

Automatische Sprachübersetzung von \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

1	Einleitung	1
2	Problemfälle	2
2.1	Herangehensweise	2
2.1.1	Struktur einzelner Beispiele	2
2.2	Elemente in einem Quelltext	3
2.2.1	Kommandos	3
2.2.2	Zusätzliche Parameter	4
2.2.3	Makros und Umgebungen	6
2.3	Mehrere Quelltexte	8
2.3.1	T _E X und Andere	8
2.3.2	Pakete und Klassen	8
2.3.3	Hilfsdateien	9
2.3.4	Zitationen	9
2.4	Spezifisch	10
2.4.1	Sprachliche Hürden	10
2.4.2	Layouting-Probleme	10
3	Stand der Technik	11
3.1	Anforderungen	11
3.2	Denkbare Ansätze	11
3.3	Existierende Ansätze	11
3.3.1	Testverfahren	11
3.3.2	Durchführung	11
3.3.3	Auswertung	11
3.4	Grenzen der Lösungen	11
3.5	Takeaways	11
4	Eigenständigkeitserklärung	12
	Literatur	13
A	Anhänge	14
A.1	Fontskalierung auf Webseiten	14

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 \LaTeX (mit dem *La* nach einem der ursprünglichen Entwickler L^amp^ort (1994)), welches selbst auf dem von Knuth (1986) entwickelten Zeichensetzungs-System und der verbundenen Programmiersprache \TeX basiert. Die \TeX -Syntax selbst basiert auf englischen Begriffen, allerdings ist nicht davon auszugehen, dass nur englischsprachige Menschen \LaTeX und \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 \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: \TeX , \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

T_EX-Quellcodes „einfach“ zu übersetzen, birgt Gefahr Fehler zu produzieren. Bereits innerhalb eines T_EX-Quellcodes können (theoretisch) unendlich viele Fehler entstehen, sollten (wohlmöglich) vorfindbare Zeichenketten unbedacht übersetzt werden („unbedacht“: ohne vorher zu prüfen, ob das Wort eine semantische Bedeutung für T_EX trägt). Bevor sich Unendlichkeiten genähert wird, werden hier, abhängig von der Größe der innerhalb T_EX-Quelltexten auftretenden Strukturen (welche der vorgesehenen Struktur folgen Knuth (1986)), jegliche Form von denkbar auftretenden Fehlerquellen präsentiert. Hierbei wären die kleinsten denkbaren Strukturen, sollte man sich auf den eigentlichen Verwendungszweck von T_EX konzentrieren, einzelne Zeichen oder Buchstaben. Diese allein bilden allerdings noch keine logische Struktur in T_EX, sondern erst konkrete Befehle/Funktionen und diesen übergebbare Parameter. Hierbei zeigen sich diverse Wege, wie sowohl die Übergabe dieser Parameter aussehen *könnte*, als auch die Parameter selbst. Die Sinnhaftigkeit einzelner Formen solcher Parameter könnte fraglich sein, kann aber mittels T_EX produziert werden und muss daher betrachtet sein. Aufbauend auf diesen kleinsten Strukturen bahnt sich der Weg Richtung größeren Strukturen, welche sich nicht nur in einem einzelnen Quelltext befinden, sondern mitunter auf mehrere Quelltextdateien zugreifen (müssen), die nicht zwingend denselben syntaktischen Vorgaben obliegen. Einige dieser Strukturen neben reinem Quellcode auch einzelne String-Literale, welche nicht immer übersetzt werden müssen, aber von Interesse für eine Übersetzung werden könnten. Hiernach bleiben jedoch einige spezifischere Probleme, für welche sich nicht immer eindeutige Lösungen finden lassen.

