokumenten-Strukturen hervorgehen. Unter- zu existiert bereits ein geläufiges System namens  $\text{\LaTeX}$  (mit dem *La* nach einem der ursprünglichen Entwickler L<sup>a</sup>mp<sup>o</sup>rt (1994)), welches selbst auf dem von Knuth (1986) entwickelten Zeichensetzungs-System und der verbundenen Programmiersprache  $\text{\TeX}$  basiert. Die  $\text{\TeX}$ -Syntax selbst basiert auf englischen Begriffen, allerdings ist nicht davon auszugehen, dass nur englischsprachige Menschen  $\text{\LaTeX}$  und  $\text{\TeX}$  nutzen werden. Quelltexte und Dokumentenbeschreibungen werden also nicht immer in einer rein englischsprachigen Form vorliegen (z.B. `\chapter{Erstes Kapitel: Einleitung}` oder der Quelltext dieses Werkes). Ein Zurückführen solcher Dokumente in die englische Sprache ist einfach, insofern ein Verständnis der deutschen Sprache besteht (`\chapter{First Chapter: Introduction}`). Die andere Richtung wirft allerdings eine Mehrzahl an Problemen auf, wenn ohne Vorkenntnisse von  $\text{\TeX}$  (bzw. dessen syntaktische Elemente) übersetzt wird. In genanntem Beispiel würde dann das Wort `chapter` aufgegriffen werden und die Zeichenkette `\Kapitel{Erstes Kapitel: Einleitung}` entstehen (ohne weitere, jedoch mögliche Anpassungen entsteht hier keine Kapitelüberschrift mehr. Das erstere „Kapitel“ würde ignoriert werden und die innerhalb der Klammern stehende Zeichenkette als einfacher Fließtext gedruckt werden).

Shannon (1948) beschäftigte sich bereits 1948 mit wesentlichen Grundlagen der heutigen Darstellung und Übertragung von Informationen, insbesondere der menschlichen Sprache und Kommunikation. Heutige maschinelle Systeme zu diesem Zweck (bspw. ChatGPT, DeepL, Gemini und co.) wirken zunächst wie „magische“ Blackboxen, arbeiten jedoch auf Grundlage von statistischen Modellen. Spricht man hier von Magie, dann ist jeder Mitarbeiter eines Wetterdienstes oder der Klimaforschung „bezaubernd“. Das zugrundeliegende Konzept kann jedoch sehr schnell auf den Punkt gebracht werden: Eine künstliche Intelligenz (KI) erhält einen Input, für welchen ein bestimmter Output erwartet wird (Beispiel: „Einfügen“ als nächstes Wort eines zu übersetzenden Satzes und „insertion“ im Kontext innerhalb des Satzes als Erwartung (substantiviertes Verb)), und produziert einen Output, welcher mit dem Erwarteten abgeglichen wird. Sollte das Resultat von der Erwartung abweichen (z.B. „insert“ entstehen), so kann dieser Fehler erkannt werden und im Modell dazu beitragen, dass (gegeben einer bestimmten, sequentiellen Folge von Wörtern (in der Satzstruktur)) dieser „Fehler“ von nun an seltener passiert. Allerdings wird klar, dass eine KI unabdingbar Fehler machen muss, denn nur so kann diese „lernen“. Diese theoretische Grundlage führt dazu, dass immer alle möglichen Permutationen einer Übersetzung in Betracht gezogen werden müssen, so unwahrscheinlich sie auch seien. Angemerkt sei zu dem Vorherigen, dass bekannte und bereits rein in den menschlichen Sprachen auftretende Problem (bzw. mögliche Missverständnisse bereits im Verstehen einer Sprache, ausgelöst durch Mehrdeutigkeiten von Wörtern) in dieser Arbeit nicht näher verfolgt werden.

Bekannte Technologien, wie z.B. Google Translate, DeepL und co. scheinen rein wort-interne Probleme zu lösen. Sprachliche Missverständnisse könnten aber immer dann entstehen, wenn sich mehrere Sprachen miteinander vermischen, welche sich ein Lexikon teilen (bspw. hier:  $\text{\TeX}$ ,  $\text{\LaTeX}$ , ... und die, zunächst, englische Sprache). Die Frage ob ein Wort übersetzt werden darf oder nicht, unterscheidet einzelne Problemarten (Fälle).

Die Hoffnung besteht, dass diese Probleme bereits gelöst sind, allerdings wird *ein Übersetzer* in der folgenden Schilderung einzelner Problemfälle Fehler machen können *müssen*.

