Institut für maschinelle Sprachverarbeitung
Universität Stuttgart
Pfaffenwaldring 5b
D–70569 Stuttgart

Diplomarbeit Nr. 3301

Konzept und Entwicklung eines Werkzeugs zur automatisierten Übersetzung natürlichsprachlicher Anforderungen

Nadine Siegmund

Studiengang:	Informatik
Prüfer:	Prof. Dr. Hinrich Schütze
Betreuer:	Dipl.-Ling. Nadya Stoyanova Dipl.-Ling. Boris Haselbach
begonnen am:	16. Januar 2012
beendet am:	12. Juli 2012
CR-Klassifikation:	I.2.7, D.2.1, D.3.4, F.4.2

Zusammenfassung

Zum erfolgreichen Projektabschluss gehört im automobilen Umfeld die Erstellung einer Anforderungsdokumentation. Diese sollte wegen der immer stärkeren internationalen Zusammenarbeit und verteilten Entwicklung am besten in mehreren Sprachen und vor allem in Englisch zur Verfügung stehen. Da aber nicht jeder Projektbeteiligte aller Sprachen mächtig ist, muss das Anforderungsdokument übersetzt werden. Die maschinelle Übersetzung bietet sich hierfür an, da dadurch standardisierte Anforderungsdokumente erzeugt werden können und im Gegenteil zu einem Übersetzer die Übersetzung günstig ist und ohne Verzögerung zur Verfügung steht. In dieser Arbeit wird das Konzept und die Implementierung einer Methodik, die in einem Werkzeug umgesetzt wird, vorgestellt. Damit können Anforderungen mittels maschineller Übersetzung von Deutsch nach Englisch übersetzt werden.
Es wird dazu ein transferbasiertes Verfahren angewandt, das einen Satz mittels einer mit Merkmalsstrukturen und Restriktionen angereicherten Phrasenstrukturgrammatik analysiert und in eine Strukturbeschreibung überführt. Die Strukturbeschreibung wird mit Hilfe von Transferregeln vom Deutschen ins Englische überführt und daraus der übersetzte Satz generiert. Das Werkzeug wurde, zum Beispiel durch die Verwendung von XML, an möglichst vielen Stellen erweiterbar gestaltet, um eine einfache Weiterentwicklung durch Experten, wie zum Beispiel Computerlinguisten, zu ermöglichen.
Zur Vereinfachung der Problematik einer maschinellen Übersetzung wird die Tatsache ausgenutzt, dass die Anforderungen mittels einer Schablone formuliert werden und dadurch die Satzstruktur stark eingeschränkt wird. Außerdem liegt ein Lexikon mit einer Subsprache vor, durch das weitere Probleme eliminiert werden können.
Diese Arbeit zeigt, dass die maschinelle Übersetzung unter den gegebenen Voraussetzungen ein geeigneter Ansatz für die Übersetzung von Anforderungen ist.

Abstract

In the automotive industry, a requirement specification is mandatory, in order to successfully manage a project. Due to the increasing international collaboration between OEMs and suppliers as well as more decentralized development requirement specifications are required in various languages and therefore need to be translated. This could be achieved by a human translator or a translation system. The focus of this work is on a standardized and cheap translation without time delay which could be achieved better by a machine translation system.
In this thesis the concept and implementation of a method is introduced which could translate requirements from German to English with machine translation. The method first analyses each sentence with a phrase structure grammar which includes attribute structures and restrictions. The result of this procedure is a syntax tree that afterwards is transferred by transfer rules into a tree build by the English syntax. The final translation of the requirement can therefore easily be generated. To enable the expand of the range of functionalities through experts like computational linguistics most parts of the tool are designed extendible for example through the use of XML.
For an easier translation the requirements have to be written according to a template and a dictionary with a sublanguage is used.
The proposed methods are a good solution for an automated translation of requirements and to be preferred over human translation.

Danksagung

Ich möchte mich mich bei allen bedanken, die mich während meines Studiums und ins besondere während meiner Diplomarbeit unterstützt haben.

Im Speziellen geht mein Dank an Herrn Prof. Dr. Hinrich Schütze für das Ermöglichen und Prüfen dieser Diplomarbeit.

Meine Diplomarbeit ist in Zusammenarbeit mit der Daimler AG am Standort Böblingen/Hulb entstanden. Hier bedanke ich mich besonders bei meiner Betreuerin Nadya Stoyanova für die vielseitige und vor allem fachliche Unterstützung, die Motivation und die beruhigenden Worte. Bei meinem Teamleiter Dr. Matthias Recknagel und den Mitarbeitern der Abteilung RD/EST bedanke ich mich für die angenehme Zeit.

Auch von Seiten der Universität wurde ich bestens betreut. Meinem Betreuer Boris Haselbach möchte ich danken für die kritische Betrachtung meiner Arbeit und sein großes Engagement. Ebenso gilt mein Dank Herrn Prof. Dr. phil. habil. Grzegorz Dogil, der mich während meines Studiums stets unterstützt hat.

Zum Schluss möchte ich mich noch bei Thomas Walter und Severa Märker bedanken für die viele Geduld, die sie mit mir und beim Durchlesen meiner Arbeit hatten. Mein Dank gilt ebenso meiner Familie, die mich stets in allen Lebenslagen begleitet und bestärkt hat.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

1.1. Motivation

Zur erfolgreichen Entwicklung eines Produkts oder eines Systems sind viele Prozesse notwendig. Zu den ersten Schritten gehört die Anforderungsanalyse, in der unter anderem die Wünsche und Anforderungen der Auftraggeber erfasst werden. Der Erfolg des Projekts ist direkt abhängig von der Qualität, mit welcher diese Anforderungen aufgenommen werden. Wird eine Anforderung beispielsweise zu ungenau formuliert, kann es zu Missverständnissen kommen oder es müssen Vermutungen über die genaue Funktionalität angestellt werden.
In der Regel arbeiten an einem Projekt viele verschiedene Menschen. Es treffen verschiedene Perspektiven und unterschiedliche Kompetenzen aufeinander. Aber alle entwickeln an dem Projekt anhand des Anforderungsdokuments. Das Anforderungsdokument hat einen sehr hohen Stellenwert und die Erstellung gehört zu einer der schwierigsten und aufwändigsten Aufgaben in der Entwicklung.
Requirements Engineering befasst sich genau mit dieser Problematik. Es werden Strategien, Werkzeuge und Richtlinien entwickelt und zur Verfügung gestellt, um die Anforderungen eines Projekts detailgenau, vollständig und einheitlich zu dokumentieren. Dadurch können viele Fehler vermieden werden, durch die ein Projekt im schlimmsten Fall scheitert. Laut Rupp lassen sich ca. 60% der Fehler während der Systementwicklung auf den Analyseprozess zurückführen [Rup09, S. 15].
Ingenieuren, Technikern und anderen wird das Schreiben von Anforderungen durch diese Hilfsmittel zunehmend erleichtert. Meistens allerdings nur in der kommunizierten Sprache und dies ist in Deutschland hauptsächlich Deutsch. Mittlerweile werden Anforderungen aber vorallem in Englisch oder wegen der immer häufigeren internationalen Zusammenarbeit auch in anderen Sprachen verlangt. Nicht jeder ist in der Lage eine Anforderung in der gewünschten Sprache zu formulieren und Schulungen kosten ein Unternehmen sowohl Geld als auch Zeit. Eine Lösung bietet die maschinelle Übersetzung: geschriebene Anforderungen können von einer Quellsprache in eine Zielsprache überführt werden, so dass eine standardisierte Übersetzung entsteht. Dies könnten natürlich auch Übersetzer leisten aber auch hier sind es die Kosten, wie zum Beispiel Gehaltskosten oder Kosten, die durch Verzögerungen entstehen, die Unternehmen nach anderen Lösungen suchen lassen. Der Bedarf an Übersetzern kann wegen dem ständig wachsendem Produktmarkt nicht gestillt werden. Nahezu jedes Produkt hat ein Handbuch in verschiedenen Sprachen. Die Produktzahlen steigen und die Produktionszeit sinkt [Ram09, S. 12/13].

Die maschinelle Übersetzung wird seit 1947 in der wissenschaftlichen Literatur diskutiert und wurde oft als unerreichbar eingeschätzt [Ram09, S. 57]. Sie ist sehr interdisziplinär aufgestellt, denn sowohl die Informatik, die Künstliche Intelligenz, die Linguistik als auch die Computerlinguistik tragen zu ihrer Entwicklung bei [Ram09, S. 13]. Je nach gewähltem Verfahren verlangt das Thema gerade von der Computerlinguistik bzw. der Linguistik einiges an Wissen. Denn außer der Produktion und dem Verstehen von Sprache sind alle Bereiche der Sprachverarbeitung gefordert [HS92, S. 3].
Die meisten Anwender kennen die maschinelle Übersetzung durch Dienste aus dem Internet, wie zum Beispiel den Übersetzungsdienst von Google[1]. Das Urteil über die Qualität der Übersetzung fällt dabei oft nicht besonders positiv aus. Dies mag daran liegen, dass die maschinelle Übersetzung nicht für jeden Zweck geeignet ist. Für eine Übersetzung ist spezifisches Wissen notwendig. Deswegen können domänenspezifische Systeme Fachtexte gut übersetzen aber kreative Texte, wie zum Beispiel ein Gedicht oder umgangsprachliche Texte, dagegen eher nicht [Hei04].
Das Empfinden bezüglich der Qualität einer Übersetzung ist teilweise subjektiv. Zur objektiveren Bewertung können aber Aspekte, wie Inhaltstreue oder Fehlerhäufigkeit herangezogen werden [Ram09, S. 158]. Zum aktuellen Forschungszeitpunkt sollte von einem Übersetzungssystem nicht mehr Leistung erwartet werden, wie von einem menschlichen Übersetzer. Ein Mensch verfügt über Weltwissen und kann Zusammenhänge im Text erschließen. Dies ist bei einer maschinellen Übersetzung im Moment nur eingeschränkt möglich. Auch professionelle Übersetzer lassen ihre Texte Korrektur lesen. Dies sollte für ein besseres Ergebnis auch einem maschinellen Übersetzungssystem zugestanden werden [HS92, S. 3].
Da Anforderungen domänenspezifisch betrachtet werden können, ist die maschinelle Übersetzung ein erfolgsversprechender Ansatz. Eine Vereinfachung der Problematik liefert zudem der Sachverhalt, dass Anforderungen mittels einer Schablone für die Satzstellung geschrieben werden können. Dies bedeutet, dass zulässige Satzstrukturen vorgegeben werden. Die maschinelle Übersetzung wird dadurch erleichtert, da sowohl der Wortschatz als auch der Satzbau stark eingeschränkt sind.

1.2. Zielsetzung

Ziel dieser Arbeit ist es, ein Werkzeug zu entwickeln, das Anforderungen, die in Deutsch mittels einer Schablone geschrieben wurden, durch maschinelle Übersetzung ins Englische überführt. Dazu muss zunächst ein geeignetes Verfahren unter Berücksichtigung der Voraussetzungen ausgewählt und ein Konzept zur Umsetzung entwickelt werden. Hoedoro hat im Rahmen einer Diplomarbeit ein Werkzeug entwickelt, welches das Schreiben von Anforderungen durch die Verwendung von Schablonen erleichtert [Hoe11]. In dieses Werkzeug soll das Ergebnis dieser Arbeit integriert werden. Ein Teil der Arbeit war die Entwicklung einer Grammatik für Anforderungen im Deutschen gemäß der Schablonen.

[1] http://translate.google.de/

Wenn möglich soll diese in den Prozess der maschinellen Übersetzung integriert werden. Zusätzlich soll eine äquivalente Grammatik für das Englische entworfen werden.
Das zu entwickelnde Konzept soll eine Vorstellung ermöglichen, wie die Implementierung des Systems aussehen muss. Anschließend soll das Werkzeug in Java nach diesem Konzept umgesetzt werden. Zum Schluss wird das Ergebnis der Arbeitet kritisch beleuchtet um Verbesserungs-, so wie Erweiterungsmöglichkeiten, aufzuzeigen.

1.3. Gliederung der Arbeit

In **Kapitel 2** werden die notwendigen Grundlagen vorgestellt, auf die sich das Konzept stützen soll. Der erste Teil der Grundlagen widmet sich dem Requirements Engineering, dies soll der Einordnung der Arbeit dienen. Im zweiten Teil werden die Grundlagen der maschinellen Übersetzung behandelt, verschiedene Verfahren vorgestellt und auf die Probleme der maschinellen Übersetzung eingegangen. Der letzte Teil des Grundlagenkapitels behandelt das Parsing und geht genauer auf Chart-Parser ein.
In **Kapitel 3** wird diskutiert, welche Systeme auf dem aktuellen Übersetzungsmarkt existieren und ob es eine Möglichkeit gibt, diese für die Übersetzung von Anforderungen zu benutzen.
In **Kapitel 4** wird das entwickelte Konzept für das geplante Werkzeug beschrieben. Teil dieses Kapitels sind die Auswahl eines Übersetungsverfahrens, die Grammatiken für Anforderungen im Deutschen als auch im Englischen und die Beschreibung der einzelnen Schritte, die für den Übersetzungsprozess notwendig sind.
Kapitel 5 beschreibt die Implementierung des Werkzeugs und geht auf einige Aspekte der praktischen Umsetzung des Konzepts ein.
Den Abschluss bildet **Kapitel 6** mit einer Zusammenfassung, einem kritischen Fazit und einem Ausblick. Es werden Verbesserungsmöglichkeiten und Ideen zur Erweiterung angesprochen.

2. Grundlagen

Das Grundlagenkapitel soll die nötige Basis für diese Arbeit schaffen und behandelt deswegen alle relevanten Bereiche. Das Unterkapitel 2.1 betont die Position des Requirements Engineerings und enthält grundlegende Begriffe und Definitionen. Genauer wird auf die für diese Arbeit sehr wichtigen Anforderungsschablonen eingegangen. Abschnitt 2.2 behandelt wichtige Begriffe der maschinellen Übersetzung und stellt verschiedene Verfahren vor. Um Anforderungsschablonen übersetzen zu können werden diese mittels einer Grammatik beschrieben. Die Grundlagen hierfür werden in Abschnitt 2.3 erläutert. Bei der Verarbeitung von Sätzen ist das Parsen meist unumgänglich. Die relevanten Begriffe und Verfahren hierzu werden in Abschnitt 2.4 behandelt.
Für dieses Kapitel und generell für diese Arbeit werden unter anderem Grundkenntnisse in der Mengenlehre und in höherer Mathematik vorausgesetzt. Außerdem sollte der Leser über Basiswissen aus der Graphentheorie und bezüglich XML verfügen.

2.1. Requirements Engineering

2.1.1. Grundlagen des Requirements Engineering

Software Engineering befasst sich mit der Entwicklung zuverlässiger und qualitativer Software. Aber trotz großer Bemühungen ist das Entwickeln von softwarelastigen Systemen immer noch eine Herausforderung. Die meisten Probleme sind dabei auf eine unzureichende Planung zurückzuführen, genauer sogar auf mangelndes Requirements Engineering, eine Teildisziplin des Software Engineerings [Par10, S. 2–5].
Zur Planung von Projekten werden Vorgehensmodelle genutzt. Diese bieten einen Überblick über den generellen organisatorischen Ablauf und können individuell an das eigene Projekt angepasst werden. Das derzeit meist benutzte Modell ist das V-Modell (siehe Abbildung 2.1). Das V-Modell kann in zwei Teile eingeteilt werden. Einen konstruktiven Teil und einen verifizierenden Teil. Im konstruktiven Teil werden die Requirements erfasst und das System entworfen. Anschließend folgt die Implementierung und der verifizierende Teil in dem das System getestet wird [Gol11, S. 89]. Das V-Modell basiert auf der schrittweisen Zerlegung des Systems in Teilsysteme. Die einzelnen Bestandteile des V-Modells an das Projekt anzupassen wird Tailoring genannt [Rup09, S. 37].