2.1.1 Struktur einzelner Beispiele

Um alle möglichen Permutationen von Problemen abzudecken, die bei einem Übersetzen von T_EX entstehen könnten, und dabei in eine übersichtliche Form zu überführen, wird durch verschiedene mögliche, denkbare Darstellungsformen gewechselt. Hierbei kann, aus einem zunächst, abstraktem Blickwinkel recht einfach nach Größe der Struktur und Zahl an möglichen Permutationen unterschieden werden, wobei weder noch eine konkrete, quantitative Angabe zulassen. Daher finden sich zunächst „kleine“ Strukturen mit „wenigen“ oder „vielen“ *erlaubten* Permutationen (in wohlmöglich entstehenden Übersetzungen) zusammen und bilden danach „Große“. Für diese abstrakten Strukturen schnell einfache Darstellungsmöglichkeiten finden, abhängig von 2-dimensionalem Platzverbrauch, bzw. in dem Minimieren dieses. So werden kleine Beispiele mit wenigen Permutationen (innerhalb eines, wie hier, Dokumentes) möglichst nebeneinander gelistet, nachdem sie zuvor kurz beschrieben waren (bspw.: <Beispiel A> wird Fehler in möglichen Übersetzungen <Beispiel B> oder <Beispiel C> produzieren, da die deutsche Grammatik oder <C> die T_EX Syntaktik verletzt, in Anwendung in z.B. ??). Bei einer höheren Zahl an Permutationen von kleinen Strukturen ist es von Interesse, diese möglichst dicht aneinander in zwei Dimensionen darzustellen, damit wohlmöglich kleine Unterschiede oder logische Muster in diesen erkennbar werden können. Diese Art der Darstellung findet sich in ?? in Anwendung und wird in Beispiel 0 vereinfacht mit Hilfe von Binärcodierung dargestellt. Hierbei soll ein Ausgangswort/Zeichenkette, welche 4 Zeichen trägt, so verändert sein,  sie genau eine 1 trägt. Aus allen 16 zulässlichen Permutationen für binäre Zeichenketten der Länge 4 (2^4) gelangt man bei 4 richtigen „Übersetzungen“ an (insofern man das Übersetzen auch als eine Form der Codierung betrachten kann). In dieser Darstellung ist zudem vorteilhaft, dass anhand der gebildeten

Richtige Übersetzungen	Fehlerhafte Übersetzungen		
1000	1111	0011	1010
0100	1110	1101	0101
0010	0111	1011	0110
0001	1100	0000	1001

Tabelle 1: 2-dimensionale Darstellung von kleinerem Quellcode (illustrativ)

„Muster“ erkenntlich wird, dass nicht alle 4 bit eines Wortes geprüft werden müssen, um eine Richtigkeit zu bestätigen, wenn man davon ausgeht, dass Zeichenketten auch von rechts nach links gelesen werden können. Diese rein logische Operation führt dazu, dass bei einem Zeitgleichen überprüfen in beide Richtungen beide „Tests“, nachdem sie sich in der Mitte trafen, im nächsten bit fertig sind. Nach dem Ausschlussverfahren müssen nämlich entweder eine *links-nach-rechts* Iteration beachten, wie viele 1 vorliegen, genauso wie die *rechts-nach-links*. Jede dieser Iterationen muss dafür den gesamten Prüfungs-Prozess abbrechen können, sobald sie mehr als nur eine 1 liest (was 7 Fehler in zwei Schritten abdeckt). Im (logischen) dritten Schritt würden beide Iterationen ihre gesammelten auf Ungleichheit prüfen. Hätten beide Iteratoren je eine oder keine 1 gelesen, wäre diese Äquivalenzrelation gebrochen, wodurch ein Fehler erkennbar wäre (denn die Abfrage schließt die übrigen 5 Fehler mit ein).¹

Ein weiterer nutzbarer Vorteil ist, dass die größeren Strukturen aus den kleineren Strukturen in T_EX bestehen (bspw. Quelltexte bestehen aus Befehlen, Funktionen, Parametern, ...), wodurch Fehler nicht bis ins Detail verfolgt werden müssen, sondern nur ihre (logischen) Auswirkungen innerhalb von Dokumenten berücksichtigt werden müssen, sowie welche Änderungen in einem Dokument in mehrere andere Dateien nachverfolgt werden müssen, um danach rückwirkend Anpassungen im eigentlichen Dokument festzustellen. Hierzu wird in der Darstellung der Quelltexte mehr räumlicher Platz beansprucht.

2.2 Elemente in einem Quelltext

2.2.1 Kommandos

2.2.1.1 Unterscheidung Kommandos sind in der reinen T_EX-Syntax durch ein Backslash \ gekennzeichnet. Die Wörter, auf welche ein \ folgt, nicht übersetzt werden dürfen, ist trivial. Die Frage, ob das Wort, das einem Kommando folgt, übersetzt werden sollte, oder nicht, zeigt jedoch erste Konflikte und ist daher nach Art des Strings zu unterscheiden. Hierbei existieren zwei Wege einen String zu interpretieren (aus logischer Perspektive). Entweder soll er als Wort (Literal) im Dokument als dieser String angezeigt werden, oder dient als anderer z.B. Parameter für eine Funktion. Beispielsweise müsste innerhalb von Auflistungen der nach einem \item folgende String übersetzt werden und steht einzeln, wodurch er leicht erkannt werden kann. Andererseits wäre in einer Definition mit \def vorerst davon abzusehen den folgenden String zu übersetzen, da dieser ein neues Makro/Kommando/etc. definiert (welche Wörter sind, welche dem Übersetzer zu „leicht“ auffallen und