## 2 Problemfälle

### 2.1 Herangehensweise

Ähnlich wie ein  $\text{\TeX}$ -Dokument selbst entsteht, können auch potentielle Fehler entstehen, wenn Quelltexte der Art übersetzt werden. Hierbei sind zuerst alle möglichen Fehler innerhalb eines einzelnen und „reinen“  $\text{\TeX}$ -Quellcodes abzudecken und danach werden Fehler ersichtlich, welche bei der Ausweitung auf Mehrere entstehen (auf Grundlage der Tatsache, warum diese anderen Quelltexte entstehen *könnten* und teilweise *müssen*). Ausgehend von den kleinen Strukturen innerhalb einer Quelltextdatei wird sich zu immer größer werdenden Strukturen hingearbeitet. Zudem sind ein paar zusätzliche, sprachliche und teils unlösbare Probleme gelistet, welche nicht unbedingt als Anforderungen der gegebenen Problemstellung zu verstehen sind und daher als „abweichend“ zu verstehen sind, aus welchen sich aber spätere Erweiterungspotentiale zeigen könnten.

#### Struktur einzelner Beispiele

Einzelne Beispiele werden nacheinander in einer originalen Datei und einer idealen, richtigen Übersetzung dargestellt. Anstatt jegliche mögliche Permutation zu listen, eignet es sich konkrete Muster abzuleiten und anhand dieser zu beschreiben, an welchen Stellen übersetzt werden darf und an welchen nicht. Das einleitende Beispiel (1) zeigt allerdings auf, welche Fehler sich bei einem imaginären Befehl `ink` zeigen könnten und inwiefern einige es erlauben *würden*, dass kleinere Fehler missachtet werden können. Der Befehl selbst soll einen String mit einer bestimmten Farbe hinterlegen und besitzt einen zusätzlichen optionalen Farbparameter. „Richtig“ wäre es im originalen String nur das Wort in den geschweiften Klammern zu übersetzen, da hierbei an keiner Stelle Information verloren geht und das Wort, nachdem es vom Deutschen ins Englische übersetzt wurde, weiterhin so wie vorgesehen hervorgehoben wird. „Zulässige“ Übersetzungen treten dann auf, wenn nur für die Formatierung (insofern hieraus keine weiteren Probleme entstehen) verloren geht. Im gegebenen Beispiel würde dann zwar die farbige Hinterlegung verloren gehen, das Wort allerdings trotzdem übersetzt werden und würde den Weg in ein Dokument finden, ohne einen sprachlichen Informationsverlust zu riskieren (für den Endnutzer/Leser).<sup>1</sup> „Unerwünscht“ sind Fälle, in denen ein Übersetzen Fehler für die  $\text{\TeX}$ -Engine produziert. Übersetzt man hier z.B. `ink` nach `Tinte` könnte es sich bei Weiterem wiederum um einen anderen Befehl handeln, das Wort *Wort* einliest, aber eigentlich den alphanumerischen Wert von *word* erwartet hätte.<sup>2</sup> Ein Fehlschlagen des Befehls `Tinte` würde zwar einen Fehler für den  $\text{\TeX}$ -Parser produzieren, dieser wüsste dann aber, dass dieser Befehl bereits einmal fehlgeschlagen ist und eine neues Kompilieren verlangen, in welchem dieser Befehl und seine Optionen ignoriert werden, wodurch das Wort auch hier im Dokument landen würde. Man kann allerdings nicht bei jedem beliebigen  $\text{\TeX}$ -Befehl davon ausgehen, dass dieses Verhalten einheitlich auftreten wird. Hierbei existieren Fälle, welche dafür sorgen könnten, dass andere Wörter nun nicht mehr Teil eines Dokumentes werden könnten ?? „Fehlerhaftes“ Verhalten beim Übersetzen von  $\text{\TeX}$ -Quelltextdateien führt zu einem Informationsverlust, da das zu übersetzende Wort entweder nicht mehr übersetzt wird oder nicht mehr im Dokument wiederzufinden ist. Sobald man beginnt mit mehreren Dateien ein einziges Dokument zu beschreiben, riskiert ein naïves Übersetzen nur von einem Quelltext ausgehend, dass aus unerwünschten Fehlern innerhalb

<sup>1</sup>Selbst bei weißer Schriftfarbe kann das Wort in einem PDF-Reader markiert und kopiert werden.