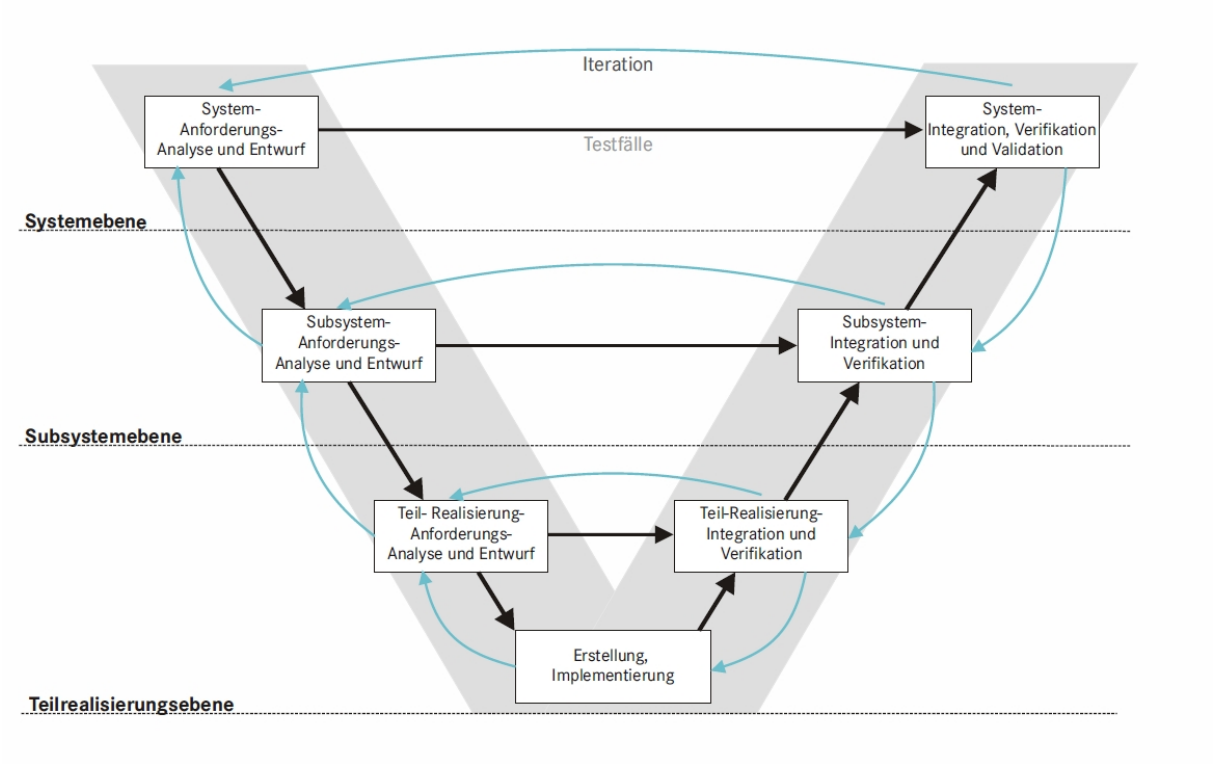

Abbildung 2.1.: V-Modell nach [Rei09]

Requirements Engineering findet je nach Strukturierung des Projekts auf mehreren Ebenen statt. Die Probleme, die aus unzureichender Sorgfalt in diesem Bereich resultieren sind jeweils die gleichen. Projekte scheitern oder es entstehen hohe Kosten durch das nachträgliche Beseitigen von Anforderungsfehlern. Dies liegt häufig am mangelnden Einbeziehen der Benutzer, unvollständigen Anforderungen, unklaren Zielen oder einer unrealistischen Erwartung [Par10, S. 6]. Abhilfe schafft das Requirements Engineering, das die Aufgabe hat, alle relevanten Anforderungen im erforderlichen Detailierungsgrad zu ermitteln, zu dokumentieren, zu prüfen und zu verwalten. Die Anforderungsspezifikation bietet dann unter anderem eine Basis für die Kommunikation zwischen allen Beteiligten und eine Basis für anschließende Tests [Rup09, S. 14].

Der Begriff „Anforderung“ wird im IEEE Standard 610.12-1990, dem „Glossary of Software Engineering Terminology“ [IEE90], definiert (siehe Abbildung 2.2).

Definition: Anforderung nach IEEE

(Übersetzung von Rupp [Rup09, S. 13])

Eine Anforderung ist...

1. eine Bedingung oder Fähigkeit, die von einem Benutzer (Person oder System) zur Lösung eines Problems oder zur Erreichung eines Ziels benötigt wird.
2. eine Bedingung oder Fähigkeit, die ein System oder Teilsystem erfüllen oder besitzen muss, um einen Vertrag, eine Norm, eine Spezifikation oder andere, formell vorgegebene Dokumente zu erfüllen.
3. eine dokumentierte Repräsentation einer Bedingung oder Eigenschaft gemäß (1) oder (2).

Abbildung 2.2.: Definition „Anforderung“ nach IEEE [IEE90]

Unter einer „Anforderung“ wird häufig aber nur ein Teil dieser Definiton verstanden. Der Begriff „Anforderung“ wird auch in dieser Arbeit nach der Definition von Rupp aufgefasst (siehe Abbildung 2.3).

Definition: Anforderung nach Rupp

Eine Anforderung ist eine Aussage über eine Eigenschaft oder Leistung eines Produktes, eines Prozesses oder der am Prozess beteiligten Personen.

Abbildung 2.3.: Definition „Anforderung“ nach Rupp [Rup09, S. 14]

Anforderungen lassen sich in funktionale und nicht-funktionale Anforderungen einteilen. Eine funktionale Anforderung beschreibt eine selbstständige Systemaktivität, eine Benutzerinteraktion, eine Interaktion zu einem anderen System oder eine allgemeine funktionale Vereinbarung bzw. Einschränkung. Eine Anforderung die nach gerade beschriebenen Eigenschaften keine funktionale Anforderung ist, ist eine nicht-funktionale Anforderung [Rup09, S. 18].

Um qualitative hochwertige Anforderungen schreiben zu können, werden verschiedene Qualitätskriterien gefordert. Rupp hat in [Rup09] solche Kriterien aufgestellt (siehe Abbildung 2.4).

Qualitätskriterien nach Rupp

- Vollständig
- Korrekt
- Abgestimmt
- Klassifizierbar
- Konsistent
- Prüfbar
- Eindeutig
- Verständlich
- Gültig und aktuell
- Realisierbar
- Notwendig
- Verfolgbar
- Bewertet

Abbildung 2.4.: Qualitätskriterien nach Rupp [Rup09, S. 24]

In der Praxis sind diese Kriterien oft nicht vollständig einhaltbar. Die Aufgabe des Requirements Engineering besteht darin, mit Werkzeugen und Methoden möglichst die Qualitätskriterien zu erreichen [Bal09, S. 477].
Um Anforderungen zu ermitteln, gibt es mehrere Möglichkeiten. Sie können unter anderem aus Altsystemen oder Dokumentationen gewonnen werden, die wichtigste Quelle sind aber die Stakeholder [Gol11, S. 166/167]. Darunter werden Personen verstanden, die direkten Einfluss auf die Anforderungen haben. Nutzer des Systems, Betreiber, Entwickler und Tester sind nur eine kleine Auswahl an möglichen Personen [Rup09, S. 62].
Bei so vielen unterschiedlichen Personen bietet sich die natürliche Sprache zum Formulieren von Anforderungen an. Ein Vorteil der natürlichen Sprache ist, dass es für den Menschen einfach ist mittels dieser zu kommunizieren, sofern alle Beteiligten dieselbe beherrschen. Außerdem ist sie flexibel (abstrakt oder konkret) und universell (für jede Domäne) einsetzbar. Nachteile sind ihre Mehrdeutigkeit (z.B. die lexikalische Mehrdeutigkeit des Wortes „Bank“) und die Verwendung vager Begriffe (z.B. „neben“). Um diese Nachteile zu reduzieren, schlägt Balzert in [Bal09, S. 481] unter anderem die Benutzung sprachlicher Anforderungsschablonen vor.

2.1.2. Anforderungsschablonen

Auch wenn genug Requirements Engineering betrieben wird, finden sich immer noch viele Probleme, die durch geeignete Beschreibungsmittel, durchgängige Methoden oder unterstützende Werkzeuge vermieden werden können [Par10, S. 13]. Eine Möglichkeit bieten

Schablonen. Die Vorteile von Schablonen liegen in der damit zu erreichenden hohen Qualität einer Anforderung und der leichten Erlernbarkeit.
Rupp stellt in [Rup09] Schablonen vor, auf diese soll hier näher eingegangen werden.

Eine Anforderungsschablone für das Deutsche für Anforderungen ohne Vorbedingung ist in Abbildung 2.5 dargestellt, eine Schablone für Anforderungen mit Vorbedingung in Abbildung 2.6.

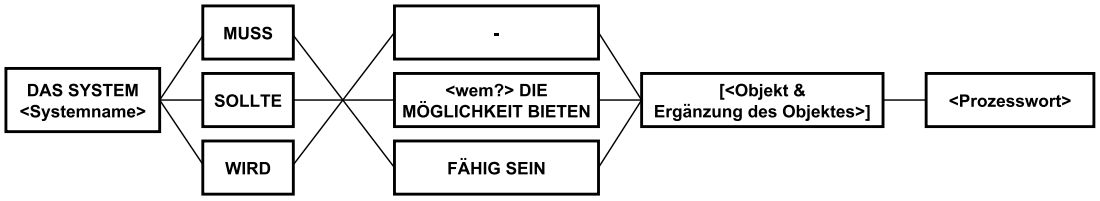

Abbildung 2.5.: Schablone für Anforderungen auf Deutsch ohne Vorbedingung [Rup09, S. 162]

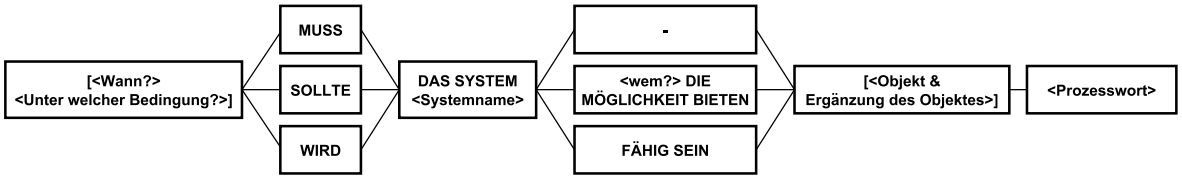

Abbildung 2.6.: Schablone für Anforderungen auf Deutsch mit Vorbedingung [Rup09, S. 166]

Die Schablonen von Rupp sind rein syntaktische Schablonen, da nur die Syntax festgelegt wird . Eine Anforderung benötigt, egal ob mit oder ohne Vorbedingung, immer eine rechtliche Verbindlichkeit. Es wird mittels definierter Hilfsverben unterschieden, ob es sich bei der Anforderung um eine Pflicht („muss“), einen Wunsch („sollte“) oder eine Absicht („wird“) handelt. Außerdem muss die Funktionalität festgelegt werden. Zur Wahl stehen die selbstständige Systemaktivität („-“), die Benutzerinteraktion („<wem?> die Möglichkeit bieten“) und die Schnittstellenanforderung („fähig sein“). Für Elemente in spitzen Klammern muss das entsprechende Wort bzw. die entsprechende Beschreibung eingesetzt werden. Elemente in eckigen Klammern sind optional, wie beispielsweise das Objekt. Ein weiterer Bestandteil ist das Prozesswort, das die Funktionalität der Anforderung beschreibt. Es ist immer ein Vollverb.
Bei einer Anforderung mit Voraussetzung werden zusätzlich eine oder mehrere Bedingungen vorangestellt unter denen die geforderte Funktionalität durchgeführt wird. Relevant sind zeitliche (z.B. „nachdem“) und logische (z.B. „falls“) Bedingungen [Rup09, S. 162–176].

Das Werkzeug, das in dieser Arbeit vorgestellt wird, übersetzt Sätze vom Deutschen ins Englische. Rupp stellt auch eine Schablone für englische Anforderungen zur Verfügung

(siehe Abbildung 2.7). Die Bestandteile sind die gleichen, wie bei deutschen Anforderungen.

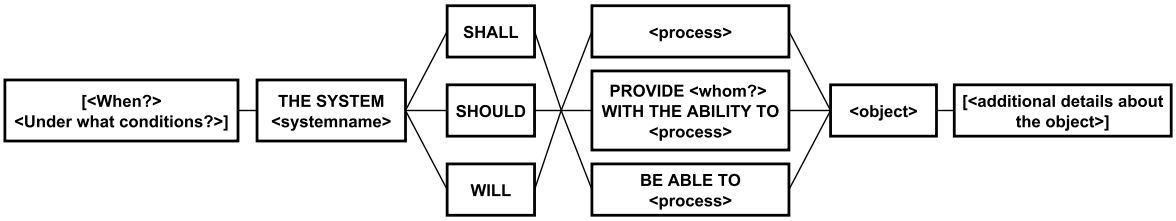

Abbildung 2.7.: Schablone für Anforderungen auf Englisch [Rup09, S. 177]

2.2. Maschinelle Übersetzung

Ziel dieser Arbeit ist es, Anforderungen, die mit den in Kapitel 2.1.2 vorgestellten Schablonen geschrieben wurden, automatisiert zu übersetzen. Hierfür gibt es verschiedene Verfahren, die in diesem Kapitel vorgestellt werden. Die maschinelle Übersetzung ist ein aktuelles Forschungsthema und beschäftigt sich unter anderem mit verschiedenen Übersetzungsproblemen. Auf einige wird in Abschnitt 2.2.3 eingegangen. Zunächst wird aber in 2.2.1 in die Begriffe der maschinellen Übersetzung eingeführt.

2.2.1. Grundlagen der maschinellen Übersetzung

Eine Übersetzung ist allgemein betrachtet die Transformation eines Textes von einer Quellsprache in eine Zielsprache. Die maschinelle Übersetzung wird von Hutchins und Somers in [HS92][1] ähnlich definiert. Sie beschreiben die maschinelle Übersetzung als ein Computersystem, das eine Übersetzung von einer natürlichen Sprache in eine andere produziert. Dabei kann das System die Unterstützung eines Menschen in Anspruch nehmen. Hutchins und Somers grenzen stark ab von Systemen, die den Menschen nur durch die Bereitstellung von Wörterbüchern oder ähnlichem unterstützen. Der Übersetzungprozess wird dann von einem Menschen durchgeführt. Diese Systeme fallen in die Kategorie „machine-aided human translation“ (MAHT). Die Systeme, die zur maschinellen Übersetzung gehören, werden unter dem Begriff „human-aided maschine translation“ (HAMT) zusammengefasst. Hierbei übersetzt der Computer und der Mensch kann das System durch Interaktion, Vor- oder Nachbearbeitung unterstützen [HS92, S. 149/150].
Ein maschinelles Übersetzungssystem kann die Übersetzung von einer Sprache in genau eine andere leisten, aber auch mehrere Sprachpaare zur Verfügung stellen. Ersteres ist ein bilinguales, zweiteres ein multilinguales System [CEE+04, S. 565]. Je nachdem, ob das

[1] vgl. [HS92] S. 3

System die Übersetzung dann nur in eine Richtung oder in beide Richtungen beherrscht, ist es unidirektional oder bidirektional.
Wie oben erwähnt, greifen manche Systeme auf die Hilfe des Menschen zurück. Systeme, die dies während des Übersetzungsprozesses tun, sind interaktiv. Alle anderen Systeme werden Batch-Systeme genannt. Ein interaktives System stellt dem Benutzer zum Beispiel mehrere Übersetzungsmöglichkeiten zur Auswahl und lässt ihn entscheiden. Dadurch kann nicht vorhandenes Wissen kompensiert werden.
Weitere Möglichkeiten die Übersetzung durch Zuhilfenahme des Benutzers zu verbessern, sind die Prä- und Postedition. Durch Präedition können vermeindliche Probleme schon vor der Übersetzung entdeckt und behoben werden. Hutchins und Somers stellen das System SUSY[2] vor, bei dem beispielsweise Eigennamen aber auch ganze Satzteile mit bestimmten Symbolen markiert werden um dem System mehr Informationen zu liefern und die Übersetzung zu vereinfachen. Das Ziel sind einfache und eindeutige Sätze. Bei der Postedition wird der übersetzte Satz vom Benutzer korrigiert. Manche Systeme markieren Sätze oder Wörter, um daraufhinzuweisen, dass möglicherweise Übersetzungsfehler im Satz vorkommen.
Systeme können aber auch durch andere Optionen verbessert werden. Die Nutzung einer kontrollierten Sprache ist eine Möglichkeit. Systeme haben häufig Probleme damit, bestimmte Satzstrukturen zu analysieren und zu interpretieren oder die Mehrdeutigkeit von Wörtern aufzulösen. Eine kontrollierte Sprache untersteht Regeln, die zum Beispiel den Satzbau einschränken oder nur Wörter aus einem Wörterbuch, die dort hinsichtlich ihrer Bedeutung eindeutig definiert sind, zulassen [Ram09, S. 118]. Außer der Benutzung einer kontrollierten Sprache kann auch eine Subsprache verwendet werden. Das Ziel, die Sprache eindeutiger zu machen, bleibt das selbe. Eine Subsprache ist beschränkt auf eine Domäne, zum Beispiel das Wetter. Die Sprache wird nicht direkt durch Regeln eingeschränkt aber unterliegt sowohl grammatikalischen als auch semantischen und lexikalischen Restriktionen. Die Restriktionen ergeben sich durch die Subsprache selbst. Wörter haben zum Beispiel abhängig vom Kontext unterschiedliche Bedeutungen (z.B. „Bank “). In einer Subsprache ist meist klar, welche Bedeutung gemeint ist. Allerdings lassen sich durch eine Subsprache nur lexikalische Mehrdeutigkeiten beheben [Ram09, S. 117]. Eine Kombination aus kontrollierter Sprache und Subsprache ist in manchen Fällen sinnvoll.

2.2.2. Verfahren zur maschinellen Übersetzung

Maschinelle Übersetzungssysteme agieren nach verschiedenen Verfahren. Die erste und zweite Generation der Verfahren sind die regelbasierten Verfahren. Diese sind heutzutage häufig Basis kommerzieller Systeme. Darunter fallen die direkte, die transferbasierte und die interlinguabasierte Übersetzung. Die dritte Generation von Systemen gehört zu den aktuellen Forschungsthemen. Dazu zählen die beispielbasierte, die wissensbasierte und die statistische Übersetzung [Hei04].

[2] vgl. [HS92] S. 151

Die regelbasierten Verfahren bestehen aus der Analyse des Satzes in der Quellsprache und der anschließenden Generierung des Satzes in der Zielsprache. Um aus dem analysierten Satz in der Quellsprache eine Repräsentation zu erhalten, aus der ein Satz in der Zielsprache generiert werden kann, kann ein Transfer notwendig sein. Abbildung 2.8 zeigt, dass bei einer direkten Übersetzung der Aufwand für die Analyse und auch für die Generierung am geringsten ist. Für eine interlinguabasierte Übersetzung sind Analyse und Generierung am aufwändigsten, dafür ist kein Transfer notwendig. Der transferbasierte Ansatz liegt vom Aufwand in der Mitte [CEE+04, S. 565].