¹Vorteil: $\mathcal{O}(\log n)$ statt $\mathcal{O}(n)$ zur Überprüfung einzelner Zeichenketten auf Fehler. Diese könnte unter Anderem auf Rechtschreibkorrekturen angewendet werden.

von diesem als T_EX-Semantik tragende syntaktische Elemente interpretiert werden). Demnach darf `\def\hello` nicht zu: `\def\hallo` werden (es sei denn alle folgenden `\hello` werden auch übersetzt).²Ob das auf diesen definierten Befehl folgende Wort übersetzt werden darf (und dies zu erkennen ist), wird später (siehe: ??) näher verfolgt, da es sich hier zunächst noch um eine andere Fehlerquelle drehen soll.

2.2.1.2 Erkennung Die zuvor beschriebenen String-Literale sind nicht, so wie es `\item` deuten lässt, an festen Zeichen zu erkennen. Verschiedene Befehle sind dazu in der Lage Strings als Parameter aufzugreifen, welche als Literale im Dokument erscheinen sollen. Neben den üblichen Klammern, welche diese fassen können (`{}` oder `[]`) existieren auch Wege andere Zeichen an diesen Stellen zu nutzen. Einfachstes Beispiel wäre hierfür der *wortwörtliche* `\verb`-Befehl, der alle für alle ASCII-Zeichen (in UTF8, inkl. LATIN-Extensions) erlaubt diese als eine Umklammerung von (wie gezeigt, mittels: `\verb\verb reproduzierbar`) einem Befehl zu nutzen, damit nicht dieser von T_EX's Parser als der eigentliche Befehl gedeutet wird, sondern als *genau* diese Zeichenkette. Ein Übersetzer muss demnach jegliche Permutation hierfür diesen Befehl abdecken. Hierbei ist zu vermerken, dass es auch möglich ist, Wort-Charaktere (A-Z,a-z) zu nutzen, insofern zuerst ein Space nach dem `\verb` folgt (wobei diese Zeichen dann logischerweise nicht mehr Teil der innerhalb von `\verb` stehenden Zeichenkette werden kann). Zu übersetzen sind in diesen Fällen nur Permutationen, welche mit Gewissheit ein Wort einer menschlichen Sprache sind, was sich entweder anhand fehlender Zeichen für ein Alphabet (heuristisch, da unter der Annahme, dass die Sprache selbst *immer* alle Zeichen ihrer Zeichensatzes verwendet) oder an überschüssigen Zeichen innerhalb des nachverfolgten Strings aufzeigen kann/wird.

2.2.1.3 Weitere Beispiele

2.2.2 Zusätzliche Parameter

2.2.2.1 Unterscheidung Parameter sind neben Kommandos eine weitere Möglichkeit Informationen an T_EX zu übermitteln und äußern sich entweder als String-Literale oder Teil einer bestimmten Menge an verfügbaren Optionen für ein Kommando (bspw. `\color{red}`). Genauso, wie aus einzelnen Kommandos bereits bekannt ist, sollten nur String-Literale übersetzt werden. Das wesentliche Problem liegt hierbei daran, wie sie erkennbar sind. Die übliche T_EX-Notation sieht hierbei geschwungene Klammern für zwingend erforderliche Parameter vor, und eckige Klammern für optionale Parameter. Jedoch erwarten unterschiedliche Befehle verschiedene optionale und erforderliche Parameter, für welche nicht vorhergesagt werden kann, ob sie übersetzt werden dürfen. Die logische Reihenfolge ob erwartete Parameter optional oder erforderlich sind, können einzelne Befehle selbst bestimmen.³ Aus theoretischer Sicht ist die Menge an möglichen Parametern unendlich, T_EX selbst begrenzt diese jedoch auf 9 und setzt damit ein technisches Limit für `def`. Dieses ist auf verschiedene Arten umgehbar (einzelne Parameter/Strings können mehr Informationen kodieren, Definitionen mit Relays erweitert werden oder *key-value*-Strukturen genutzt werden (bspw. in `hyperref`'s `\hypersetup` wiederzufinden)), zeigt aber nur Probleme auf, wenn wortsprachliche Strings und keine alphanumerischen Werte

²Letzteres ist allerdings nicht immer als gewährleistet zu betrachten.