<sup>2</sup> $57_{16} + 6f_{16} + 72_{16} + 74_{16} = 25 \times 16^1 + 28 = 400$  statt:  $77_{16} + 6f_{16} + 72_{16} + 64_{16} = 26 \times 16^1 + 28 = 416$ . Wofür der Befehl `Tinte` einen/den Integer 416 benötigt, kann ich Ihnen allerdings nicht erläutern.

von einem Dokument fehlerhaftes Verhalten für das entstehende Produkt (meint: die kompilierte PDF) entsteht. Abstrahiert man von diesem detaillierterem Beispiel, so sind „richtige“ Übersetzungen frei von Informationsverlust, „zulässige“ Übersetzungen nur dazu fähig Informationen für die graphische Aufbereitung (allerdings nicht den sprachlichen Inhalten) zu entwenden, „unerwünschte“ Übersetzungen dazu in Lage Informationen verbergen können und „falsche“ Übersetzungen fehlende sprachliche Inhalte innerhalb eines Dokumentes, sowie fehlende Übersetzungen dieser. Nicht jede Gruppe von Beispielen führt dazu, dass alle benannten Kategorien auftreten.

## 2.2 Elemente in einem T<sub>E</sub>X-Quellcode

### 2.2.1 Die Präambel

Der unsichtbare Header eines T<sub>E</sub>X-Dokumentes zeigt an (zunächst) wenigen, sehr spezifischen Stellen eine Schwierigkeit auf. In dieser sind meistens Informationen enthalten, welche nicht zu übersetzen sind, da sie z.B. Parameter für dokumentenweite (globale) Einstellungen setzen. Die Art und Weise *diese* zu setzen ist in späteren Fehlerbeschreibungen theoretisch abgedeckt und wäre daher eigentlich nicht von weiterem Interesse, eignet sich allerdings dazu, einige triviale Fehlerquellen abzudecken (bzw. ungeeignete Ansätze). So kann man sich nicht immer gewiss sein, man einzelne Zeilen gesondert auswerten kann, da z.B. Pakete wie hyperref es in ihren Optionen (hypersetup) erlauben einige Einstellungen durch Zeilenbrüche voneinander getrennt darzustellen. Nur weil ein Quelltext also T<sub>E</sub>X Syntaktik beinhaltet (und demnach T<sub>E</sub>X-Semantik trägt), kann nicht direkt ein ganzer Quelltext von der Übersetzung ausgeschlossen sein, darf aber auch nicht vollständig übersetzt werden, ohne einzelne Zeilen genauer zu betrachten. Genauso sind allerdings auch einzelne Zeilen nicht direkt als „insgesamt T<sub>E</sub>X syntaktisch“ zu betrachten, nur weil ein syntaktisches Element von T<sub>E</sub>X innerhalb dieses Strings vorliegt. Auch innerhalb einzelner Zeilen (bspw. `\title{carriage return, line feed}`) können sowohl T<sub>E</sub>X-syntaktische (dadurch: T<sub>E</sub>X-Semantik tragende), als auch wortsprachliche Inhalte vorliegen, welche bei einem Übersetzen nicht verloren gehen dürfen. (Genanntes Beispiel müsste in: `\title{Karren-Rückkehr, Zeilen-Einspeisung}` oder Ähnliches übersetzt werden).

### 2.2.2 Kommandos

Kommandos selbst sind in der reinen T<sub>E</sub>X-Syntax durch ein Backslash `\` gekennzeichnet. Die Frage, ob das Wort, das einem Kommando folgt, übersetzt werden sollte, oder nicht, bringt Konflikte<sup>3</sup>. Beispielsweise müsste innerhalb von Auflistungen der nach einem `\item` folgende String übersetzt werden. Andererseits wäre in einer Definition mit `\def` vorerst davon abzugehen den folgenden String zu übersetzen, da dieser ein neues Makro/-Kommando/etc. definieren zu sucht. Demnach darf `\def\hello` nicht zu: `\def\hallo` werden (es sei denn alle folgenden `\hello` werden auch übersetzt).<sup>4</sup>

Innerhalb des Quelltextes können also Strukturen auftreten, welche direkt folgende Zeichenketten als syntaktisches Element zur Dokumentenbeschreibung benötigen. Da hierbei beide Fälle auftreten könnten und nicht an dem Kommando selbst erkennbar sind, muss hierbei nach Art des Kommandos selbst abgewogen werden. Demnach genügt es nicht nur reine Befehle anhand des `\` zu erkennen oder als festes Maß dafür zu nehmen, ob ein folgender String (der kein weiteres Muster aufweist) zu übersetzen ist, oder nicht.