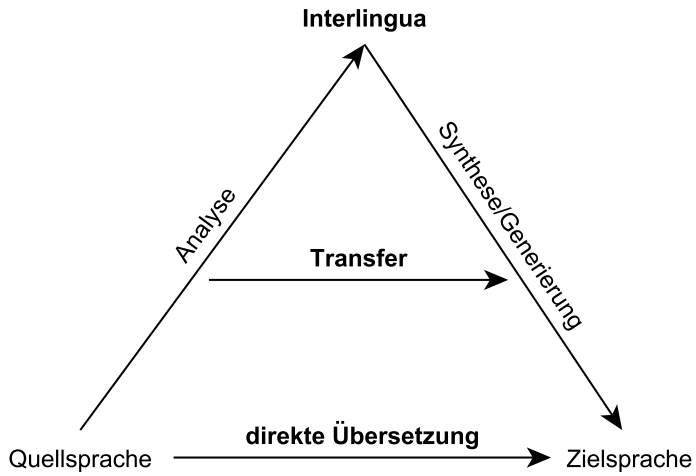

Abbildung 2.8.: Dreieck der maschinellen Übersetzung von Vauquois [CEE+04, S. 565]

Bei einer **direkten Übersetzung** wird auf der Basis eines quellsprachlichen Satzes direkt der Satz in der Zielsprache generiert [Ram09, S. 73]. Der Quelltext (QT) wird nur sehr flach analysiert, d. h. es findet nur eine Analyse der Morphologie statt, aber keine syntaktische oder semantische Analyse, dadurch ensteht die Zwischenrepräsentation QT'. Anschließend wird jedes Wort mithilfe eines bilingualen Lexikons übersetzt. Es entsteht ein Zieltext ZT', der noch syntaktisch angepasst werden muss. Deswegen wird die Wortstellung anhand einfacher Regeln korrigiert. Das Ergebnis ist der übersetzte Satz in Zielsprache (ZT). Abbildung 2.9 stellt diesen Vorgang dar. Die Vorteile der direkten Übersetzung liegen in den einfachen Verarbeitungsschritten, der Geschwindigkeit und Robustheit. Allerdings ist eine ansatzweise gute Qualität nur dann zu erwarten, wenn die Quell- und Zielsprache sich syntaktisch sehr ähnlich sind [CEE+04, S. 566/567]. Durch die unzureichende linguistische Analyse des Satzes kann nicht immer entschieden werden, ob ein Satz in Quellsprache grammatisch korrekt ist und es werden deswegen eventuell auch ungrammatische Sätze in der Zielsprache generiert [Ram09, S.74].

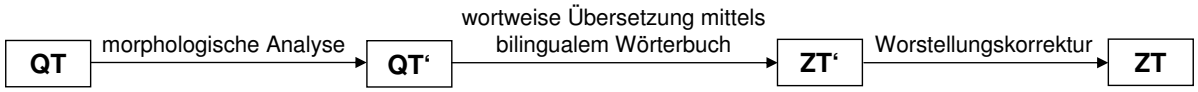

Abbildung 2.9.: Direkte Übersetzung

In Abbildung 2.10 wird die **transferbasierte Übersetzung** dargestellt. Diese baut auf eine tiefere Analyse. Dadurch steigt allerdings auch der Aufwand zur Generierung des Satzes in der Zielsprache. Im ersten Schritt wird der Satz syntaktisch analysiert. Das Ergebnis der Analyse ist eine abstrakte Repräsentation des Satzes. Diese Repräsentation ist allerdings immer noch abhängig von der Quellsprache (QS). Im zweiten Schritt wird die quellsprachliche Repräsentation mittels Transferregeln in eine Repräsentation überführt, die abhängig von der Zielsprache (ZS) ist. Daraus wird anschließend der endgültige Satz generiert [Ram09, S. 75–77]. Die Analyse und Generierung ist im Gegensatz zur direkten Analyse aufwändiger, aber dadurch lassen sich auch wesentlich bessere Ergebnisse erzielen. Die erzeugten Repräsentationen sind jedoch sprachenabhängig. Wenn eine neue Sprache in das System integriert werden soll, muss auch für diese ein Analyse- bzw. Generierungsmodul entwickelt werden. Auch die Transferregeln sind meist unidirektional und müssen für eine umgekehrte Übersetzungsrichtung neu entworfen werden [CEE+04, S. 567].

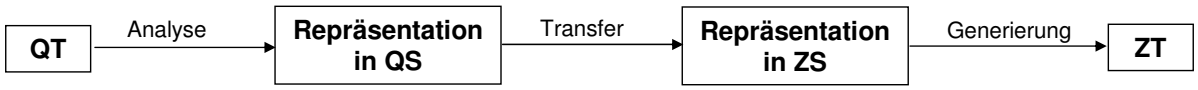

Abbildung 2.10.: Transferbasierte Übersetzung

Die meisten Nachteile der transferbasierten Übersetzung werden im **interlinguabasierten Ansatz** aufgegriffen und verbessert. Die Interlingua ist eine sprachenunabhängige Repräsentation. Durch die Analyse wird der Satz in diese überführt und mit dem Generierungsmodul daraus ein Satz in der Zielsprache generiert (siehe Abbildung 2.11). Für jede Sprache ist genau ein Analyse- und ein Generierungsmodul notwendig. Transferregeln werden nicht gebraucht, da die Repräsentation dank der Sprachunabhängigkeit für einen Satz immer dieselbe ist. Bis jetzt ist es aber unmöglich gewesen eine solche Repräsentation zu finden und ein solches System zu entwickeln. Der Grund hierfür kann leicht eingesehen werden. Zum Beispiel für das Wort „Beine“ müsste in der Repräsentation unterschieden werden zwischen menschlichen Beinen, Tischbeinen, Tierbeinen, etc. Es ist nicht möglich eine vollständige Interlingua zu erstellen. Denkbar ist eine Interlingua nur für eine sehr eingeschränkte Subsprache oder kontrollierte Sprache [CEE+04, S. 567/568].
Der interlinguabasierte Ansatz hat, wie erwähnt, den größten Aufwand für die Analyse und die Generierung. Dies kann als Nachteil gesehen werden, da immer eine volle Analyse und Generierung für jede Sprache durchgeführt werden muss. Bei neueren Verfahren wird dies durch eine andere Herangehensweise umgangen.

Abbildung 2.11.: Interlinguabasierte Übersetzung

Zur dritten Generation und damit zu den neueren Methoden der maschinellen Übersetzung, gehört der **beispielbasierte Ansatz**. Grundlage des Ansatzes bietet ein bilinguales Korpus, das Zuordnungen zwischen Wörtern, (Teil-)Phrasen und Sätzen der beiden Sprachen enthält. Für einen zu übersetzenden Satz werden möglichst große Teile des Satzes im Korpus gesucht. Umso größer diese Teilsätze sind, desto besser ist die resultierende Übersetzung [CEE+04, S. 568]. Auf linguistische Regeln wird in diesem Ansatz verzichtet. Gibt es keine passenden Elemente im Korpus, kann keine Übersetzung erzeugt werden [Ram09, S. 79/80].

Ein weiteres Verfahren der dritten Generation ist das **statistische Verfahren**. Dieses benötigt ebenso wie der beispielbasierte Ansatz ein großes bilinguales Korpus. Der Ansatz basiert auf drei Basismodellen: dem Alignment-Modell, dem Sprachmodell und dem Übersetzungsmodell. Das Alignment-Modell enthält die Wahrscheinlichkeit für jedes Wort in bestimmten Positionen innerhalb eines Satzes vorzukommen. Das Sprachmodell gibt die Wahrscheinlichkeit an mit der Wörter als Nachfolger anderer Wörter auftauchen und das Übersetzungsmodell ordnet einem Wort Übersetzungsmöglichkeiten und deren Wahrscheinlichkeit zu [Ebe09, S. 46]. Um einen Satz, der in der Quellsprache vorliegt, zu übersetzen, wird die Übersetzung mit der größten Wahrscheinlichkeit ausgewählt. Auch dieser Ansatz benutzt kein linguistisches Wissen [Ram09, S. 80–82].

Allen bisher vorgestellten Verfahren fehlt Weltwissen. Weltwissen ist das allgemeine Wissen über die Umwelt, das es möglich macht unbekannte Informationen einzuordnen und Erkenntnisse daraus abzuleiten. Ein Satz kann eigentlich nur erfolgreich übersetzt werden, wenn die Bedeutung des Satzes erfasst und im Zielsatz wiedergegeben werden kann. Dazu ist oft Weltwissen notwendig. Der **wissensbasierte Ansatz** macht vorallem Sinn, wenn ganze Texte übersetzt werden sollen. Die anderen Verfahren arbeiten derzeit noch ausschließlich satzweise. Der Kontext der Sätze enthält notwendiges Wissen, das für eine Übersetzung genutzt werden muss. Zusätzlich ist weiteres Wissen, das als bekannt vorausgesetzt wird, notwendig für eine Übersetzung.
Der Schwerpunkt der Analyse des Verfahrens liegt in der Semantik. Diese wird mittels vorhandenem Weltwissen interpretiert und in einer sprachenunabhängigen Repräsentation gespeichert. Daraus wird dann der Text in Zielsprache generiert. Das Verfahren ist stark an das interlinguabasierte Verfahren angelehnt. Auch hier taucht das Problem auf, dass nicht das komplette Weltwissen erfasst und gespeichert werden kann. Erfolgversprechend ist das Verfahren deswegen nur in einer klar abgegrenzten Domäne [Ram09, S. 82].

Der Trend im Bereich der maschinellen Übersetzung geht zu Hybridsystemen. Angestrebt wird eine Kombination aus einem regelbasierten System und einem neueren Ansatz. Die neuen Verfahren werden um linguistisches Wissen angereichert, damit eine bessere Übersetzungsqualität erreicht werden kann.

2.2.3. Probleme der maschinellen Übersetzung

Bei einer Übersetzung tauchen verschiedene Übersetzungsprobleme auf. Auch menschliche Übersetzer werden mit diesen Problemen konfrontiert, sind aber bei der Lösung stets flexibler als maschinelle Übersetzungssysteme. Auf einge Probleme soll im Folgenden eingegangen und der Sachverhalt an Beispielen erläutert werden.
Die Probleme können unterschieden werden in solche, die bei der Übersetzung und Probleme, die bei der Analyse einer Sprache auftreten. Eine Kategorie von Problemen, die bei der Übersetzung vorkommen, sind lexikalische Lücken (gaps) und fehlende Entsprechungen (mismatches). Dies liegt vor, wenn ein Wort der einen Sprache nicht direkt mit einem Wort der anderen Sprache übersetzt werden kann. Als Beispiel gibt es für das deutsche Wort „sich verwählen“ keine direkte englische Übersetzung, sondern muss mit der Umschreibung „dial the wrong number“ übersetzt werden. Ein weiteres Übersetzungsproblem, das auftreten kann, ist die unterschiedliche Granularität von Wörtern. Das Wort „know“ kann sowohl mit „kennen“ als auch mit „wissen“ übersetzt werden. Die richtige Übersetzung kann allerhöchstens aus dem Kontext erschlossen werden [Hei04].
Auch auf der Satzebene tauchen Probleme auf. Übersetzungen können sich in ihrer syntaktischen Struktur unterscheiden. „Das Buch gefällt Eva“ wird übersetzt mit „Eva likes the book“. Das Subjekt von „gefallen“ wird zum Objekt von „like“ und das Objekt zum Subjekt. Diese Kategorie von Problemen wird Divergenzen genannt [CEE+04, S. 564]. Eine andere Form der Divergenz ist das Head Switching. Das Adverb „gerne“ im Satz „John schwimmt gerne.“ hat keine englische Übersetzung in der gleichen Kategorie. Der Satz muss durch eine Vertauschung des Kopfs (Erklärung „Kopf“ und Phrasenstruktur siehe Kapitel 2.3.2) übersetzt werden. Die Übersetzung lautet dann „John likes to swim“, wobei „like“ ein Verb ist (siehe Abbildung 2.12) [Hei04].

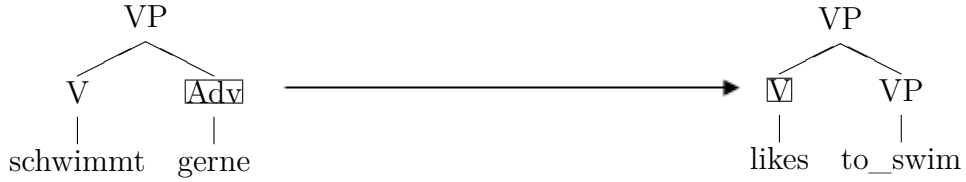

Abbildung 2.12.: Head Switching

Zu den Problemen, die bei der Analyse auftreten, gehören auch Mehrdeutigkeiten (Ambiguitäten). Diese können schon während der morphologischen Analyse auftreten. Zum

Beispiel das Wort „Staubecken“ kann entweder in Stau-becken oder in Staub-ecken zerlegt werden. Auf der Syntaxebene gibt es strukturelle oder kategoriale Ambiguitäten. Syntaktisch ambig sind Phrasen, wie zum Beispiel „Mütter und Kinder unter 12 Jahren“. Es ist nicht klar, ob sich die Modifikation „unter 12 Jahren“ nur auf das Wort „Kinder“ oder auch auf das Wort „Mütter“ bezieht [Hei04]. Eine kategoriale Ambiguität liegt dagegen vor, wenn ein Wort mehreren Kategorien zugeordnet werden kann. Zum Beispiel kann das englische Wort „flies“ ein Nomen (im Sinn von „die Fliegen“) oder Verb (im Sinn von „fliegen“) sein [Ebe09, S. 38/39].
Auch auf der semantischen Ebene können Ambiguitäten auftreten. Bekannt sind vorallem Homographe. Dies sind Wörter die gleich geschrieben werden aber eine unterschiedliche Bedeutung haben. Zu dieser Kategorie gehört das Wort „Bank“. Es ist zunächst unklar, ob es sich dabei um eine Sitzgelegenheit oder eine Finanzinstitution handelt. Ähnlich sind polyseme Wörter, wie zum Beispiel das Wort „Zweige“. Hier stellt sich die Frage, ob die Zweige eines Baums oder die verschiedenen Bereiche eines Fachgebiets gemeint sind [Hei04].
Als letzte Ambiguität sei noch die referentielle Ambiguität erwähnt. Referentiell ambig ist folgendes Beispiel: „Die Katze spielt mit der Maus. Sie mag das nicht.“ Das Pronomen „sie“ kann sich sowohl auf die Katze als auch auf die Maus beziehen [Hei04]. Referentielle Ambiguitäten können bei Übersetzungsstrategien, die satzweise arbeiten, nicht aufgelöst werden.

2.3. Grammatiken

Im Rahmen dieser Arbeit wird ein transferbasiertes Übersetzungssystem entwickelt. Eine Diskussion über die Auswahl des Verfahrens findet sich in Kapitel 4.1. Dieses Verfahren beinhaltet zunächst eine Analyse des zu übersetzenden Satzes. Für eine Analyse der Anforderungen, die mit den in Kapitel 2.1.2 vorgestellten Schablonen geschrieben wurden, wird ein Beschreibungsmittel benötigt. Hierfür eignen sich Grammatiken, die in diesem Kapitel vorgestellt werden.
Außerdem können, wenn eine Grammatik für deutsche Anforderungen und eine Grammatik für englische Anforderungen existiert, leichter Transferregeln definiert und der übersetzte Satz auf syntaktische Richtigkeit überprüft werden.

2.3.1. Grundlagen der Grammatiktheorie

Mit einer Grammatik können Wörter erzeugt werden, welche dann eine formale Sprache bilden. Formale Sprachen kommen aus dem Bereich der theoretischen Informatik, sind aber gerade für die maschinelle Sprachverarbeitung von großer Bedeutung. Denn, um die natürliche Sprache zu verarbeiten, muss bekannt sein, mit welchen Mitteln diese analy-

siert werden kann. Dazu ist es möglich die natürliche Sprache oder relevante Ausschnitte in eine Hierarchie formaler Sprachen einzuordnen. Den Hierarchiestufen sind jeweils geeignete Verarbeitungsmöglichkeiten zugeordnet. Mit Grammatiken lassen sich also nicht nur formale Sprachen, sondern auch natürliche Sprachen beschreiben [CEE+04, S. 63]. Abbildung 2.13 zeigt die Definition einer Grammatik.

Definition: Grammatik

Eine Grammatik G ist ein Viertupel (V, Σ, P, S), wobei

- V die endlichen Menge der Nichtterminale,
- Σ die endlichen Menge der Terminale,
- P die Menge der Regeln und
- S das Startsymbol

ist.