³Ein Parameter eines Befehls wird optional beim Aufrufen des Kommandos, sollte ein initialer/standardmäßiger Wert bei der Deklaration mittels `,`, wobei das erste Paar hierbei die Anzahl an Parametern festlegt und jedes folgende mit einer #-Nummer korrespondiert.

innerhalb diesen genutzt werden, solche Strings also als String-Literale genutzt werden und daher übersetzt werden sollen. Konkreter muss abgewägt werden, ob es sich bei einzelnen Parametern um Werte handelt, welche für Operationen ihren exakten Wert tragen müssen (bspw. `\hypersetup{urlcolor=red}` darf weder zu `\hypersetup{URL-Farbe=rot}`, noch `\hypersetup{URL-Farbe=red}` oder `\hypersetup{urlcolor=rot}` werden) oder um Strings, welche im eigentlichen Dokument erscheinen sollen (bspw. `\paragraph{Trick or Treat}` muss zu `\paragraph{Süßes oder Saueres}` werden, darf aber, bekanntlich, nicht `\Paragraph{Süßes oder Saures}` produzieren). Aus den reinen Funktions-/Kommandoaufrufen geht allerdings nicht immer die Information hervor, ob die Übersetzung stattfinden darf. Betrachtet man z.B. die Kommandos des `theorem` Paketes (*package(s)* werden später noch eingeführt), so erfordern diese String-Literale als anzuzeigender String für eine neue nummerierte Umgebung.

Beispielsweise

```
\newtheorem{lemma}{problem}
```

würde bei jedem Aufruf von `\begin{lemma}` (bzw. inkl. `\end{lemma}`) den String `problem` gefolgt mit einer Nummer korrespondierend zu `kapitel.unterkapitel.problemNummer` produzieren. Wichtig ist hierbei, dass

```
\newtheorem{lemma}{problem}
```

nicht zu

```
\newtheorem{Lemma}{Problem}
```

im Deutschen werden darf, sondern idealerweise in

```
\newtheorem{lemma}{Problem}
```

übersetzt werden sollte.⁴ Ähnliches ist bei optionalen Parametern denkbar. So dürfte ein `\usepackage[english]{babel}` nicht zu `\usepackage[Englisch]{babel}` werden. Gegenüber zuvor erwähnten *key-value*-Paaren ist in solchen Fällen keine Zuweisung eines Schlüssels zu einem Wert anhand eines `=` erkennbar.

2.2.2.2 Erkennung Beschränkt man die Anzahl an Parametern (optional und erforderlich) zunächst auf 3, gelangt man bei $2 * 2^3 = 64$ möglichen Permutationen an (ein Parameter kann entweder optional oder erforderlich sein, dessen Inhalte dürfen entweder Übersetzt werden oder nicht und es existieren, bekanntlich, drei Parameter. Die Zahlen 1–4 in beliebiger Reihenfolge in einem 3-elementigen Array anordnen.). Verschiedene zuvor beschriebene Permutationen sind beispielhaft in unterem Beispiel, in welchem zur einfacheren Lesbarkeit die Kürzel *t* für übersetzbare Felder (translatable string) und *p* für nicht zu übersetzende Strings (parameter) stehen.

Bereits eine solche kleine Anzahl an Parametern macht erkenntlich, dass jeder dieser einzeln ausgewertet werden muss und so weit ins Backend der Software verfolgt werden muss, bis klar ist, dass mit diesem String keine logische Operation vorgenommen wird. Die maximale Anzahl an Parametern für jedes einzelne Kommando beträgt 9, was die Anzahl an möglichen Permutationen auf circa 250 Tausend erhöht.

⁴„Lemma“ statt „Problem“ in diesem Kontext zu produzieren, könnte man als eine Art *Bonus* interpretieren, in welchem der Kontext erkannt wurde und ein „üblicheres“ Wort im Deutschen gewählt wurde.