### Parameter

**Geschwungene Klammern** Auf viele Kommandos (Makros) folgen ein Paar geschwungener Klammern. Die Inhalte dieser sind in einigen Fällen zu übersetzen und in Anderen nicht, aber insgesamt als eine Art

<sup>3</sup>Kommandos selbst werden hier noch als fester Teil der T<sub>E</sub>X-Syntax betrachtet und als auszuschließender Teil für den Übersetzer gewertet

<sup>4</sup>Da aber, wie bereits bekannt, immer eine Wahrscheinlichkeit, so gering sie auch sein mag, besteht, dass an einer nicht vorgesehenen Stelle eine Übersetzung auftreten *könnte*, kann man sich nicht auf die vorherige Aussage verlassen



Parameter für dieses Kommando zu verstehen.

Beispielsweise wäre das Kommando `\paragraph{This is a string of many characters}` eines, in welchem die Inhalte der Klammern übersetzt werden sollen und das Kommando `\usepackage{geometry}` ein Beispiel, in welchem das Übersetzen von `geometry` zu `Geometrie` dazu führen würde, dass das entsprechende Paket nicht im Quelltext verwendet wird. Diese Art von Fällen, welche der Übersetzung ausgeschlossen werden soll, ist besonders kritisch, wenn mit mehreren Dateien gearbeitet wird und so würde ein Übersetzen von z.B. `\include{clock.tex}` (oder `\include{clock}`) zu `\include{Uhr.tex}` (oder: `\include{Uhr}`) dazu führen, dass größere Teile eines Dokumentes komplett ausgeschlossen werden. Klammern dieser Art genügen also alleine noch nicht dafür eine Aussage darüber zu treffen, ob innenliegende Strings übersetzt werden dürfen, allerdings diese Zeichen (meint: die geschwungenen Klammern) kein Teil direkter Teil einer menschlichen Sprache und man würde davon meist erwarten, dass diese Klammern im Normalfall ein Indikator dafür sind, dass nicht übersetzt werden soll und in bestimmbar Ausnahmefällen (bspw. `\section{}`, `\paragraph*{}`, `\chapter{}`, `\title{}`, etc.) von Interesse für ein Übersetzen sind.

**Nutzer-eigene Klammern** Klammern für Parameter und Optionen belaufen sich nicht nur auf eckige und Geschwungene, sondern jegliche Zeichen könnten als Klammern betrachtbar werden. Einfachstes Beispiel liefert hierbei der `\verb`-command, dessen (auf diesen) folgendes Zeichen als Klammer für die umschlossene Zeichenkette (bis benanntes Zeichen wieder auftritt) zu betrachten sei. Daher wäre ein `\verb|something|` mit verschiedensten Sonderzeichen denkbar (bspw. `+, -, *, /, ...` anstatt `|`). Darüber hinaus lässt es der Befehl `\catcode` zu, dass einzelne Zeichen, die üblicherweise in der standardmäßigen T<sub>E</sub>X-Syntax eine eigene Semantik tragen (bspw. eckige, geschwungene Klammern oder das Backslash, die Tilde oder das At-Zeichen, bzw. Wort-Charaktere oder Zahlen), eine neue/andere Semantik ggb. des T<sub>E</sub>X-Parsers erhalten könnten.

Hieraus entsteht also eine darstellbare Zeichenkette nach dem Kompilieren von T<sub>E</sub>X, ein Übersetzer könnte allerdings Strings, wie: `\verb AthingA` finden, in welcher ein Versuch einer Rechtschreibkorrektur in einer Übersetzung von nicht: `\verb ADingA`, sondern: `\verb Ein Ding enden` könnte (wodurch bis zum nächsten E im Text alles als die tatsächliche und nicht weiter interpretierte Zeichenkette gedruckt werden könnte, was in einigen Dokumenten Textüberläufe produzieren kann).<sup>5</sup>

Inwiefern sich dies auf einen Informationsverlust für ein Dokument selbst auswirkt, ist fraglich, denn