Zusätzlich muss gelten:

- $V \cap \Sigma = \emptyset$
- P ist eine endliche Teilmenge von $(V \cup \Sigma)^+ \times (V \cup \Sigma)^*$
- $S \in V$

Abbildung 2.13.: Definition „Grammatik“ nach [Sch03a]

Ein Terminalzeichen ist ein Zeichen, das nicht allein auf der linken Seite einer Regel vorkommen kann. Es ist nicht möglich ein Terminalzeichen alleine zu ersetzen, im Gegensatz zu einem Nichtterminalzeichen, welches nicht in einem terminalen Wort der Sprache vorkommen kann und durch Regeln ersetzt werden muss. Eine Regel besteht aus $u \in (V \cup \Sigma)^+$, $v \in (V \cup \Sigma)^*$ und sieht folgendermaßen aus: $u \rightarrow v$. Dies bedeutet, dass u unmittelbar in v übergeht bzw. u durch v ersetzt wird. Das Anwenden einer Regel entspricht einem Ableitungsschritt. Eine Ableitung ist das mehrmalige Anwenden der Regeln um ein Wort der Sprache zu erzeugen, die durch G beschrieben wird. Die erzeugte Sprache wird L(G) genannt (Definiton siehe Abbildung 2.14) [Sch03a, S. 13/14].

Definition: Formale Sprache

Sei G = (V, Σ, P, S) eine Grammatik. Dann ist die durch G erzeugte formale Sprache

$L(G) = \{w \in \Sigma^* \mid S \Rightarrow^* w\}$

Abbildung 2.14.: Definition „Formale Sprache“ nach [Sch03a]

Beispielgrammatik 1
G = (V, Σ, P, S) wobei V = {S,A,B,C}, Σ = {a,b} und
P = {S → AC, B → b, A → a, C → BA, C → AB}

Abbildung 2.15.: Beispielgrammatik 1

Mit der Beispielgrammatik in Abbildung 2.15 kann unter anderem folgendes Wort mittels Ableitung produziert werden: S ⇒ AC ⇒ ABA ⇒ aBA ⇒ abA ⇒ aba
Die durch G beschriebene Sprache L(G) besteht nur aus zwei Wörtern: L(G) = {aba, aab}

Eine Ableitung kann auch durch einen Syntaxbaum dargestellt werden. Die Wurzel des Baums ist die Startvariable S. Wird in einem Ableitungsschritt die Regel $u \to v$ angewandt, dann hat der Knoten u im Syntaxbaum $|v|$ viele Kinder. Diese werden mit den Symbolen v_1, v_2, etc. (wegen v = $v_1 v_2 ... v_n \in (V \cup \Sigma)^*$) beschriftet [Sch03a, S. 23].

Abbildung 2.16 zeigt einen Syntaxbaum für die Ableitung in obigem Beispiel.

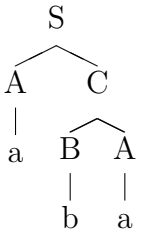

Abbildung 2.16.: Syntaxbaum einer Ableitung der Beispielgrammatik G

Um Grammatiken zu kategorisieren und Verarbeitungsmittel zuzuordnen, hat Noam Chomsky, die nach ihm benannte Chomsky-Hierarchie definiert. Diese hat er 1956 zum ersten Mal in seinem Werk „Three models for the description of language“ [Cho56] vorgestellt. Es existieren Grammatiken vom Typ 0 bis Typ 3. Jede Grammatik ist eine Typ 0-Grammatik, denn für sie gelten keine Einschränkungen. Bei den anderen Typen müssen die Regeln bestimmten Kriterien entsprechen.
In einer Typ 1-Grammatik oder auch *kontextsensitiven* Grammatik gibt es nur Regeln $u \to v$ für die Folgendes gilt: $|u| \leq |v|$. Eine Grammatik ist *kontextfrei* bzw. vom Typ 2, wenn für alle Regeln $u \to v$ in P gilt, dass u eine einzelne Variable ist d. h. $u \in V$. Typ 3 oder *regulär* ist eine Grammatik, wenn die rechte Seite aller Regeln entweder aus einem einzelnen Terminalzeichen oder aus einem Terminalzeichen gefolgt von einem Nichtterminalzeichen besteht. Die Regeln werden für jeden höheren Typ stärker eingeschränkt. Die Typen sind deswegen jeweils Teilmengen des vorhergehenden Typs d. h., eine Typ 3-Grammatik ist auch immer eine Grammatik vom Typ 2, Typ 1 und Typ 0, eine Typ 2-Grammatik immer auch vom Typ 1 und Typ 0, usw. Die im obigen Beispiel angegeben Grammatik ist vom Typ 2 [Sch03a, S. 17].

2.3.2. Phrasenstrukturgrammatik und X-Bar-Theorie

Ein Satz ist im Prinzip eine lineare Folge von Wörtern. Würde dies in der Syntax genauso aufgefasst werden, würde, je nach Andwendung, notwendiges Wissen über die innere Struktur eines Satzes verloren gehen. Deswegen werden weitere Kenntnisse hinzugezogen, um eine tiefere Struktur zu konstruieren. Zu diesen Kenntnissen gehört, dass die Wörter eines Satzes in Gruppen unterteilt sind, den so genannten Konstituenten. Diese Konstituenten sind wiederum Teil einer größeren Konstituente bis irgendwann der Satz den Abschluss bildet [Car02, S. 27].
Um Konstituenten zu bilden werden zunächst den Wörtern Wortarten zugeordnet. Anschließend können Aussagen darüber gemacht werden, welche Konstituenten (auch Phrasen genannt) aus welchen Wortarten bestehen. Anstatt von Wortarten wird hier auch von syntaktischen Kategorien gesprochen. Es gibt lexikalische und funktionale Kategorien. Zu den lexikalischen gehören die Nomen N, Verben V, Präpositionen P und Adjektive A. Die übrigen Kategorien (z.B. Det oder I) gehören zu den funktionalen Kategorien [Car02, S. 33–41].
Eine Grammatik, die den Aufbau des Satzes durch Phrasen und die Struktur der Phrasen selbst beschreibt, heißt Phrasenstrukturgrammatik. Diese besteht aus lexikalischen Regeln (z.B. N → Händler) und Phrasenstrukturregeln (z. B. NP → Det N') [Sah09, S. 1].

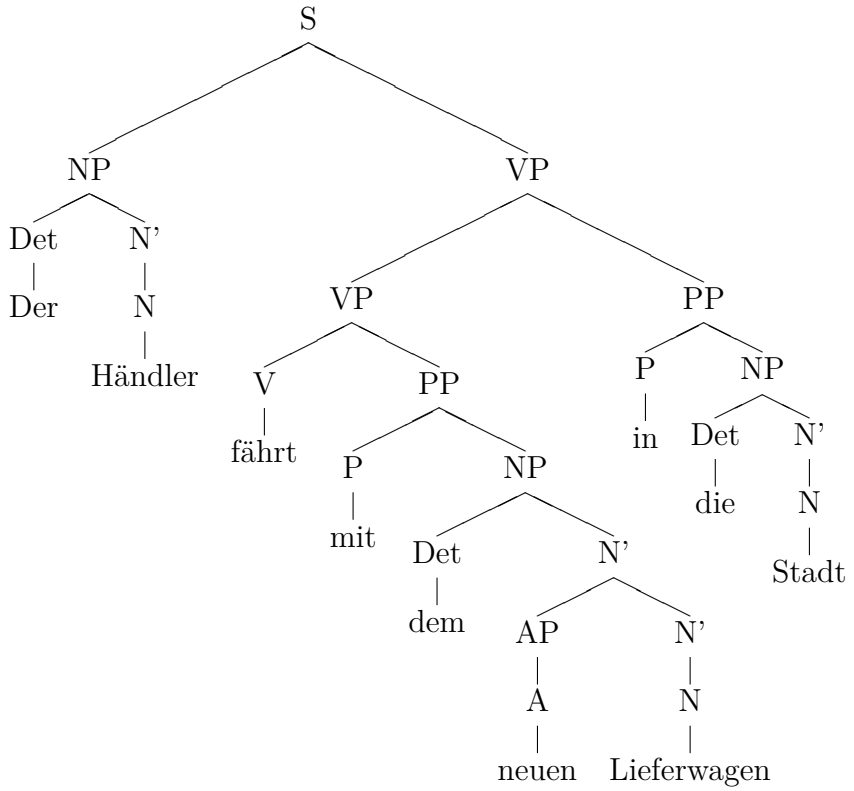

Abbildung 2.17.: Phrasenstrukturbaum: „Der Händler fährt mit dem neuen Lieferwagen in die Stadt."

Lexikalische Regeln bestehen aus einem Nichtterminal auf der linken Seite und einem Terminal auf der rechten Seite. Phrasenstrukturregeln beschreiben die Struktur des Satzes und beinhalten weder auf der linken noch auf der rechten Seite Terminale.
Abbildung 2.17 zeigt den Aufbau eines Satzes durch eine Phrasenstrukturgrammatik, dargestellt als Ableitungsbaum. Der Phrasenstrukturbaum besteht aus der Satzkonstituente S, diese wiederum aus der Nominalphrase NP „Der Händler“ und der Verbalphrase VP „fährt mit dem neuen Lieferwagen in die Stadt“. Die NP „Der Händler“ ist unterteilt in das Nomen „Händler“ und den Artikel „der“. Auch die Verbalphrase besteht aus zwei Teilen, der VP „fährt mit dem neuen Lieferwagen“ und der Präpositionalphrase PP „in die Stadt“. Die VP „fährt mit dem neuen Lieferwagen“ enthält das Verb „fährt“ und die PP „mit dem neuen Lieferwagen“. Die Präposition „mit“ und die NP „dem neuen Lieferwagen“ bilden zusammen die eben erwähnte PP. Die NP „dem neuen Lieferwagen“ ist unterteilt in eine Zwischenprojektion N’ (Erklärung siehe Seite 21) und einen Artikel „dem“. Zum Schluss kommt zusätzlich zum Nomen „Lieferwagen“ in der Zwischenprojektion N’ noch eine Adjektivphrase vor, die nur aus dem Adjektiv „neuen“ besteht. Die PP „in die Stadt“ ist analog aufgebaut.

Eine Phrasenstrukturgrammatik für dieses Beispiel kann aussehen, wie in Abbildung 2.18 dargestellt.

Beispielgrammatik 2	
Phrasenstrukturregeln:	lexikalische Regeln:
S → NP VP	Det → der
NP → Det N’	Det → dem
N’ → N	Det → die
N’ → AP N’	N → Lieferwagen
VP → VP PP	N → Stadt
VP → V PP	N → Händler
PP → P NP	P → mit
AP → A	P → in
	A → neuen
	V → fährt

Abbildung 2.18.: Beispielgrammatik 2

Der Aufbau der Phrasenstrukturregeln wurde in einer einheitlichen Theorie formuliert, der X-Bar-Theorie. Danach sind alle Phrasenstrukturregeln nach dem gleichen Prinzip aufgebaut [Car02, S. 114]. Dieses Prinzip wird in Abbildung 2.19 dargestellt.

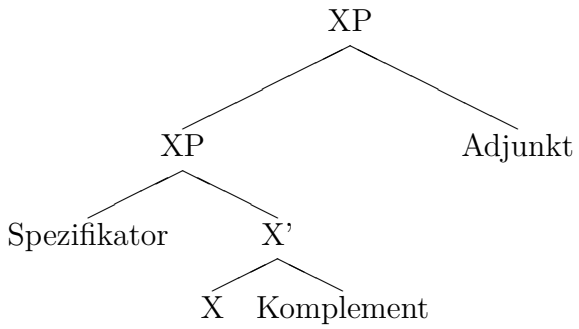

Abbildung 2.19.: Allgemeine X-Bar-Struktur

Eine allgmeine Phrase XP besitzt ein obligatorisches Element X. Dieses Element ist der Kopf der Phrase. Bezogen auf die vorher genannten Phrasen bedeutet das, dass zum Beispiel eine Verbalphrase als Kopf immer ein Verb V und eine Nominalphrase immer ein Nomen N hat [Sah06, S. 3]. X vererbt seine lexikalischen Merkmale an die gesamte Phrase, indem es diese über eine Zwischenprojektion X' auf die maximale Projektion XP projiziert. Eine Zwischenprojektion kann notwendig sein, da Phrasen sehr komplex sein können, um eine weitere Verzweigung zu erlauben. Mehrere Zwischenprojektionen sind denkbar. Zur Vereinfachung wird oft abgekürzt und die Zwischenprojektionen weggelassen (siehe Abbildung 2.17). Durch das X-Bar-Prinzip sind nur binäre Verzweigungen erlaubt [BEW06, S. 2–4].
Zusätzlich zur maximalen Projektion XP, der Zwischenprojektion X' und dem Kopf X kann es optional noch Spezifikatoren, Komplemente und Adjunkte geben. Alle sind wiederum Phrasen mit eigenen Köpfen.
Ein Spezifikator liefert weitere Informationen über den Kopf, spezifiziert diesen genauer [BEW06]. Er ist immer der Schwesterknoten einer Zwischenprojektion X'. Welcher Phrasentyp als Spezifikator vorkommt, ist abhängig vom Kopf.
Komplemente und Adjunkte werden meist durch den Subkategorisierungsrahmen des Kopfs verlangt. Ein Subkategorisierungsrahmen gibt die Beziehungen zwischen dem Kopf und anderen Bestandteilen des Satzes an. Besonders bei Verben ist eine Angabe, welche Kategorien mit welchen Merkmalen in Beziehung stehen, sinnvoll. Ein Verb kann mehrere Subkategorisierungsrahmen haben. Das Verb „fahren“, zum Beispiel, hat sowohl den Subkategorisierungsrahmen <np:nom, pp:in (akk)> als auch den Subkategorisierungsrahmen <np:nom, np:akk> und <np:nom>. Der erste Subkategorisierungsrahmen ermöglicht Sätze, wie zum Beispiel „(Der Händler)$_{np:nom}$ fährt (in die Stadt)$_{pp:in\_akk}$“, der zweite Subkategorisierungsrahmen ermöglicht Sätze, wie zum Beispiel „(Der Händler)$_{np:nom}$ fährt (den Lieferwagen)$_{np:akk}$“ und der dritte Subkategorisierungsrahmen ermöglicht Sätze, wie „(Der Händler)$_{np:nom}$ fährt.“. Der Subkategorisierungsrahmen drückt aus, dass das Verb „fahren“ entweder nur eine Nominalphrase im Nominativ oder eine Nominalphrase im Nominativ und eine Nominalphrase im Akkusativ oder eine Nominalphrase im Nominativ und eine Präpositionalphrase mit der Präposition „in“ im Akkusativ verlangt.

Das Komplement ist immer der Schwesterknoten des Kopfs X und das Adjunkt immer der Schwesterknoten der maximalen Projektion XP. Ob das Komplement links oder rechts vom Kopf steht ist sprachspezifisch. Im Deutschen steht es bei VPs und APs links, bei den PPs rechts vom Kopf. Im Englischen steht das Komplement generell rechts vom Kopf [Sah09, S. 4]. Im Phrasenstrukturbaum der Abbildung 2.17 auf Seite 19 ist zum Beispiel der Artikel „der“ Spezifikator des nominalen Kopfs „Händler“ und die Präpositionalphrase „mit dem neuen Lieferwagen“ ist das Komplement des Kopfs V. Die Präpositionalphrase „in die Stadt“ hingegen ist ein Adjunkt zur Verbalphrase „fährt mit dem neuen Lieferwagen“.
Die Begriffe „Spezifikator“, „Adjunkt“ und „Komplement“ sind nur im syntaktischen Sinn und nach obiger Definition zu verstehen.
Eine Phrasenstrukturgrammatik, aufgebaut nach dem X-Bar-Schema, kann mit Verweis auf die im vorherigen Unterkapitel 2.3.1 vorgestellte Chomsky-Hierarchie als Typ 2-Grammatik eingeordnet werden.
Auf den hier verwendeten Grundlagen, speziell dem X-Bar-Schema, basieren auch andere Grammatikmodelle, wie zum Beispiel die LFG (Lexikalisch-Funktionale Grammatik). Das in dieser Arbeit verwendete Prinzip ist stark an die LFG angelehnt. Näheres zur LFG kann in [Bre01] nachgelesen werden.

2.3.3. Merkmalsstrukturen

Ein großes Problem für die maschinelle Übersetzung stellen übergenerierende Grammatiken dar. D. h. es können Sätze konstruiert werden, die es in der jeweiligen Sprache nicht gibt. Mit der in Kapitel 2.3.2 in Abbildung 2.18 einfachen Beispielgrammatik können zum Beispiel Sätze, wie „Dem Händler fährt mit die neuen Lieferwagen“, generiert werden. Der zugehörige Phrasenstrukturbaum ist in Abbildung 2.20 dargestellt.
Um dies bei kontextfreien Sprachen zu verhindern, können viele neue Regeln und Nichtterminale hinzugefügt werden, wie zum Beispiel NP_{sg} und NP_{pl}. Oder es werden Merkmalsstrukturen eingeführt. Eine Merkmalsstruktur besteht aus Attribut-Wert-Paaren und wird meist in Form einer Matrix dargestellt. Dadurch können zum Beispiel Wörter aber auch Nichtterminale genauer beschrieben werden. Mögliche Merkmale und Werte sind in Tabelle 2.1 aufgelistet.
Die Merkmalsstruktur des Wortes „Händler“, dargestellt in Abbildung 2.21, kann beispielsweise aus den Merkmalen Numerus (NUM), Genus (GEND), Person (PERS) und Kasus (KAS) bestehen. NUM wird dann der Wert Singular (*sg*), GEND der Wert maskulin (*masc*), PERS der Wert *3* für dritte Person und KAS der Wert Nominativ (*nom*) zugewiesen. Bezogen auf die Beispielgrammatik kann verlangt werden, dass die Merkmale und die dazugehörigen Werte der Merkmalsstrukturen der Nomen, Artikel und Verben übereinstimmen. Außerdem kann über die Merkmale geprüft werden, ob der Subkategorisierungsrahmen der Verben mit den vorhandenen Elemente erfüllt ist. Durch die Merkmalsstrukturen sollen ungrammatische Sätze verhindert werden [CEE+04, S. 94].