Übersetzbar	Risikiert Fehler
<code>\cmd {t}{t}{\cmd {t}{t}{\cmd {t}{p}{\cmd {t}{p}{\cmd {p}{t}{\cmd {p}{t}{\cmd {p}{p}{\cmd {p}{p}{p}</code>	
<code>\cmd {t}{t}{\cmd {t}{t}{\cmd {t}{p}{\cmd {t}{p}{\cmd {p}{t}{\cmd {p}{t}{\cmd {p}{p}{\cmd {p}{p}[p]</code>	
<code>\cmd {t}{t}{\cmd {t}{t}{\cmd {t}{p}{\cmd {t}{p}{\cmd {p}{t}{\cmd {p}{t}{\cmd {p}{p}{\cmd {p}{p}{p}</code>	
<code>\cmd {t}{t}{\cmd {t}{t}{\cmd {t}{p}{\cmd {t}{p}{\cmd {p}{t}{\cmd {p}{t}{\cmd {p}{p}{\cmd {p}{p}[p]</code>	
<code>\cmd {t}{t}{\cmd {t}{t}{\cmd {t}{p}{\cmd {t}{p}{\cmd {p}{t}{\cmd {p}{t}{\cmd {p}{p}{\cmd {p}{p}[p]</code>	
<code>\cmd {t}{t}{\cmd {t}{t}{\cmd {t}{p}{\cmd {t}{p}{\cmd {p}{t}{\cmd {p}{t}{\cmd {p}{p}{\cmd {p}{p}[p]</code>	
<code>\cmd {t}{t}{\cmd {t}{t}{\cmd {t}{p}{\cmd {t}{p}{\cmd {p}{t}{\cmd {p}{t}{\cmd {p}{p}{\cmd {p}{p}[p]</code>	
<code>\cmd {t}{t}{\cmd {t}{t}{\cmd {t}{p}{\cmd {t}{p}{\cmd {p}{t}{\cmd {p}{t}{\cmd {p}{p}{\cmd {p}{p}[p]</code>	
<code>\cmd {t}{t}{\cmd {t}{t}{\cmd {t}{p}{\cmd {t}{p}{\cmd {p}{t}{\cmd {p}{t}{\cmd {p}{p}{\cmd {p}{p}[p]</code>	

Tabelle 2: 2-dimensionale Darstellung möglicher Permutationen von Kommandos mit 3 Parametern

2.2.2.2.1 Problem bei diesem Ansatz zeigt sich darin, wenn man versuchen würde alle nach einem Kommando stehenden Klammer-Paare ggb. diesem Kommando zurückzuverfolgen. Dieser Ansatz vergisst, dass geschwungene Klammern auch dazu dienen könnten, dass Umgebungen geöffnet und geschlossen werden.

2.2.3 Makros und Umgebungen

2.2.3.1 Umgebungen \TeX trägt ein Konzept von „Umgebungen“, für deren Inhalte (ob textlich oder mittels parameterisierter Kommandos produzierte) dieselben Befehle innerhalb dieser prozedural angewendet werden. So könnte man die Farbe der ausgegebenen Zeichenkette mittels `{\color{red} text}` (`text`) zu rot wechseln. Solche einfachen Umgebungen selbst produzieren noch keine neuen Fehler, allerdings existieren spezifische, teils vordefinierte Umgebungen, innerhalb welcher von Übersetzungen abzusehen ist (bzw. zu übersetzende Inhalte nur anhand bestimmter Elemente erkannt werden können).

Mathematische Einfachstes Beispiel hierfür sind mathematische Umgebungen, welche z.B. Formeln, griechische Buchstaben und sonstige mathematische Sonderzeichen darstellen können, jedoch auch an vorher nicht unbedingt bekannten Stellen textliche Inhalte tragen können (siehe: unten). Ein Wechsel zwischen solchen Umgebungen ist theoretisch unendlich oft möglich, solange auf jedes Betreten ein Verlassen (der Umgebung) erfolgt. Hierbei wäre es ideal, wenn alle textuellen Inhalte (des Quellcodes) übersetzt werden und alle Mathematischen nicht (im Beispiel: *reminder*: zu *Erinnerung*: und *frequency* zu *Frequenz*).

```

\begin{align*}\label{problems:tables:eq}
  \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{ reminder: } \$\omega = 2\pi \text{ \text{}}
\end{align*}