### 2.2.3 Optionen

**Darstellungsform** Optionen sind, neben Parametern, eine weitere Möglichkeit einem Kommando in T<sub>E</sub>X zusätzliche Informationen mitzuliefern, z.B. für zusätzliche Formatierung. Aus logischer Reihenfolge müssten diese Optionen *eigentlich* immer vor der Zeichenkette, welche formatiert werden soll, stehen, da erst nach Einlesen der Parameter die Information dieser erkannt und auf den folgenden String angewendet werden kann. Diese Aussage wäre logisch, gilt allerdings nicht einheitlich (bspw. bei der Verwendung von TikZ-Bibliotheken). Optionen in eckigen Klammern insgesamt einer Übersetzung zu entziehen, verhindert allerdings die Möglichkeit, dass die Optionen einiger Befehle selbst ein String sein könnten, welcher im Dokument gedruckt und somit übersetzt werden muss. Die übliche T<sub>E</sub>X-Notation sieht hierbei geschwungene Klammern für zwingend erforderliche Parameter vor

---

<sup>5</sup>Inwiefern sich dies auf einen Informationsverlust für ein Dokument selbst auswirkt, ist fraglich, denn eigentlich liegen alle Informationen immer noch in der virtuellen Datei vor.

und Eckige für optionale. Einzelne Kommandos erwarten zunächst optionale, zusätzliche Parameter in eckigen Klammern, welche vorher festgesetzt wurden und danach übrige, benötigte in Geschwungenen (ein Beispiel hierfür zeigt 2.2.3). Für manche Befehle impliziert allerdings der Befehl selbst (bspw. `includegraphics`), dass eine Parameter folgen muss (hier: URL, bei welchen auch erwartet werden muss, dass sie menschengesprochene Wörter/Phrasen enthalten). Aus theoretischer Sicht wäre die Menge an möglichen Parametern unendlich,  $\text{\TeX}$  selbst begrenzt diese jedoch auf 9 und setzt damit ein technisches Limit. Dieses ist auf verschiedene Arten umgehbar,<sup>6</sup> zeigt aber nur Probleme auf, wenn in irgendeiner Form mit Name-Wert-Paaren gearbeitet wird, in welchen der erstere Key nicht übersetzt werden darf, jedoch zweiter Parameter es soll.

**Unvorhersagbarkeiten** Eigene, definierte Makros erlauben für eine quasi-beliebige Zahl an Stellen, innerhalb welcher Strings erwartet werden könnten. So könnte beispielsweise ein Befehl definiert werden, welcher 7 Eingabeparameter erwartet: `\def\whatever a#1a#2a#3a#4a#5a#6a#7{This #1 is #2 to #3 be #4 troublesome. Hence we will only print one hundred now}` bei einem Aufruf der Form `\whatever aMACROgoingtoaverya1a0a0` erwarten, dass sowohl alle menschengesprochenen Zeichenketten innerhalb der Definition (*This is to be troublesome. Hence we will only print one hundred now*), als auch außerhalb dieser (*MACRO going to very*) übersetzt werden und einen schlüssigen Satz bilden.

Dies ist auf technischer Ebene nachvollziehbar (`\whatever` produziert: „This MACRO is going to to be very troublesome. Hence we will only print one hundred now 100“), kann aber erneut zu unendlichen Möglichkeiten führen,

## Strukturen

**Umgebungen und Makros** Große Strukturen in der  $\text{\TeX}$ -Syntax zeigen sich oftmals in sog. Umgebungen auf und stellen gerne/oft Graphiken dar (wie bspw. in `TikZ`), müssen dies allerdings nicht zwingend. Genauso wäre zu erwarten, dass größere Texte innerhalb dieser Umgebungen reinen Text-Formatierungen obliegen und damit gänzlich zu übersetzen wären. Allerdings können verschiedene Umgebungen auch eigene Syntaktik tragen (bspw. Tabellen), wodurch sich innerhalb von solchen Umgebungen sowohl zu übersetzende Strings, als auch zu Erhaltende (= nicht zu übersetzende Strings) verbergen können.