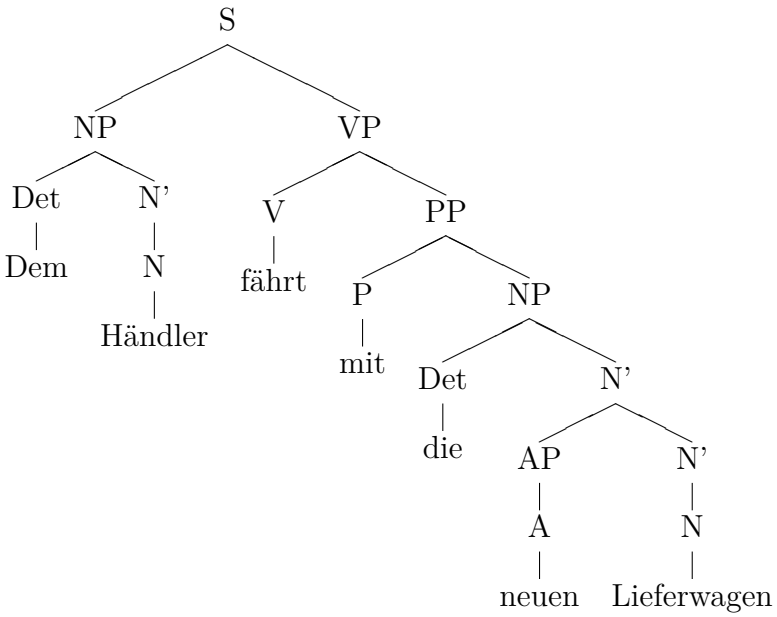

Abbildung 2.20.: Ungrammatischer Phrasenstrukturbaum: „Dem Händler fährt mit die neuen Lieferwagen.“

$$\begin{bmatrix} \text{NUM} & \text{sg} \\ \text{GEND} & \text{masc} \\ \text{PERS} & 3 \\ \text{KAS} & \text{nom} \end{bmatrix}$$

Abbildung 2.21.: Beispiel für Merkmalsstruktur des Wortes „Händler“

MERKMALE		WERTE	
Numerus	NUM	Plural	pl
		Singular	sg
Genus	GEND	maskulin	masc
		feminin	fem
		neutrum	neut
Person	PERS	erste Person	1
		zweite Person	2
		dritte Person	3
Kasus	KAS	Nominativ	nom
		Genitiv	gen
		Dativ	dat
		Akkusativ	akk

Tabelle 2.1.: Merkmale und Werte für eine Merkmalsstruktur

2.4. Parsing

Grundlage für das Parsen von Sätzen sind die in Kapitel 2.3 beschriebenen Grammatiken. Eine Grammatik ist ein Beschreibungsmittel, um die Struktur von Sätzen darzustellen. Parsen bedeutet, die syntaktische Struktur eines Satzes zu erkennen. Besonders bei der maschinellen Sprachverarbeitung wird das Parsen für die Analyse benötigt. Im folgenden Kapitel werden die Grundlagen des Parsings erläutert und auf ein Parsingverfahren, das Chart-Parsen, näher eingegangen.

2.4.1. Grundlagen des Parsings

Ein Parser ist ein Programm, das Sätze, Teilstrukturen oder Wörter analysiert, auf ihre Korrektheit prüft und als Ausgabe eine Strukturbeschreibung liefert. Ein Parsingalgorithmus baut auf einem Erkennungsalgorithmus auf. Die Aufgabe eines Erkennungsalgorithmus ist es, für eine Folge von Symbolen zu entscheiden, ob sie zur Sprache L(G), die durch die Grammatik G beschrieben wird, gehört. Ein Parsingalgorithmus macht das gleiche, geht aber noch einen Schritt weiter und liefert eine Strukturbeschreibung der Folge [NL94, S. 19]. Eine solche Folge von Symbolen kann zum Beispiel ein natürlichsprachlicher Satz sein. Zusätzlich muss noch eine beschreibende Grammatik G vorliegen, für die entschieden wird, ob der Satz mit dieser erzeugt werden kann. Ein Parser gibt dann zusätzlich als Strukturbeschreibung zum Beispiel einen Syntaxbaum bzw. Ableitungsbaum des Satzes aus.
Das Parsen natürlichsprachlicher Sätze ist im Vergleich zum Parsen formaler Sprachen, wie zum Beispiel Programmiersprachen, deutlich aufwändiger. Denn es muss mit Ambiguitäten, einer großen Strukturvielfalt und einer erheblich höheren Komplexität umgegangen werden [NL94, S. 14].

Zur Klassifikation von Parsingalgorithmen werden in der Regel drei Kriterien betrachtet [NL94, S. 22]:

1. Verarbeitungsrichtung,
2. Analyserichtung und
3. Suchstrategie.

Die Verarbeitungsrichtung kann unidirektional oder bidirektional sein. Unter den Begriff unidirektionale Verarbeitungsrichtung fällt die inkrementelle Verarbeitung von links nach rechts oder von rechts nach links. Bei einer bidirektionale Verarbeitung wird von zwei Richtungen gleichzeitig analysiert. Die Analyse kann zum Beispiel gleichzeitig am Satzende und Satzanfang beginnen, oder in der Mitte des Satzes und dann zu beiden Seiten laufen [NL94, S. 22].

Das zweite Kriterium ist die Analyserichtung. Die meisten Parser arbeiten entweder bottom-up oder top-down. Selten kann es auch Mischformen der beiden geben. Bottom-up bedeutet, dass der Ausgangspunkt der Analyse der zu analysierende Satz ist. Durch Anwendung der Regeln der Grammatik als Reduktion, d. h. die Elemente der rechten Regelseite werden durch das Symbol der linken Regelseite ersetzt, wird eine Ableitung des Satzes gesucht. Das Verfahren terminiert, wenn der Satz auf das Startsymbol S reduziert wurde. Top-down Analysen beginnen dagegen beim Startsymbol und finden eine Ableitung des Satzes durch Expansion der Regeln, d. h. die linke Seite der Regel wird ersetzt durch die rechte [CEE+04, S. 254].
In Abbildung 2.23 bzw. Abbildung 2.24 wird der Satz „Der Hund bellt.“, der ein Wort der Sprache der in Abbildung 2.22 dargestellten Beispielgrammatik ist, von links nach rechts und bottom-up bzw. top-down geparst.

Das Parsen besteht zu einem Teil aus Suchprozessen. Immer wenn es verschiedene Analysemöglichkeiten gibt, wird eine Suchtstrategie benötigt um die Möglichkeiten strategisch durchzugehen [NL94, S. 22]. Die bekanntesten Suchstrategien sind die Tiefen- und die Breitensuche. Aber auch eine Best-First-Suche oder Beam-Suche ist möglich [CEE+04, S. 255]. Für eine ausführliche Beschreibung der Strategien sei auf [RN04] verwiesen.

Eine weitere Einteilung der Parsingalgorithmen könnte nach der Grammatikkategorie (siehe Chomsky-Hierarchie in Kapitel 2.3.1), die von ihnen akzeptiert wird, geschehen. Im Folgenden werden aber nur Parsingalgorithmen für kontextfreie Sprachen betrachtet, da von einer Phrasenstrukturgrammatik, die nach dem X-Bar-Schema aufgebaut ist (siehe Kapitel 2.3.2), ausgegangen wird.

Ein Problem der elementaren Parsingverfahren, wie zum Beispiel einfacher Bottom-Up- oder Top-Down-Parser, ist, dass Teilstrukturen mehrfach analysiert werden. Dies macht sie sehr ineffizient [NL94, S. 102]. Chart-Parser, die auf kontextfreien Sprachen arbeiten, speichern die schon analysierten Teile und können diese wiederverwenden. Das nächste Unterkapitel beschreibt das Verfahren des Chart-Parsings näher.

Beispielgrammatik 3	
S → NP VP	Det → der
NP → Det N	N → Hund
VP → V	V → bellt

Abbildung 2.22.: Beispielgrammatik 3

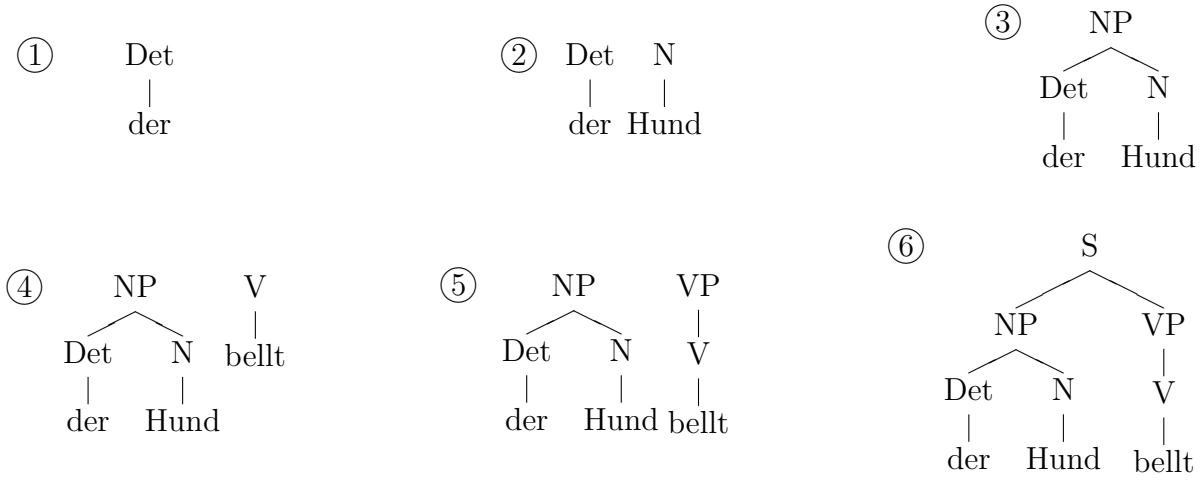

Abbildung 2.23.: Aufbau einer Strukturbeschreibung bottom-up, links-rechts nach [CEE+04, S. 255]

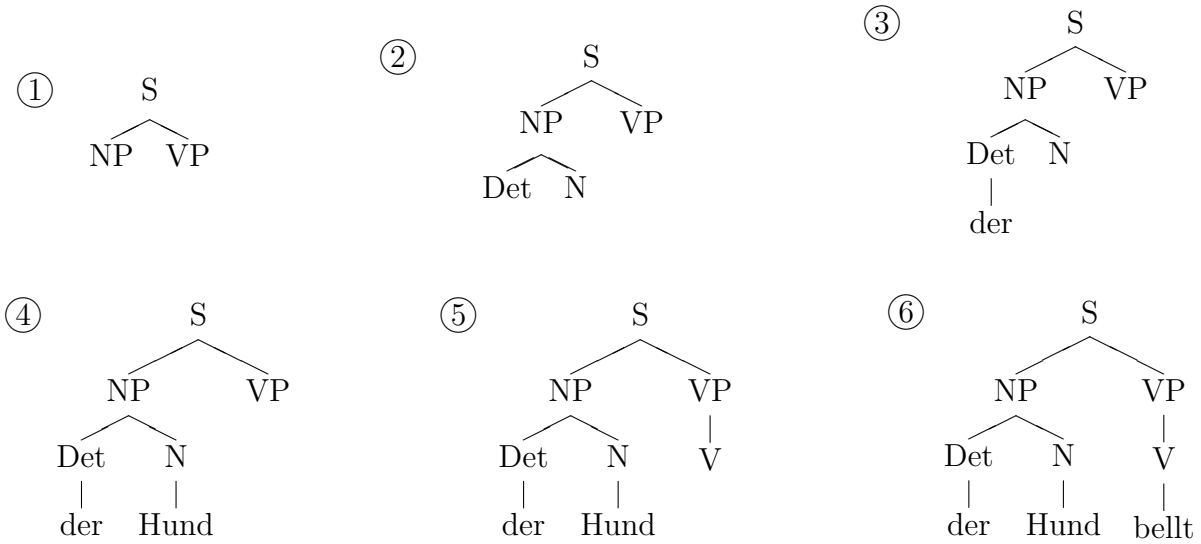

Abbildung 2.24.: Aufbau einer Strukturbeschreibung top-down, links-rechts nach [CEE+04, S. 254]

2.4.2. Chart-Parsing

Ein Chart-Parser speichert Teilresultate in einer Datenstruktur, der sogenannten Chart. Dieser Vorgang entspricht dem Prinzip des dynamischen Programmierens. Durch Nachschlagen der Teilresultate werden unnötige Rekursionen vermieden. Chart-Parsing kann als Top-Down-Chart-Parsing, Bottom-Up-Chart-Parsing, etc. realisiert werden [CEE+04, S. 267].
Die bekanntesten Chart-Parser-Algorithmen sind der Earley-Algorithmus und der Cocke-Younger-Kasami-Algorithmus (CYK-Algorithmus) [CEE+04, S. 267]. Die Grammatik für den CYK-Algorithmus muss in Chomsky-Normalform[3] vorliegen. Jede epsilonfreie Grammatik kann in Chomsky-Normalform umgeformt werden, dies bedeutet aber zusätzlichen Aufwand und die Grammatik würde nicht mehr dem X-Bar-Schema entsprechen. Außerdem arbeitet der CYK-Algorithmus ausschließlich bottom-up. Dadurch kann es vorkommen, dass Konstituenten gebildet werden, ohne dass klar ist, ob diese zu größeren Konstituenten weiter kombiniert werden können. Der Vorteil des Earley-Parsers hingegen ist, dass beliebige kontextfreie Grammatiken genutzt werden können. Er kommt sowohl mit Tilgungsregeln, als auch mit zyklischen Regeln und links- bzw. rechts-rekursiven Regeln zurecht. Außerdem arbeitet er sowohl bottom-up als auch top-down. Das Weiterverfolgen von nicht erfolgsversprechenden Konstituenten wird weitgehend vermieden [NL94, S. 267]. Deswegen wird im Folgenden nur das Prinzip des Earley-Parsers beschrieben, das in der vorliegenden Arbeit verwendet werden soll.

Eine Chart wird meistens als Vektor, Matrix oder azyklischer Graph dargestellt [NL94, S. 103]. Ein Eintrag in der Chart wird Item oder Kante genannt. Er beinhaltet unter anderem immer den Abschnitt des Satzes, auf den sich das Item bezieht, und die Regel der Grammatik, die angewandt wurde. Im zu analysierenden Satz werden die Zwischenräume der Wörter von 0 bis n nummeriert, so dass der Satzabschnitt durch zwei Zahlen angegeben werden kann. Die erste Zahl entspricht dem Anfang und die zweite Zahl dem Ende des Abschnitts [CEE+04, S. 266].
Soll zum Beispiel der Satz „der Hund bellt“ analysiert werden, wird er wie folgt annotiert: ${}_0Der_1Hund_2bellt_3$. Verwendet der Chart-Parser die Grammatik in Abbildung 2.22, kann ein Eintrag in einer entsprechenden Chart, der den Satzabschnitt 0 bis 2 als NP analysiert, folgendermaßen aussehen: 0 2 NP → Det N

Chart-Parser können in aktive und passive Chart-Parser unterteilt werden. Bei aktiven Chart-Parsern wird die Regel eines Items in zwei Abschnitte, einen aktiven und einen inaktiven, geteilt. Trennzeichen ist ein Punkt („dot“). Diese Items werden auch „dotted Items“ genannt. Alle Kategorien vor dem Punkt sind im passiven Abschnitt und alle nach dem Punkt im aktiven. Durch die Trennung kann eine Aussage über den Analysezustand getroffen werden. Sieht ein Charteintrag wie folgt aus: 0 1 NP → Det • N, dann bedeutet dies, dass das erste Wort als Artikel erkannt wurde und anschließend ein Nomen erwartet wird. Die Bereiche rechts oder links des Punktes können auch leer sein. Ist der Punkt ganz

[3] vgl. [Sch03a] S. 52

am Ende einer Regel, wird dieser Itemeintrag auch passive Kante genannt. Alle anderen Kanten sind aktive Kanten, diese müssen noch vervollständigt werden [CEE+04, S. 267].

Earley Parser basieren auf drei Basisoperationen: der Expansion der Regeln (EXPAND), der Verarbeitung eines Terminalsymbols (SCAN) und der Kombination zweier Einträge (COMPLETE).

Die Basisoperationen sind definiert, wie in Abbildung 2.25, Abbildung 2.26 und Abbildung 2.27, dargestellt. Es gilt $\alpha, \beta \in (V \cup \Sigma)^*$ und $\gamma \in (V \cup \Sigma)^+$.

EXPAND (top-down)	
Existiert in der Chart eine Kante der Form	i j A $\rightarrow\alpha \bullet B\beta$ mit i≤j,
dann wird für jede Grammatikregel	B $\rightarrow \gamma$
ein neues Item der Form	j j B $\rightarrow \bullet\gamma$
in die Chart eingefügt.	

Abbildung 2.25.: Expand [CEE+04, S. 268]

SCAN
Existiert in der Chart ein Item der Form i j-1 A $\rightarrow\alpha \bullet w_j\beta$ mit i≤j-1
und ist w_j das j-te Wort der Eingabekette $w = w_1w_2...w_n$ mit $1 \leq n$,
dann wird ein neues Item der Form i j A $\rightarrow\alpha w_j \bullet \beta$
in die Chart eingefügt.