```

Insbesondere müssen Fälle betrachtet werden, in welchen Sätze (oder teils Paragraphen) durch mathematische Inhalte unterbrochen/segmentiert werden. So sollte bspw. The below equation $x=4+x$ is false. zu Die untenstehende Gleichung $x=4+x$ ist falsch. werden und nicht als zwei einzelne Sätze aufgegriffen werden (Die untere Gleichung. $x=4+x$ Ist falsch.). Andererseits darf nicht nur, weil eine mathematische Umgebung betreten wurde, auf einmal Texte innerhalb dieser vergessen werden oder erwartet sein, dass die mathematischen Inhalte nicht selbst menschen sprachliche Worte tragen. Beispielsweise sollte $a \neq b \wedge a \text{ isn't } b \text{ and } a$ in $a \neq b \wedge a \text{ ist nicht } b \text{ und } a$ resultieren, aber $\text{cars} = 3, \text{ value} = 20 \text{ ct}$ die Variablen unverändert lassen, dementsprechend nicht $\text{Autos} = 3, \text{ Wert} = 20 \text{ ct}$ produzieren. Hierbei sind höhere Verschachtelungen (bzw. Vernestungen) zu prüfen.

Captions und Table of Contents In ähnlicher Art und Weise wie mathematische Umgebungen können auch Tabellen syntaktische Elemente, Umgebungen und textliche Inhalte miteinander vermischen. Neben recht einfach erkennbaren Unterschieden ist es ebenso denkbar, dass vereinzelt Umgebungen innerhalb einzelner Tabellen-Zellen genutzt werden, welche ihrerseits entweder Wörter enthalten, welche nicht übersetzt werden dürfen oder auf String-Literale zugreifen, die an anderer Stelle stehen. Bei der Übersetzung in eine andere Sprache als Englisch muss hierbei natürlich bedacht werden, dass vordefinierte Untertitel bzw. Beschreibungen angepasst werden, damit z.B. eine Tabelle oder Abbildung nicht als *table* oder *figure*, sondern als *Tabelle* oder *Abbildung* bezeichnet wird. Gleiches muss für einzelne Abschnitte, wie zum Beispiel die Überschriften des Inhalts-, Abbildungs- und Tabellenverzeichnisses, als auch einer Nomenklatur oder eines Glossares gewährleistet sein und beinhaltet Änderungen, welche idealerweise in die Präambel des Dokumentes aufgenommen werden.

Verbatim Ähnlich wie bei der Unterbrechung von einzelnen Sätzen durch mathematische Formeln können in T_EX auch verschiedene Wege genutzt werden, wortwörtliche Zitate zu verwenden um Sätze oder Paragraphen zu unterbrechen. Neben dem bekannten Befehl `\verb` existiert auch eine großflächigere *verbatim*-Umgebung. Denkbar ist ihre Nutzung um kleinere T_EX-Quellcodes⁵ darzustellen. Innerhalb dieser Bereiche sind selbstverständlich alle bisher bekannten Fehlerquellen ihrerseits zu umgehen. Wie verhält sich jedoch ein Übersetzer, sollten Teile dieser *verbatim*-Umgebungen zum Wortlaut des formulierten Satzes beitragen? Soll aus `I detest using the \verb|\verb command| unnecessarily.` eher Ich lehne ein unnötiges Nutzen des `\verb|\verb` oder Ich lehne ein unnötiges Nutzen des `\verb|\verb Kommando|` ab. werden? Denkbar sind, in diesem Kontext, beide Alternativen, allerdings müssen größere Beispiele die Inhalte der wortwörtlichen Zitierung näher auswerten und als T_EX-Quellcode behandeln.

Mit eigener Syntax Umgebungen können mitunter ihre eigenen syntaktischen Forderungen stellen, für welche sie beispielsweise englische Begriffe nutzen, wie z.B. *TikZ*. In solchen Umgebungen muss wieder ausgewertet werden, wann einzelne Zeichenketten übersetzt werden dürfen, denn z.B. *nodes* erlauben es, dass Texte an diesen platziert werden, aber die Erstellung dieser erfordert den englischen Begriff *at* um ihrer Position innerhalb einer Graphik eine Koordinate zu geben. `\node at (1,1) {hello}` muss in `\node at (1,1) {Hallo}` resultieren und darf nicht zu `\node bei (1,1) {Hallo}` werden.

⁵Quelltexte anderer Programmiersprachen werden später betrachtet.

Kommentare Wohingegen einzelige Kommentare in T_EX mit einem `\%` gekennzeichnet werden, sind auch Mehrzeilige erlaubt, oder solche, die innerhalb ein- und derselben Zeile starten und enden. Hierbei müssen diese Kommentare missachtet werden. So muss ein `This statement is \begin{comment}not\end{comment} true`. nicht zwingend in `Diese Aussage ist \begin{comment}nicht\end{comment} wahr`. enden, darf aber unter keinen Umständen `Diese Aussage ist falsch`. produzieren, innerhalb welcher der Kommentar interpretiert wurde und die Ausgabe kommentarlos angepasst wurde.