### Vordefinierte

Insbesondere Problematisch wird die Ermittlung solcher bei Umgebungen, welche nicht mittels entsprechender `begin` und `end` Kommandos betreten/verlassen werden, sondern auf welche mittels eigener Symbolik ein access gewährt werden kann. Innerhalb reinen  $\text{\TeX}$ 's sind diese an wenigen Zeichenketten beschränkt und so können nur ein/zwei Dollar-Zeichen ( $\$, \$\$$ ) oder  $\backslash$  ( $\backslash$ ),  $\backslash$  [ $\backslash$ ],  $\backslash$ ] auf den/das Beginn/Ende einer ein-/ mehrzeiligen mathematischen Umgebung hindeuten. Darüberhinaus erlauben gerade diese Umgebungen den Wechsel in *normale* Umgebungen (also hinein in die Textumgebung des eigentlichen Dokumentes, aus welcher solche mathematische Umgebungen zu fliehen suchten) innerhalb welcher wieder in beschriebene mathematische Umgebungen gewechselt werden kann. Solange auf jeden Wechsel *in* eine solche Umgebung ein Wechsel *aus* einer solchen Umgebung heraus erfolgt, kann  $\text{\TeX}$  dies interpretieren und alle rein textlichen Strings *sollten* übersetzt werden.

<sup>6</sup> Einzelne Parameter tragen mehr Informationen. Verknüpfungen mit Relays oder Newcommand in Verbindung mit *key-value*-Paaren (elegantere Lösung, bspw. in `hyperref's` zu finden)

**Tabellen und Formeln** Tabellarische Strukturen eignen sich für eine übersichtliche Darstellung von z.B. Messwerten oder auch mathematischer Formeln. Zwar ist bei z.B.

$$\begin{aligned}\sin(\omega t - k\vec{r}) &= 0 \\ \arcsin(\sin(\omega t - k\vec{r})) &= \arcsin(0) \\ (\omega t - k\vec{r}) &= 0 \\ \omega t = k\vec{r} \text{ note that: } \omega &= 2\pi f \text{ (frequency)}\end{aligned}$$

keine „gewöhnliche“ Tabelle zu erkennen, allerdings  man sich einzelne Terme innerhalb vor und/oder nach einer Äquivalenzrelation wie Inhalte einer Zelle vorstellen. Gegebenes Beispiel erhält seine Struktur im T<sub>E</sub>X-Quelltext durch eine eigene Syntax innerhalb solcher Umgebungen. Hierbei kann allerdings auch der Fall, wie zum Beispiel innerhalb einer Tabelle, Strings einzelner Zellen zu übersetzen sind, andere jedoch nicht. Bereits obiges Beispiel zeigt Wörter, welche interessant für einen Übersetzer sein sollten. Anders ist dies jedoch bei

**Eigene Makros und Logik**    Unterer Beispielsatz ist eine Möglichkeit einen String (Hello to this world:) zu erzeugen. Jedoch ist mit dieser Definition auch der String: Goodbye for now in der Form `\appendstring{now}{Goodbye}{for}` produzierbar. Diesem Muster entsprechend könnten (theoretisch) unendlich viele eckige Klammern zu übersetzende Strings beinhalten. Dies wäre zunächst unproblematisch unter der Vorannahme, dass alle diese Strings zu übersetzen sind. Was allerdings, wenn eine Definition der Form `\newcommand{\addandappend}[3][1][2]{#1+#2 ist #3 #1+}` vorliegt, in welcher für #3 ein String (hier denkbar: gleich) erwartet wird? Solche Erwartungen sind nicht immer vorhersagbar und können dadurch nur schwer beschrieben werden (sind allerdings deterministisch nachvollziehbar, da die  $\text{\TeX}$ -Engine solche Makros schließlich auch interpretieren und rendern kann).

Problematisch wird dieser Fall, wenn solche Makros eigene und optionale Parameter in beliebiger Reihenfolge erwarten wollen, wie das folgende Beispiel ( ??). Übersetzt man z.B. das Wort „sunny“ auch nur an einer Stelle nicht, riskiert man Fälle, in welchen falsche Inhalte angezeigt werden. Auch kann man sich nicht gewiss sein, dass zu übersetzende Optionen oder Parameter immer in einheitlicher und vorhersagbarer Reihenfolge auftreten werden. Ob man einen String innerhalb eines Makros übersetzen darf, bestimmt sich danach, ob das String-Literal logisch benötigt wird. Dies ist daran erkennbar, dass wie im gelisteten Beispiel Vergleiche mit dem String geschehen und liegt zur Kompilierzeit im Quelltext vor.