Abbildung 2.26.: Scan [CEE+04, S. 269]

COMPLETE (bottom-up)	
Enthält die Chart ein Item der Form	i j A $\rightarrow\alpha \bullet B\beta$ mit i≤j
und eine weiteres Item der Form	j k B $\rightarrow\gamma\bullet$ mit j≤k,
dann wird ein neues Item der Form	i k A $\rightarrow\alpha B \bullet \beta$
in die Chart eingefügt.	

Abbildung 2.27.: Complete [CEE+04, S. 270]

Der Algorithmus eines Earley Parser kann ganz allgemein wie in Abbildung 2.28 formuliert werden.

ALGORITHMUS

Eingabe: Eine Eingabekette $w = w_1 w_2 ... w_n$ mit $1 \leq n$

1. Füge 0 0 S' → • S zur Initialisierung in die Chart ein (S entspricht dem Startsymbol der Grammatik, S' ist ein Nichtterminalsymbol, das nicht in der Grammatik vorkommt)
2. Wende EXPAND, SCAN und COMPLETE solange an, bis keine weiteren Charteinträge erzeugt werden können.

Ausgabe: JA, falls 0 n S' → S • in der Chart, sonst NEIN

Abbildung 2.28.: Earley Algorithmus

Der vorgestellte Algorithmus ist nur ein Erkenner, aber aus der Chart kann eine Strukturbeschreibung extrahiert werden.

Für die Beispielgrammatik in Abbildung 2.29 und den Satz „der Hund bellt“ soll eine Chart nach dem Schema in Abbildung 2.28 erstellt werden. Der Ablauf wird in Tabelle 2.2 beschrieben.

Beispielgrammatik 4

S → NP VP	Det → der
NP → Det N	Det → die
VP → V	N → Hund
VP → V NP	N → Katze
	V → bellt
	V → sieht

Abbildung 2.29.: Beispielgrammatik 4 [CEE+04, S. 236]

Nr.	**Item**		**Begründung**
1	0 0	S' → • S	Initialisierung
2	0 0	S → • NP VP	EXPAND 1.
3	0 0	NP → • Det N	EXPAND 2.
4	0 0	Det → • der	EXPAND 3.
5	0 0	Det → • die	EXPAND 3.
6	0 1	Det → der •	SCAN 4.
7	0 1	NP → Det • N	COMPLETE 3. mit 6.
8	1 1	N → • Hund	EXPAND 7.
9	1 1	N → • Katze	EXPAND 7.
10	1 2	N → Hund •	SCAN 8.
11	0 2	NP → Det N •	COMPLETE 7. mit 10.
12	0 2	S → NP • VP	COMPLETE 2. mit 11.
13	2 2	VP → • V	EXPAND 12.
14	2 2	VP → • V NP	EXPAND 12.
15	2 2	V → • bellt	EXPAND 13. oder 14.
16	2 2	V → • sieht	EXPAND 13. oder 14.
17	2 3	V → bellt •	SCAN 15.
18	2 3	VP → V •	COMPLETE 13. mit 17.
19	2 3	VP → V • NP	COMPLETE 14. mit 17.
20	0 0	S → NP VP •	COMPLETE 12. mit 18.

Tabelle 2.2.: Chart für den Satz „Der Hund bellt.“ [CEE+04, S. 272]

3. Stand der Technik

Dieses Kapitel soll einen Einblick in den aktuellen Markt der Übersetzungssysteme bieten und abwägen, ob diese als Grundlage für diese Arbeit verwendet werden können.

Übersetzungsprogramme gibt es mittlerweile viele. Die bekanntesten Systeme sind der Google Translator[1] und Yahoo Babelfish[2]. Der Google Translator (siehe Abbildung 3.1) wird sehr häufig als Basis für andere frei verfügbare Übersetzungssysteme genutzt. Er basierte viele Jahre auf dem kommerziellen System SYSTRAN[3], mittlerweile hat Google aber ein eigenes statistisches System entwickelt, das sogar Benutzerinteraktionen zulässt. Der Benutzer kann zwischen Übersetzungsoptionen wählen, oder eine eigene Übersetzung eingeben.

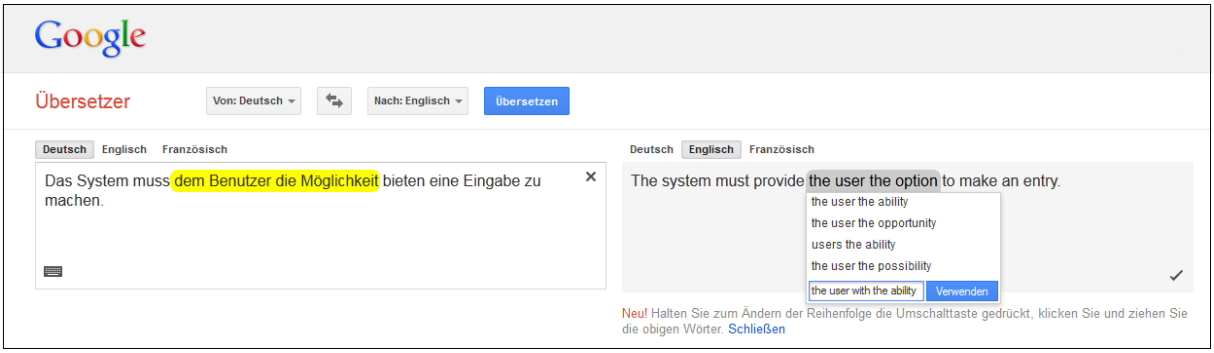

Abbildung 3.1.: Google Translator (http://translate.google.de)

Yahoo Babelfish (siehe Abbildung 3.2) dagegen bietet dem Benutzer nur die Möglichkeit die Übersetzung zu editieren, macht aber keine Alternativvorschläge. Es basiert immer noch auf SYSTRAN. SYSTRAN ist eins der bekanntesten kommerziellen Systeme, das zunächst als regelbasiertes System entwickelt und mittlerweile zu einem Hybridsystem, bestehend aus einem regelbasierten und einem statistischen System, erweitert wurde.
Ein bekanntes nichtkommerzielles maschinelles Übersetzungssystem ist GramTrans[4]. Auch

[1] http://translate.google.de
[2] http://babelfish.yahoo.com
[3] http://www.systran.de
[4] http://gramtrans.com

dies war ein regelbasiertes, genauer sogar ein transferbasiertes System und wurde mittlerweile weiterentwickelt zu einem Hybridsystem. Es beinhaltet aber keine Übersetzung von Deutsch nach Englisch.

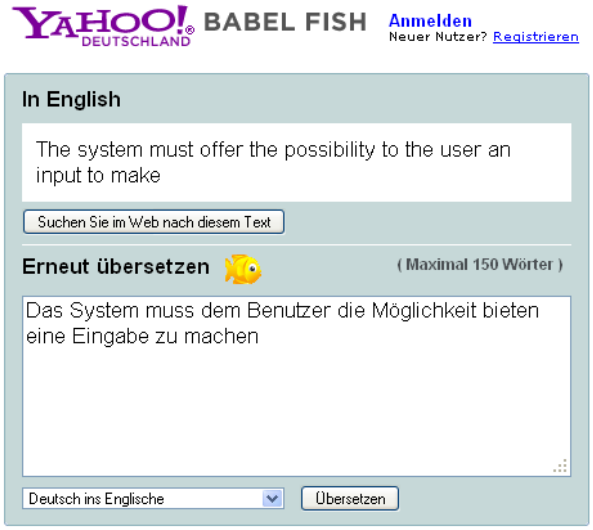

Abbildung 3.2.: Yahoo Babelfish (http://babelfish.yahoo.com)

Die meisten Systeme auf dem Markt sind multilingual und bidirektional. Häufig werden als maschinelle Übersetzungssysteme auch Systeme aufgelistet, die dem Anwender nur Wörterbücher zur Verfügung stellen. Diese Systeme sind damit streng genommen aber keine Übersetzungssysteme.
Auf dem Markt existiert kein System, das ausschließlich für Anforderungen (Requirements) entwickelt wurde. Eine Übersetzung könnte aber durch die Benutzung einer Subsprache optimiert werden. Andere Systeme erzielen bei der Übersetzung weniger gute Ergebnisse. Zum aktuellen Zeitpunkt gibt es nur ein System zur Unterstützung bei der Übersetzung von Anforderungen vom Arabischen ins Englische und umgekehrt. Das System heißt ARAT[5] und erweitert das RAT-System. Das RAT-System unterstützt den Anwender beim Schreiben von Anforderungen durch das Bereitstellen von Schablonen und analysiert die geschriebenen Anforderungen. ARAT realisiert aber keine automatische maschinelle Übersetzung, sondern es gehört in die Kategorie MAHT (siehe Kapitel 2.2.1). Es analysiert den zu übersetzenden Satz und zerlegt ihn dazu in Phrasen. Auf der Syntaxebene gibt es ein Mapping zwischen der arabischen und der englischen Syntax. So kann ein arabischer Satz, nachdem er zerlegt wurde, in die englische Syntax transferiert werden. Die eigentliche Übersetzung wird vom Anwender geleistet, indem er die einzelnen Phrasen manuell übersetzt.
Das ARAT-System ist wegen der manuellen Übersetzung kein maschinelles Übersetzungssystem. Ein solches System ist aber Ziel dieser Arbeit. Die Benutzung des Google Trans-

[5] vgl. [Ela11]

lators würde sich anbieten, da dieser schon sehr gute Ergebnisse liefert, aber der Benutzer bräuchte zu viele Kenntnisse über die Schablonen und die Subsprache, um die Übersetzung editieren zu können, als dass er ihn sinnvoll einsetzen könnte. Eine Möglichkeit wäre es den Google Translator in die eigene Software zu integrieren und die Ausgabe maschinell anzupassen. Da der Google Translator aber nur online verfügbar ist und die zu übersetzenden Daten an Google geschickt werden müssten, ist dies keine Alternative. Besonders bei Spezifikationen für Neuentwicklungen ist es nicht erwünscht geheime Daten an Dritte weiterzugeben.
Da für die Übersetzung von Anforderungen bis jetzt kein System existiert, die Qualität der allgemeinen maschinellen Übersetzungssysteme nicht überzeugt, sowie auch nicht für schablonenbasierte Anforderungen ausgelegt oder nur online verfügbar ist, lohnt es sich im Rahmen dieser Arbeit ein neues System zu entwickeln und nicht auf vorhandene zurückzugreifen.

4. Konzept

In diesem Kapitel soll das Konzept des im Rahmen dieser Arbeit zu entwickelnden Werkzeugs beschrieben werden. Zunächst wird in Abschnitt 4.1 ein Verfahren zur maschinellen Übersetzung ausgewählt. Die Übersetzung wird für die in Kapitel 2.1.2 vorgestellten Schablonen entwickelt. Dazu ist eine Grammatik für die deutschen, ebenso wie für die englischen Anforderungen nötig. Diese werden in Abschnitt 4.2 vorgestellt. Anschließend werden die einzelnen durchzuführenden Schritte der Übersetzung in den Abschnitten 4.3.1, 4.3.2 und 4.3.3 näher spezifiziert.

4.1. Auswahl eines Verfahrens zur maschinellen Übersetzung

In Kapitel 2.2.2 werden verschiedene Verfahren zur maschinellen Übersetzung vorgestellt. Nicht jedes Verfahren ist für jeden Anwendungszweck geeignet. Deswegen muss für die gegebenen Voraussetzungen ein Verfahren ausgewählt werden.
Das schnellste und einfachste Verfahren ist die direkte Übersetzung. Allerdings gibt es mittlerweile Verfahren, die zum Beispiel durch eine strukturelle Analyse des Satzes, eine bessere Übersetzung erzielen. Es ist möglich eine solche Analyse des zu übersetzenden Satzes zu realisieren. Damit kommt eine transferbasierte Übersetzung in Frage. Betrachtet werden muss noch die Tiefe und Art der Analyse.
Auch für eine Interlingua wird der Satz detailliert analysiert. Allerdings kann dieses Verfahren ausgeschlossen werden, da es nicht möglich ist eine vollständige Interlingua zu modellieren. Es kann zwar von einer Subsprache ausgegangen werden, diese ist aber im Bezug auf die Semantik nicht hinreichend eingeschränkt, da sie ständig um neue Begriffe erweitert wird. Selbes gilt für den neueren Ansatz der wissensbasierten Übersetzung. Das notwendige Wissen, um Ambiguitäten aufzulösen, oder andere Übersetzungsprobleme zu vermeiden, kann bislang und im Speziellen in dieser Arbeit nicht vollständig erfasst werden.
Die beispielbasierte oder statistische Übersetzung benötigt unter anderem ein großes paralleles, bilinguales Korpus, bestehend aus schablonenbasierten Anforderungen. Dies ist nicht vorhanden, da die bisher verfassten Anforderungsdokumente ohne Schablonen geschrieben wurden und nicht für jedes eine Übersetzung vorhanden ist. Es existieren andere parallele Korpora, die allerdings auch nicht geeignet sind, weil die Struktur und Art der

Sätze bei allgemeinsprachlichen Texten oder auch anderen Domänen verschieden ist. Satzteile aus diesen Korpora für eine Übersetzung zu verwenden führt nicht zum gewünschten Ergebnis, da die übersetzten Sätze dann nicht der englischen Schablone entsprechen.
Ein Hybridsystem, das ein regelbasiertes und ein beispielbasiertes oder statistisches Verfahren kombiniert, ist im Rahmen dieser Arbeit nicht möglich. Es stellt aber eine Erweiterungsmöglichkeit dar.

Nach Betrachtung der verschiedenen Verfahren kommt nur ein transferbasiertes maschinelles Übersetzungssystem in Frage. Ein Konzept dazu wird in den folgenden Kapiteln erläutert.

4.2. Grammatik der Anforderungen

Das Werkzeug beschränkt sich auf die Übersetzung von funktionalen Anforderungen. Für diese gibt es Schablonen, die in Kapitel 2.1.2 erläutert sind. Es wird eine Grammatik für diese Schablonen benötigt, um die Sätze analysieren und übersetzen zu können. Hoedoro analysiert in seinem Werkzeugl Anforderungen, die nach eben genannten Schablonen aufgebaut sind [Hoe11]. Er benutzt dazu allerdings einen deterministischen Automaten. Deterministische Automaten können nur Grammatiken vom Typ 3 verwenden (siehe Chomsky-Hierarchie Kapitel 2.3.1), allerdings kann mit diesen nur eine sehr flache Analyse erreicht werden. Für eine Übersetzung ist aber, wie in Abschnitt 4.1 motiviert, eine etwas tiefere Analyse sinnvoll. Dies ist mit einer Phrasenstrukturgrammatik, deren Prinzip in Kapitel 2.3.2 vorgestellt wird, möglich. Dadurch können Strukturinformationen für eine bessere Übersetzung genutzt werden. Die Grammatik von Hoedoro kann daher nicht verwendet werden. Allerdings können als Ansatz für die Entwicklung einer Phrasenstrukturgrammatik die von Hoedoro beschriebenen Phrasen[1], die er als Basis für seinen Automat benutzt, verwendet werden.
Im Folgenden wird eine kontextfreie Phrasenstrukturgrammatik für ein transferbasiertes maschinelles Übersetzungssystem vorgestellt, mit der Sätze, deren Struktur den Schablonen entpricht, (re-)konstruiert werden können. Diese Grammatik besteht aus Phrasenstrukturregeln und lexikalischen Regeln. Es wird sowohl eine Grammatik für deutsche als auch für englische Anforderungen vorgestellt.

4.2.1. Phrasenstrukturregeln

Zunächst wird eine Basisgrammatik verwendet, die Anforderungen ohne Vorbedingungen abdeckt. Dies entspricht der Schablone in Abbildung 2.5. Zusätzlich gilt, dass der Systemname nur aus einem bestimmten oder unbestimmten Artikel und einem Nomen be-

[1] vgl. [Hoe11] S. 33–41

stehen darf. Ein Objekt und dessen Ergänzung ist optional. In einer Benutzerinteraktion oder Schnittstellenanforderung ist das Objekt, wenn es vorhanden ist, eine Nominalphrase (NP) oder eine Präpositionalphrase (PP). Mögliche Strukturen sind in Abbildung 4.1 dargestellt. Bei einer selbstständigen Systemaktivität kann das Objekt komplexer aufgebaut sein und aus zwei Phrasen bestehen. Die möglichen Strukturen eines Objekts mit dem folgenden Prozesswort sind in Abbildung 4.2 dargestellt. Für das „X“ kann eine Nominalphrase oder eine Präpositionalphrase eingesetzt werden.