Makro Erneuerungen Ähnlich wie Kommandos sind Makros eine Art und Weise kürzere Befehle zu nutzen um Ausgaben zu produzieren.

2.3 Mehrere Quelltexte

2.3.1 T_EX und Andere

Nicht alle Informationen, welche benötigt sind, um ein Dokument zu kompilieren, liegen zwingend in einem Quelltext vor. Insbesondere größere Dokumente, wie bspw. Bücher können davon profitieren ihre Texte auf mehrere `.tex`-Dateien zu verbreiten und ausgehend von einem ursprünglichen Dokument diese per `\include` oder `\input` mit einzubinden. Die Inhalte dieser Dateien sollten hierbei als weitere Teile des vorliegenden Quelltextes (für eine Übersetzung) zu sehen sein und demnach nicht übersehen werden.

Neben der Verwendung mehrerer T_EX-Quelltextdateien kann es ebenso von Interesse sein, dass verschiedene Quelltexte (verschiedener Programmiersprachen) in einem Dokument darstellen möchte. Pakete wie `lstlisting` oder `minted` erlauben dies auf verschiedene Wege. Quelltexte (quasi-) beliebiger Sprachen können entweder innerhalb entsprechender Umgebungen eingefügt werden oder es kann ein URL zur Datei-Adresse an passender Stelle angegeben sein. Hierbei sind nicht nur syntaktisch relevante Wörter von bekannten Sprachen zu erhalten, sondern auch z.B. in diesen Quelltexten vorhandene Kommentare, welche die Funktionsweise des Codes erklären oder auszugebende Strings (falls es sich um Codefetzen mit String-Ausgaben handelt) für einen Übersetzer in Betracht zu ziehen.⁶

2.3.2 Pakete und Klassen

Bereits einige vorherige Beispiele zogen sog. „Pakete“ zur Erklärung der potentiellen Fehlerquellen heran. Mit Hilfe dieser können einem T_EX-User (die nicht zu verwechseln mit späteren „Endnutzern“ sind, bei welchen es sich um die Leser des entgeltigen, kompilierten Dokumentes handelt) zusätzliche Umgebungen, Kommandos oder Makros zu Verfügung gestellt werden. Diese Funktionen der Pakete sind bekannterweise (genauso, wie vorangegangene Kommandos, Umgebungen, etc.) dazu in der Lage Strings zu tragen, welche möglicherweise im Dokument landen könnten. Zwar technisch eigentlich unterschiedlich, sind Klassen trotzdem gleich behandelbar, da auch in diesen nur Makro-Erstellungen, -Erneuerungen und Umgebungen nachvollzogen werden müssen und deren textliche Inhalte, welche im Dokument landen werden, übersetzt werden sollen.

⁶Wobei hier die Übersetzung eingebundener T_EX-Codes insbesondere von Interesse ist, da diese überwiegend textliche Inhalte tragen könnten.

2.3.3 Hilfsdateien

Einige typische Elemente von Dokumenten erfordern es aus logischer Sicht, dass einzelne Informationen in eine Hilfsdatei geschrieben werden müssen und erst beim einem erneuten Kompilieren eine richtige, vorgesehene Ausgabe produzieren können. Hierbei sind vielerlei Möglichkeiten denkbar, daher wird zunächst mit einer, den meisten T_EX-Usern vertrautesten (Quelltext-), Datei gestartet und zwar einer `.bib`, welche für die Literaturverwaltung genutzt werden kann. Innerhalb dieser existieren verschiedene Felder, welche für eine Übersetzung interessant werden könnten, beispielsweise Titel oder Abstrakt einzelner zitierter Werke, allerdings auch Notizen oder sonstige Anpassungen, wie zum Beispiel ein Vermerk „verfügbar unter: <URL>“ im entsprechenden Feld eines BibT_EX assoziierten Paketes.⁷ Bei der Nutzung dieser Technologie (BibT_EX) bemerkt ein T_EX-User direkt, dass aus logischer Perspektive mehrfach kompiliert werden muss, da zunächst ein T_EX-Compiler alle Zitationen im Quellcode sehen muss, diese Information dann über die `.aux` übermittelt, auf welche BibT_EX zugreift. Nachdem BibT_EX dann diese Datei angepasst hat, kann der T_EX-Compiler im nächsten Lauf die Zitationen an richtiger Stelle einfügen und am Ende entsprechend im Literaturverzeichnis vermerken. Die gleiche Denkweise lässt sich auf andere Hilfsdateien anwenden. Bei einem Übersetzen von z.B. Kapiteltiteln reicht es nicht aus nur `\chapter{Incredible}` zu `\chapter{Unglaublich}` zu übersetzen, sondern genauso muss neu kompiliert werden (einfach, wenn die `.toc` angepasst wurde, ansonsten zweifach). Genauso verhält es sich mit Verzeichnissen anderer Art (bspw. Abbildungen, Tabellen, etc.) oder den Inhalten von Glossaren oder Nomenklaturen, sollten sich diese in anderen Dateien befinden.⁸