---

## Englisches Original

```
1\ink[red]{word}
```

T<sub>E</sub>X Code 1: Original

---

## Richtige Übersetzung

```
1\ink[red]{Wort}
```

T<sub>E</sub>X Code 2: Beispielübersetzung

---

## Zulässiges Verhalten

```
1\ink[Rot]{Wort}
```

T<sub>E</sub>X Code 3: Beispielübersetzung

---

## Unerwünschtes Verhalten

```
1\Tinte[Rot]{Wort}
```

T<sub>E</sub>X Code 4: Beispielübersetzung

---

## Falsches Verhalten

```
1\Tinte[Rot]{word}
```

T<sub>E</sub>X Code 5: Beispielübersetzung

```
1\begin{table}[h!tb]
2  \centering
3  \begin{tabularx}{\textwidth}{X X X}
4    \toprule
5      English words & Value of the first word & Value of the second word \\
6    \midrule
7      finite fields & 639 & 663 \\[-13px]
8      limited areas & 744 & 556 \\[1em]
9    \midrule
10     Galois field & 607 & 548 \\[-13px]
11     infinite set & 854 & 364 \\[1em]
12   \midrule
13     place & 517 & \\[-13px]
14     set & 332 & \\[-1em]
15   \bottomrule
16 \end{tabularx}
17 \caption{Several english strings and the summed up ASCII-values of their
    ↪respective words.}
18\end{table}
```

TEX Code 6: Original

```
1\begin{table}[h!tb]
2  \centering
3  \begin{tabularx}{\textwidth}{X X X}
4    \toprule
5      English words & Value of the first word & Value of the second word \\
6    \midrule
7      finite fields & 639 & 663 \\[-13px]
8      limited areas & 744 & 556 \\[1em]
9    \midrule
10     Galois field & 607 & 548 \\[-13px]
11     infinite set & 854 & 364 \\[1em]
12   \midrule
13     place & 517 & \\[-13px]
14     set & 332 & \\[-1em]
15   \bottomrule
16 \end{tabularx}
17 \caption{Several english strings and the summed up ASCII-values of their
18 ↪respective words.}
18\end{table}
```

TeX Code 7: Beispielübersetzung

---

Tabelle 2: Beispiel für die Vermischung von Tabellarischen Strukturen und Texten



---

English Original

---

```
1\newcommand*{\appendstring}[3]{#2 #3 #1}  
2\appendstring{world:}{Hello}{to this}
```

T<sub>E</sub>X Code 8: Original

---

Ideale Übersetzung

---

```
1\newcommand*{\appendstring}[3]{#2 #3 #1}  
2\appendstring{Welt:}{Hallo}{an diese}
```

T<sub>E</sub>X Code 9: Beispielübersetzung

---

Tabelle 3: Beispiel für die Fähigkeit, dass Makros einzelne Strings einlesen können

```
1\newcommand{\weather}[2][sunny]
2{
3  Heute war es
4  \ifthenelse{\equal{#1}{sunny}}
5  {sonnig.}
6  {nicht sonnig. Hoffentlich ist es morgen nach dem Regen wieder
7    \ifthenelse{\equal{#2}{rainy}}
8    {
9      regnerisch, da die Wälder zu trocken sind.
10   }
11   {
12     wieder sonnig werden.
13   }
14 };
15}
16
17\weather{sunny}\\\\
18\weather{rainy}\\\\
19\weather[rainy]{sunny}\\\\
20\weather{rainy}{rainy}\\\\
21\weather{rainy}{rainy}\\\\
```

TeX Code 10: Original

```
1\newcommand{\weather}[2][sonnig]
2{
3  Heute war es
4  \ifthenelse{\equal{#1}{sonnig}}
5  {sonnig.}
6  {nicht sonnig. Hoffentlich ist es morgen nach dem Regen wieder
7    \ifthenelse{\equal{#2}{regnerisch}}
8    {
9      regnerisch, da die Wälder zu trocken sind.
10   }
11   {
12     wieder sonnig werden.
13   }
14 };
15}
16
17\weather{sonnig}\\\\
18\weather{regnerisch}\\\\
19\weather[regnerisch]{sonnig}\\\\
20\weather{regnerisch}{sonnig}\\\\
21\weather{regnerisch}{regnerisch}\\\\
```

TEX Code 11: Beispielübersetzung

---

Tabelle 4: Beispiel für die Fähigkeit, dass Makros Strings logisch verarbeiten können

## **3 Stand der Technik**

### **3.1 Anforderungen**

Abgelitten aus der Problemliste werden hier die Probleme umformuliert als Anforderungen dargestellt und in absteigender Reihenfolge nach Relevanz in Bezug auf die gegebene Aufgabenstellung aufgeführt.