Abbildung 4.1.: Mögliche Phrasenstrukturen für ein Objekt und das folgende Prozesswort einer Benutzerinteraktion oder Schnittstellenanforderung

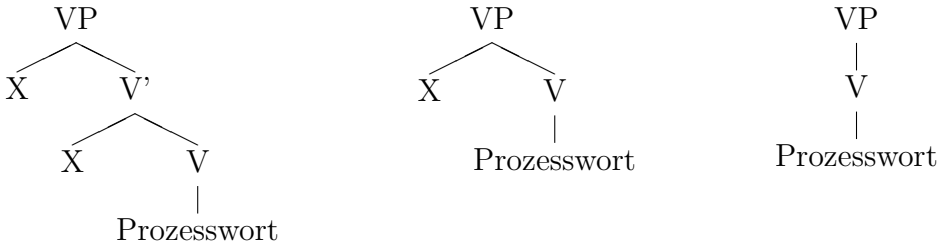

Abbildung 4.2.: Mögliche Phrasenstrukturen für ein Objekt und das folgende Prozesswort einer selbstständigen Systemaktivität

Die Basisgrammatik für deutsche Anforderungen ist in Abbildung 4.3, die Basisgrammatik für englische Anforderungen ist in Abbildung 4.4 dargestellt. Diese Grammatiken können um zusätzliche Komponenten erweitert werden, um noch mehr Sätze abzudecken. Sowohl mit der in Abbildung 4.3 als auch mit der in Abbildung 4.4 dargestellten Grammatik lassen sich drei verschiedene Typen von Anforderungen beschreiben. Abbildung 4.5 zeigt die erste Möglichkeit, eine deutsche Anforderung, die eine selbstständige Systemaktivität beschreibt. Abbildung 4.6 ist die dazugehörige Übersetzung auf Englisch.
Der zweite Typ Anforderung ist eine Benutzerinteraktion. Ein Beispiel für das Deutsche ist in Abbildung 4.7 gegeben. Die englische Übersetzung ist in Abbildung 4.8 dargestellt.
Abbildung 4.9 zeigt ein Beispiel für den letzten Typ Anforderung, eine Schnittstellenanforderung. Die entsprechende Übersetzung ist in Abbildung 4.10 dargestellt.
Variieren kann bei diesen drei Typen der Systemname, die rechtliche Verbindlichkeit (Hilfsverb), der Benutzer, das Prozesswort (Verb) und das damit eventuell verbundene subkategorisierte Objekt.
Aufgebaut sind die Grammatiken nach dem in Kapitel 2.3.2 erläuterten X-Bar-Schema.

Basisphrasenstrukturregeln Deutsch	
IP → NP I’	VP → VP IP
IP → NP I	VP → NP V’
IP → PP I	VP → PP V’
IP → I	VP → V
NP → Det N	VP → NP V
PP → P NP	VP → PP V
I’ → I VP	V’ → NP V
	V’ → PP V

Abbildung 4.3.: Basisphrasenstrukturregeln Deutsch

Basisphrasenstrukturregeln Englisch	
IP → NP I’	VP → VP IP
IP → I NP	VP → V’ NP
IP → I PP	VP → V’ PP
IP → I	VP → V
NP → Det N	VP → V NP
PP → P NP	VP → V PP
I’ → I VP	V’ → V NP
	V’ → V PP

Abbildung 4.4.: Basisphrasenstrukturregeln Englisch

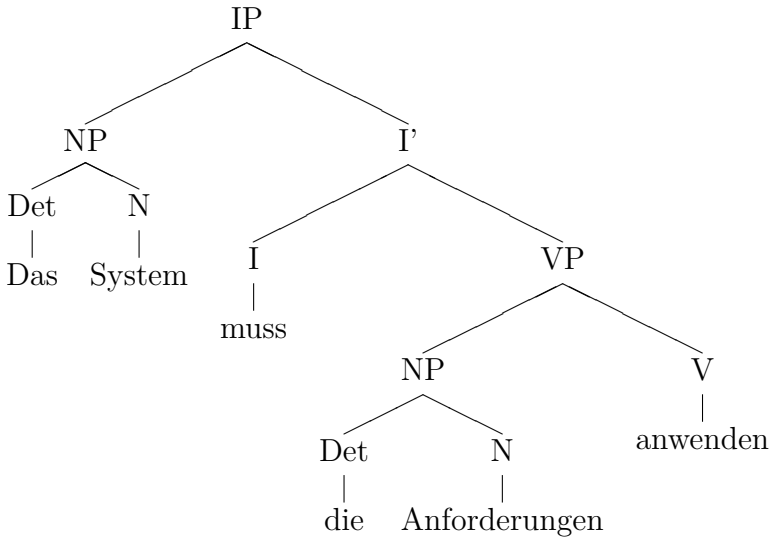

Abbildung 4.5.: Selbstständige Systemaktivität auf Deutsch

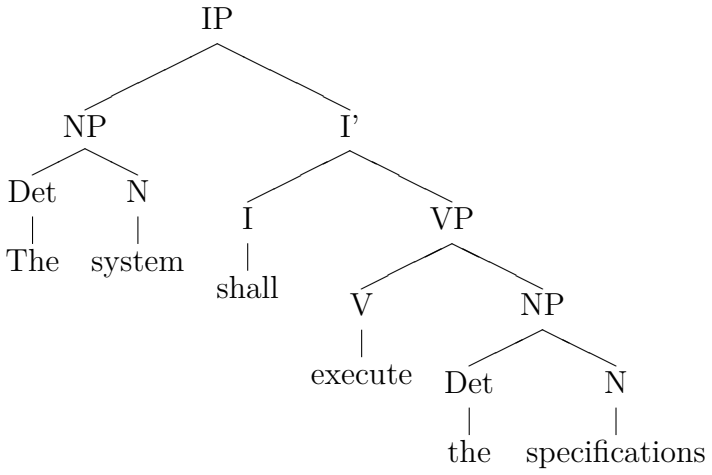

Abbildung 4.6.: Selbstständige Systemaktivität auf Englisch

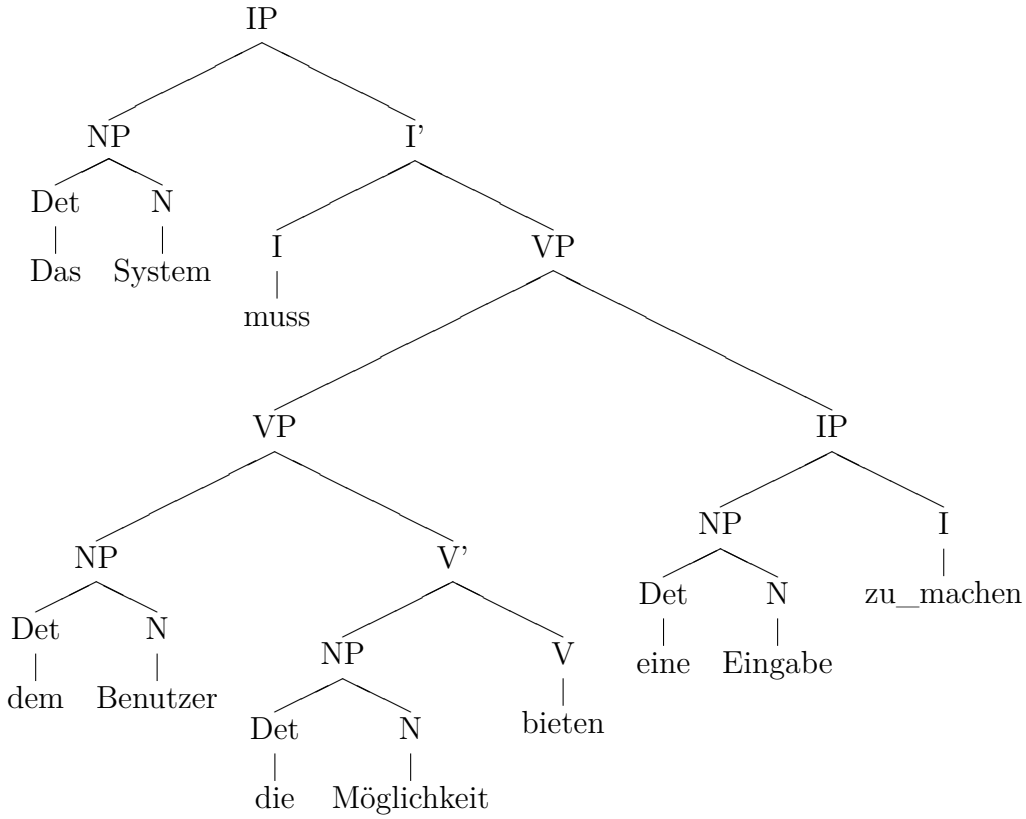

Abbildung 4.7.: Benutzerinteraktion auf Deutsch

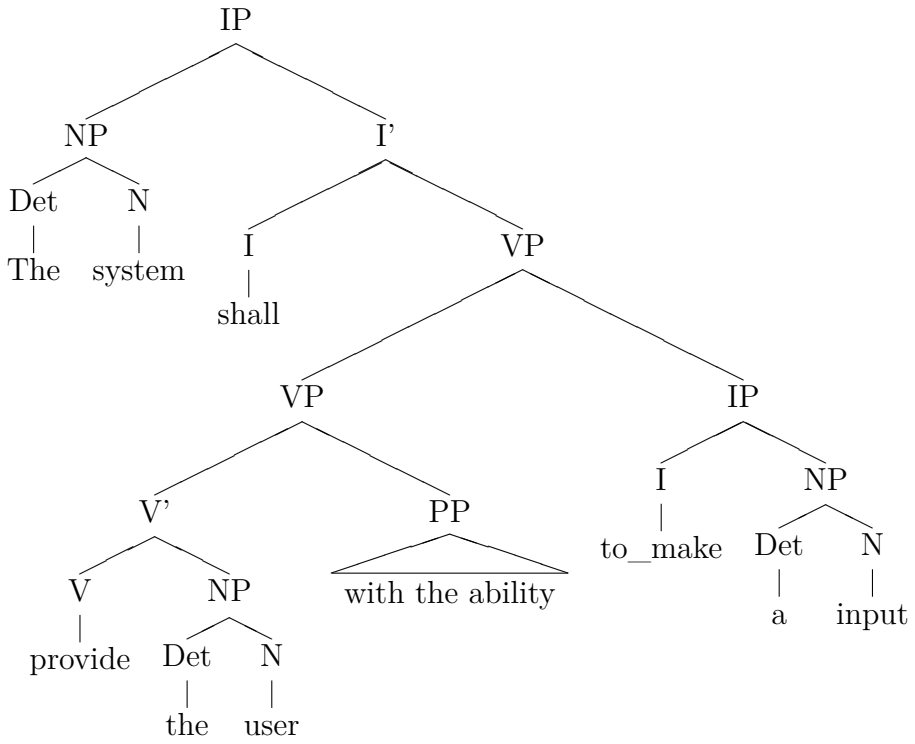

Abbildung 4.8.: Benutzerinteraktion auf Englisch

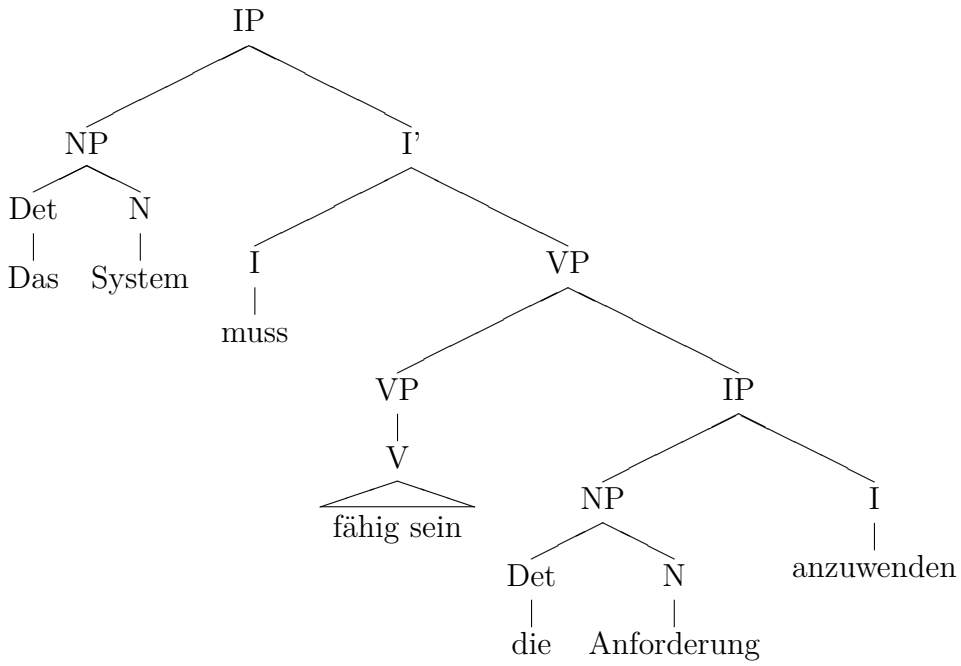

Abbildung 4.9.: Schnittstellenanforderung auf Deutsch

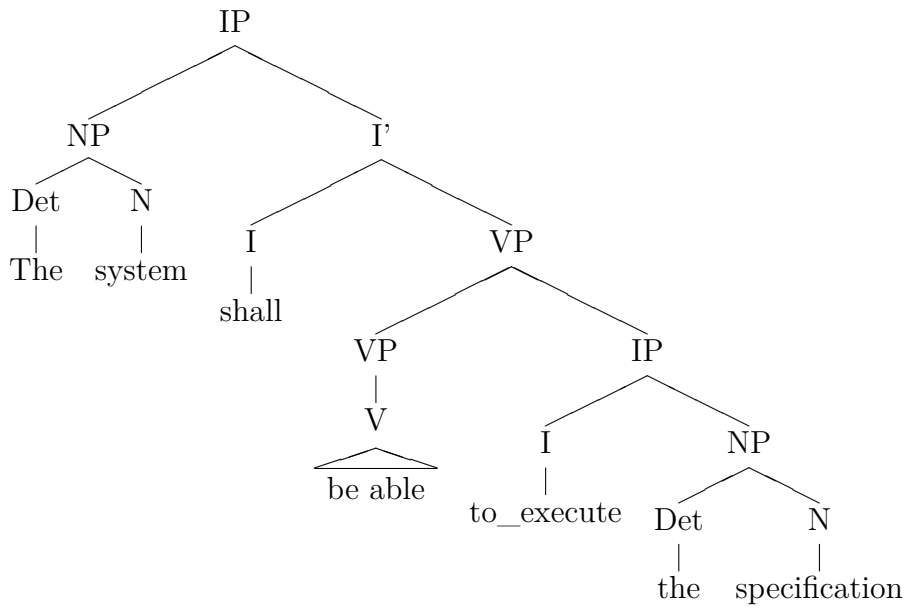

Abbildung 4.10.: Schnittstellenanforderung auf Englisch

Die Grammatiken aus Abbildung 4.3 und Abbildung 4.4 generieren über, d. h. es können auch Sätze konstruiert werden, die nicht erwünscht sind, wie zum Beispiel der Satz in Abbildung 4.11. Bei diesem Satz verlangt der hier angenommene Subkategorisierungsrahmen des Verbs „bieten“ (<np:nom, np:dat, np:akk, ip:inf>) noch einen Infinitivsatz (ip:inf). Der Subkategorisierungsrahmen ist nicht vollständig gesättigt und der Satz daher ungrammatisch.

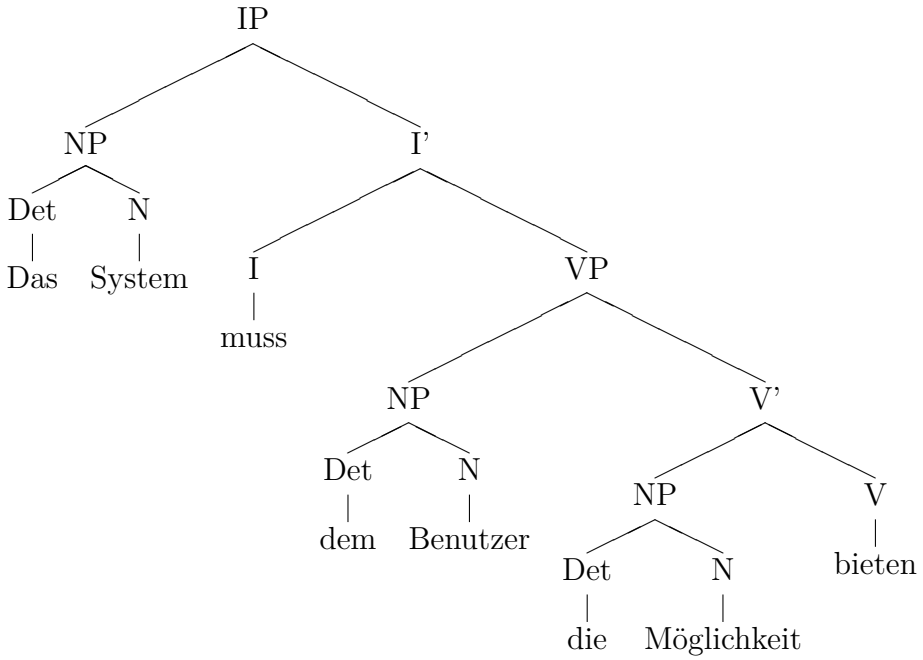

Abbildung 4.11.: Übergenerierter Satz der Basisgrammatik

Mit Merkmalsstrukturen (siehe Kapitel 2.3.3) und Restriktionen bezüglich dieser Merkmalsstrukturen kann ein solches Übergenerieren verhindert werden. Es genügt Merkmalsstrukturen und Restriktionen für die deutsche Basisgrammatik anzugeben. Die Merkmalsstrukturen für die verschiedenen Wortarten werden in Abschnitt 4.2.2 beschrieben. Die Restriktionen für die Phrasenstrukturreglen sind nach folgendem Schema aufgebaut:

$$\underset{\begin{bmatrix} M1:W1 \\ M2:W2 \end{bmatrix}}{XP} \rightarrow \underset{\begin{bmatrix} M1:W1 \\ M3:W3 \\ M2:W2 \end{bmatrix}}{YP} \quad \underset{\begin{bmatrix} M1:W1 \\ M3:W3 \\ M4:W4 \end{bmatrix}}{X'}$$

Sei XP → YP X' eine beliebige Regel. Über die Merkmalsstrukturen kann verlangt werden, dass bestimmte Merkmale (M) vorhanden sein müssen und deren Werte (W) übereinstimmen oder dass ein Merkmal einen bestimmten Wert hat. In dieser Grammatik werden nur gleiche Merkmale miteinander verglichen.