2.3.4 Zitationen

Hierbei handelt es sich nicht um ein direktes, technisches Problem von übersetzenden Programmen, aber um ein Implizites. Viele Wörter benötigen in bestimmten Kontexten eine spezifische Übersetzung. Diese Kontexte müssen in L^AT_EX nicht nur aus den direkten Sätzen hervorgehen, sondern können auch hinter Verweisen/Referenzen oder Zitationen an anderer Stelle im Quelltext verborgen stehen. Zitiert man beispielsweise ein Originalwerk vom französischen Mathematiker Évariste Galois und verwendet danach das Wort *fields* („ ? | defines fields as ...“) so sollte dieses nicht einfach zum Deutschen *Felder* werden, sondern man würde das Wort *Körper* erwarten (von „endlichen Körpern“, englisch: *finite fields* oder *galois fields*). Fraglich ist hierbei, wie weit ein solcher Kontext nachverfolgt wird. Liegt er direkt in einem vorigen Teil des Quelltextes vor und mittels `\ref` ein mit passendem *key* ein entsprechendes `\label` verlinkt, so gibt es wenig Gründe, die für einen Verlust des Kontextes sprächen. Ähnlich zu betrachten ist eine (Hyper-) Referenz auf eine andere, lokale Quelltextdatei. Interessant könnten auch BibT_EX Zitationen in diesem Kontext werden, da bei diesen verschiedene, theoretisch ausreichende Verweise auf Informationsquellen angegeben werden können, denn ein solcher Literatureintrag muss nicht zwingend vollständig ausgefüllt werden. Es könnte der Fall eintreten, dass nur eine URL oder DOI angegeben wird, eine ISBN (zu mitunter nicht allgemein zugängigen Werken) oder ISSN. Zu erwarten wäre, dass aus solchen Angaben genau so viel Kontext entnommen wird, wie es auch ein Mensch könnte. Dieses Konzept wäre per se auch auf jegliche URL (meist) hinter Sätzen/Paragraphen anwendbar, da dies eine recht schnelle und einfache

⁷BibLaT_EX oder NatBib

⁸Obwohl dies theoretisch gesehen schon unter „T_EX“ und andere behandelt sein sollte, wäre eine Auswirkung auf insb. Glossare interessant und inwiefern Abkürzungen angepasst werden. Eine deutsche WDF (Wahrscheinlichkeitsdichtefunktion) wäre im Englischen als PDF (probability density function) zu erwarten.

Art und Weise ist, mit welcher sich nötigste Quellangaben liefern ließen, wird allerdings aufgrund mangelnder Professionalität solcher Zitationsweisen nicht weiter untersucht.

2.4 Spezifisch

2.4.1 Sprachliche Hürden

2.4.2 Layouting-Probleme

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

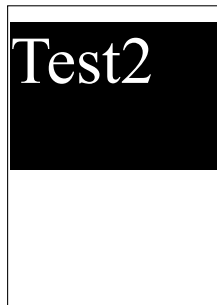
A.1 Fontskalierung auf Webseiten

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

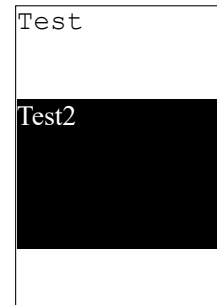
```
<html>
  <head>
    <title>Example</title>
    <style>
      /*formatting options are: none and black*/
      .t{
        font-size:13em;
        height:50%;
      }
      /*formatting option: none = no background, black, courier*/
      .t#none{
        font-family: 'Courier New', Courier, monospace;
      }
      /*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