Die Technologien dienen den Anforderungen, sollten sie:

1. kompilierbare Dokumente erzeugen
2. alle Abschnitte in Dokumenten übersetzen
3. kontextuell terminologisch richtige Übersetzungen wählen (die richtigen Lexeme/Wörter treffen)
4. den Kontext selbstständig aus den wörtlichen und erreichbaren (lokalen) Informationen (Dateien) ablesen können
5. den Kontext aus den mathematischen, graphischen, tabellarischen, ... Inhalten einer Datei ablesen können
6. den Kontext aus externen Verweisen (Links) erfassen können (Lokal, als auch Web)
7. ...

### **3.2 Denkbare Ansätze**

Alle Lösungswege und Workflows, die ich mir vorstellen kann und denken konnte. Definiert evtl. Rollen,

### **3.3 Existierende Ansätze**

Alle Technologien, die diese Rolle (n) in den entsprechenden Ansätzen füllen könnten.

#### **3.3.1 Testverfahren**

logischerweise: In den denkbaren Ansätzen schon gegenargumentieren, was unsinnig ist und warum. Reduziert die Menge an zu testenden Lösungen.

#### **3.3.2 Durchführung**

#### **3.3.3 Auswertung**

### **3.4 Grenzen der Lösungen**

### **3.5 Takeaways**

## 4 Eigenständigkeitserklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit selbstständig angefertigt und ohne fremde Hilfe verfasst habe. Dazu habe ich keine außer den von mir angegebenen Hilfsmitteln und Quellen verwendet und die den benutzten Werken inhaltlich und wörtlich entnommenen Stellen habe ich als solche kenntlich gemacht. Ich versichere, dass die eingereichte elektronische Fassung mit den gedruckten Exemplaren übereinstimmt.

Rostock, den 02.12.2025

---

Hendrik Theede

## Literatur

Knuth, D. E. (1986), *The TeXbook*, ISBN: 9780201134476, Addison-Wesley Professional.

Lamport, L. (1994), *LaTeX: A Document Preparation System, 2nd Edition*, ISBN: 9780201529838, Addison-Wesley Professional. available at: <https://www.latex-project.org/help/books/tlc3-digital-chapter-samples.pdf> (last Access: 04.10.2025).

Shannon, C. E. (1948), 'The mathematical theory of communication', *The Bell System Technical Journal*, ISSN: 0343-6993 (vol. 27). Harvard Reprint available at <https://people.math.harvard.edu/~ctm/home/text/others/shannon/entropy/entropy.pdf> (last Access: 16.10.2025).

## A Anhänge

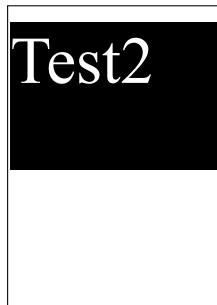
### A.1 Fontskalierung auf Webseiten

Beispielsweise produziert die folgende HTML-Notation bei einer Skalierung im Browser von 120 Prozent (Abbildung 2a) und 50 Prozent (Abbildung 2b) jeweilig zwei verschiedene PDF (unter welchen nur Zweitere alle textlichen Inhalte offenbart). Ähnliches kann auch innerhalb  $\text{T}_{\text{E}}\text{X}$  geschehen, sollte

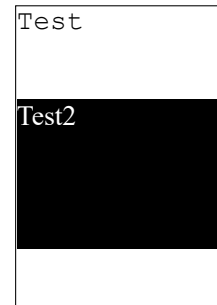
```
<html>
  <head>
    <title>Example</title>
    <style>
      /*formatting options are: none and black*/
      .t{
        font-size:13em;
        height:50%;
      }
      /*formatting option: none = no background, black, courier*/
      .t#none{
        font-family: 'Courier New', Courier, monospace;
      }
      /*formatting option: black = black background, white, serif*/
      .t#black{
        background-color:black;
        color:white;
        margin-top: -2em;
      }
    </style>
  </head>
  <body>
    <div class="t" id="none">Test</div>
    <div class="t" id="black">Test2</div>
  </body>
</html>
```

Abbildung 1: HTML-Beschreibung einer Webseite mit zwei Textflächen

Abbildung 2: Um die Dokumente von der restlichen Papierfläche abzugrenzen wurden schwarze Rahmen mittels TikZ hinzugefügt.



(a) Zuvorige HTML-Beschreibung liefert bei einer Browser-Skalierung von 120% obige Graphik



(b) Zuvorige HTML-Beschreibung liefert bei einer Browser-Skalierung von 50% obige Graphik