Eine Merkmalsstruktur besteht immer aus Merkmal-Wert-Paaren (M:W). Bei den Restriktionen kann als Wert ein exakter Wert verlangt oder eine Variable angegeben werden. Über eine Variable wird sichergestellt, dass die Werte der Merkmale verschiedener Kategorien übereinstimmen. Das Merkmal *M1* von XP muss beispielsweise den gleichen Wert haben wie das Merkmal *M1* von YP und das Merkmal *M1* von X'. Selbes gilt für das Merkmal *M2* von XP das den gleichen Wert haben muss, wie das Merkmal *M2* von YP. X' hat kein Merkmal *M2*. Der Eintrag *M4 : W4* macht beispielsweise nur Sinn, wenn *W4* ein Wert ist und keine Variable, da das Merkmal *M4* bei keiner anderen Kategorie vorkommt.

Die deutsche Basisgrammatik sieht mit Restriktionen bezüglich der Merkmalsstrukturen folgendermaßen aus:

$$IP \rightarrow \underset{\begin{bmatrix} NUM:N \\ PERS:3 \\ KAS:nom \end{bmatrix}}{NP} \quad \underset{\begin{bmatrix} NUM:N \\ PERS:3 \\ FIN:+ \end{bmatrix}}{I'}$$

$$\underset{\begin{bmatrix} FIN:F \\ ZU:Z \\ SUBKAT:X_1X_2 \end{bmatrix}}{IP} \rightarrow \underset{\begin{bmatrix} KAS:K \end{bmatrix}}{NP} \quad \underset{\begin{bmatrix} FIN:F \\ ZU:Z \\ SUBKAT:X_1(np:K)X_2 \end{bmatrix}}{I}$$

$$\underset{\begin{bmatrix} FIN:F \\ ZU:Z \\ SUBKAT:X_1X_2 \end{bmatrix}}{IP} \rightarrow \underset{\begin{bmatrix} KAS:K \\ PREP:P \end{bmatrix}}{PP} \quad \underset{\begin{bmatrix} FIN:F \\ ZU:Z \\ SUBKAT:X_1(pp:P\_K)X_2 \end{bmatrix}}{I}$$

$$\underset{\begin{bmatrix} FIN:F \\ ZU:Z \\ SUBKAT:S \end{bmatrix}}{IP} \rightarrow \underset{\begin{bmatrix} FIN:F \\ ZU:Z \\ SUBKAT:S \end{bmatrix}}{I}$$

$$\underset{\begin{bmatrix} NUM:N \\ PERS:P \\ KAS:K \\ GEND:G \end{bmatrix}}{NP} \rightarrow \underset{\begin{bmatrix} NUM:N \\ PERS:P \\ KAS:K \\ GEND:G \end{bmatrix}}{Det} \quad \underset{\begin{bmatrix} NUM:N \\ PERS:P \\ KAS:K \\ GEND:G \end{bmatrix}}{N}$$

$$\underset{\begin{bmatrix} KAS:K \\ PREP:P \end{bmatrix}}{PP} \rightarrow \underset{\begin{bmatrix} KAS:K \\ PREP:P \end{bmatrix}}{P} \quad \underset{\begin{bmatrix} KAS:K \end{bmatrix}}{NP}$$

$$I' \rightarrow I \quad VP$$
$$\begin{bmatrix} NUM:N \\ PERS:P \\ FIN:F \end{bmatrix} \quad \begin{bmatrix} NUM:N \\ PERS:P \\ FIN:F \end{bmatrix} \quad \begin{bmatrix} FIN:- \\ ZU:- \\ SUBKAT:np:nom \end{bmatrix}$$

$$VP \rightarrow VP \quad IP$$
$$\begin{bmatrix} FIN:F \\ ZU:Z \\ SUBKAT:X_1X_2 \end{bmatrix} \quad \begin{bmatrix} FIN:F \\ ZU:Z \\ SUBKAT:X_1(ip:inf)X_2 \end{bmatrix} \quad \begin{bmatrix} FIN:- \\ ZU:+ \\ SUBKAT:np:nom \end{bmatrix}$$

$$VP \rightarrow NP \quad V'$$
$$\begin{bmatrix} FIN:F \\ ZU:Z \\ SUBKAT:X_1X_2 \end{bmatrix} \quad \begin{bmatrix} PERS:3 \\ KAS:K \end{bmatrix} \quad \begin{bmatrix} FIN:F \\ ZU:Z \\ SUBKAT:X_1(np:K)X_2 \end{bmatrix}$$

$$VP \rightarrow PP \quad V'$$
$$\begin{bmatrix} FIN:F \\ ZU:Z \\ SUBKAT:X_1X_2 \end{bmatrix} \quad \begin{bmatrix} KAS:K \\ PREP:P \end{bmatrix} \quad \begin{bmatrix} FIN:F \\ ZU:Z \\ SUBKAT:X_1(pp:P\_K)X_2 \end{bmatrix}$$

$$VP \rightarrow V$$
$$\begin{bmatrix} FIN:F \\ ZU:Z \\ SUBKAT:S \end{bmatrix} \quad \begin{bmatrix} FIN:F \\ ZU:Z \\ SUBKAT:S \end{bmatrix}$$

$$VP \rightarrow NP \quad V$$
$$\begin{bmatrix} FIN:F \\ ZU:Z \\ SUBKAT:X_1X_2 \end{bmatrix} \quad \begin{bmatrix} KAS:K \end{bmatrix} \quad \begin{bmatrix} FIN:F \\ ZU:Z \\ SUBKAT:X_1(np:K)X_2 \end{bmatrix}$$

$$VP \rightarrow PP \quad V$$
$$\begin{bmatrix} FIN:F \\ ZU:Z \\ SUBKAT:X_1X_2 \end{bmatrix} \quad \begin{bmatrix} KAS:K \\ PREP:P \end{bmatrix} \quad \begin{bmatrix} FIN:F \\ ZU:Z \\ SUBKAT:X_1(pp:P\_K)X_2 \end{bmatrix}$$

$$V' \rightarrow NP \quad V$$
$$\begin{bmatrix} FIN:F \\ ZU:Z \\ SUBKAT:X_1X_2 \end{bmatrix} \quad \begin{bmatrix} KAS:K \end{bmatrix} \quad \begin{bmatrix} FIN:F \\ ZU:Z \\ SUBKAT:X_1(np:K)X_2 \end{bmatrix}$$

$$V' \rightarrow PP \quad V$$
$$\begin{bmatrix} FIN:F \\ ZU:Z \\ SUBKAT:X_1X_2 \end{bmatrix} \quad \begin{bmatrix} KAS:K \\ PREP:P \end{bmatrix} \quad \begin{bmatrix} FIN:F \\ ZU:Z \\ SUBKAT:X_1(pp:P\_K)X_2 \end{bmatrix}$$

Wird der in Abbildung 4.11 dargestellte übergenerierte Satz um Merkmalsstrukturen und Restriktionen ergänzt, scheitert der Aufbau. Abbildung 4.13 zeigt den Syntaxbaum mit den relevanten Merkmalsstrukturen. Es ist nicht möglich die Restriktionen der Regel I' → I VP, die nochmals in Abbildung 4.12 dargestellt sind, anzuwenden, da eine VP mit einem Subkategorisierungsrahmen <np:nom> verlangt wird aber die VP des Satzes den Subkategorisierunsgrahmen <np:nom, ip:inf> hat.

$$\underset{\begin{bmatrix} NUM:N \\ PERS:P \\ FIN:F \end{bmatrix}}{I'} \rightarrow \underset{\begin{bmatrix} NUM:N \\ PERS:P \\ FIN:F \end{bmatrix}}{I} \quad \underset{\begin{bmatrix} FIN:- \\ ZU:- \\ SUBKAT:np:nom \end{bmatrix}}{VP}$$

Abbildung 4.12.: Regel I' → I VP mit Merkmalsstrukturen und Restriktionen

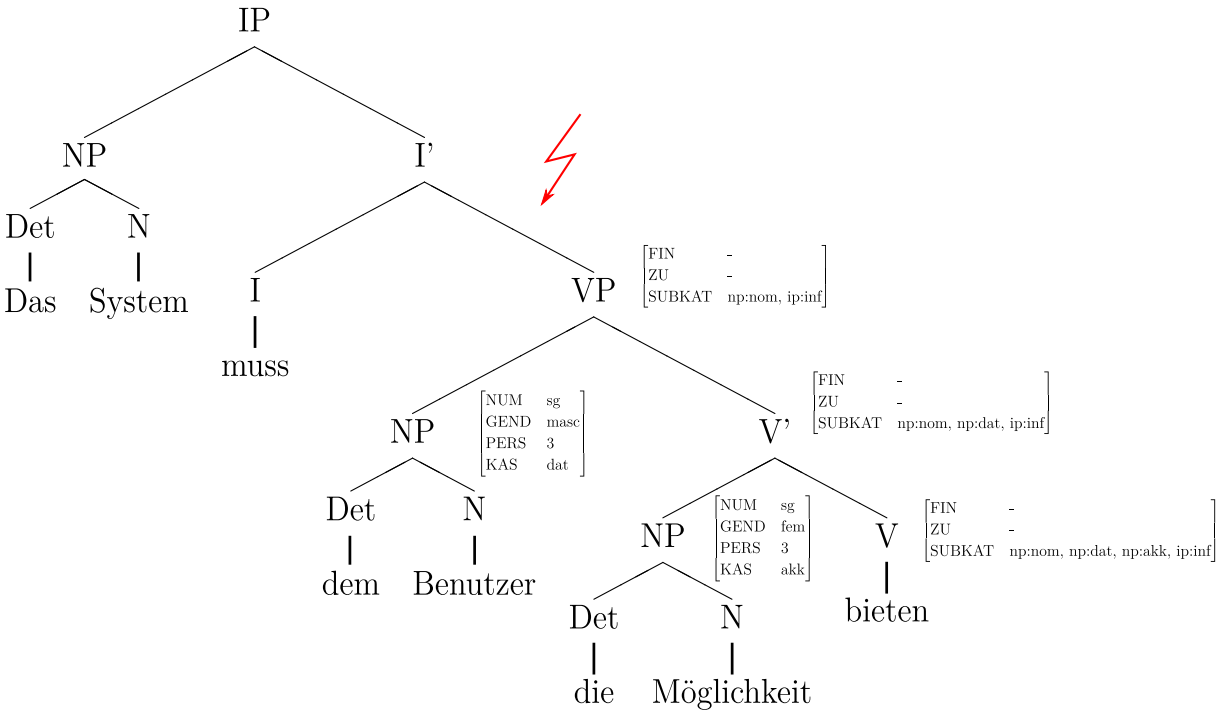

Abbildung 4.13.: Übergenerierter Satz aus Abbildung 4.11 mit Merkmalsstrukturen

Zusätzlich zu den Restriktionen bezüglich der Merkmalsstrukturen werden drei weitere Bedingungen zur Überprüfung des Anforderungstyps aufgestellt. Es wird verlangt, dass der Strukturbaum des Satzes entweder eine der beiden Strukturen aus Abbildung 4.14 enthält, wobei wirklich nur eine der beiden Strukturen im Baum wieder gefunden werden darf, oder die Struktur aus Abbildung 4.15 nicht vorhanden ist. Der linke Strukturbaum in Abbildung 4.14 findet sich in einem Strukturbaum, der eine Benutzerinteraktion repräsentiert wieder. Der rechte Strukturbaum dagegen entspricht einer Schnittstellenanforderung.

Kommen diese beiden Strukturen nicht im Satz vor, darf auch die Struktur aus Abbildung 4.15 nicht vorkommen, sonst wäre der Satz auch keine selbstständige Systemaktivität.

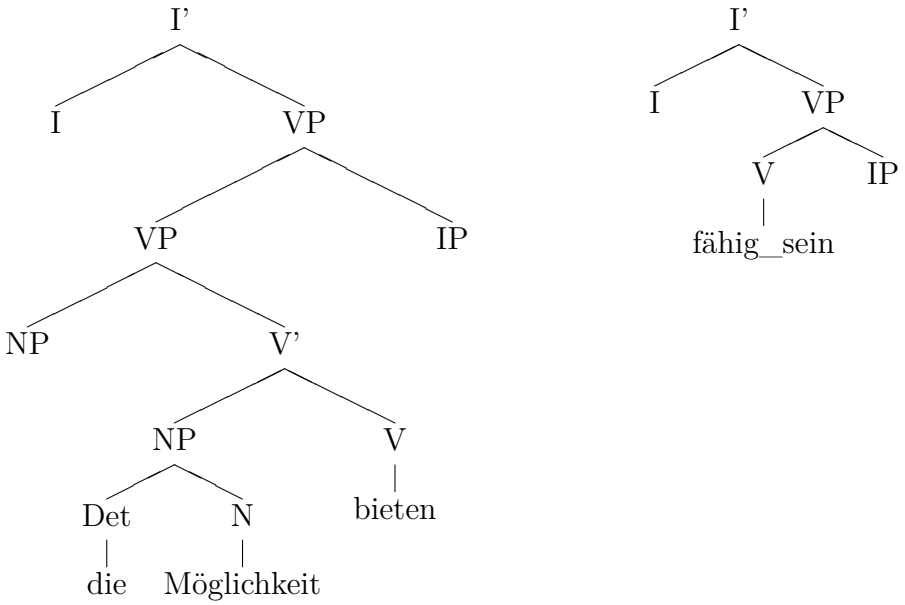

Abbildung 4.14.: Restriktionen bezüglich der Grammatik aufgrund der Schablone

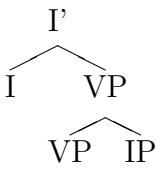

Abbildung 4.15.: Restriktionen bezüglich der Grammatik aufgrund der Schablone

Schon durch die Basisgrammatik wird weitestgehend gewährleistet, dass die Struktur des Satzes der Schablone entspricht. Allerdings besteht die Basisgrammatik nur aus Phrasenstrukturregeln. Durch die Restriktionen wird überprüft, ob auch die lexikalischen Kategorien korrekt sind.

4.2.2. Lexikon (lexikalische Regeln)

Das Lexikon, aus dem die lexikalischen Regeln extrahiert werden, besteht für die in Abschnitt 4.2 vorgestellte Basisgrammatik aus Nomen, Verben, Modalverben, Präpositionen und Artikeln. Ein Auszug aus dem Lexikon ist als Anhang A beigefügt. Die enthaltenen Wörter stammen aus einem großen monolingualen Korpus, extrahiert aus verschiedenen Spezifikationsdokumenten. Dadurch ist eine domänenspezifische Sprache gewährleistet.

Die Wörter werden aus diesem Korpus extrahiert und um Informationen, die für die Merkmalsstrukturen relevant sind, ergänzt.
Wie in Abschnitt 4.2.1 angesprochen, können mit der Basisgrammatik unerwünschte Sätze generiert werden. Deswegen werden Restriktionen bezüglich der Merkmalsstrukturen für die Phrasenstrukturregeln benötigt.
Die Merkmalsstruktur für Nomen und Artikel hat vier Argumente. Angegeben werden muss, wie in Abbildung 4.16 dargestellt, Numerus, Genus, Person und Kasus.

$$\begin{bmatrix} \text{NUM} & \text{sg, pl} \\ \text{GEND} & \text{masc, fem, neut} \\ \text{PERS} & \text{1, 2, 3} \\ \text{KAS} & \text{nom, akk, dat, gen} \end{bmatrix}$$

Abbildung 4.16.: Merkmalsstruktur für Nomen und Artikel

Die Merkmalsstruktur für Verben (siehe Abbildung 4.17) verlangt die Angabe der Finitheit und des Subkategorisierungsrahmens. Zusätzlich wird angegeben, ob das Verb ein zu-Infinitiv ist.

$$\begin{bmatrix} \text{FIN} & + \,/\, - \\ \text{ZU} & + \,/\, - \\ \text{SUBKAT} & * \end{bmatrix}$$

Abbildung 4.17.: Merkmalsstruktur für Verben

Für die Modalverben muss in der Merkmalsstruktur, welche in Abbildung 4.18 dargestellt ist, wie bei den Nomen und Artikeln der Numerus und die Person angegeben werden, zusätzlich wie bei den Verben die Finitheit.

$$\begin{bmatrix} \text{NUM} & \text{sg, pl} \\ \text{PERS} & \text{1, 2, 3} \\ \text{FIN} & + \,/\, - \end{bmatrix}$$

Abbildung 4.18.: Merkmalsstruktur für Modalverben

Die Präpositionen sind durch Angabe des Kasus vollständig beschrieben. Bei den Restriktionen für die Basisgrammatik 4.3 tauchen bei den Präpositionen zusätzlich das Merkmal

PREP auf. Dieses wird aus übersetzungstechnischen Gründen benötigt und entspricht der Präposition (z.B. an, auf, ...).

$$\begin{bmatrix} \text{KAS} & \text{nom, akk, dat, gen} \\ \text{PREP} & \text{auf, an, ...} \end{bmatrix}$$

Abbildung 4.19.: Merkmalsstruktur für Präpositionen

Mit diesen Merkmalen sind die Wörter, für die im Rahmen dieser Arbeit zu übersetzenden Sätze, ausreichend beschrieben. Die Werte der Merkmale eines Wortes können direkt aus dem entsprechenden Lexikoneintrag entnommen und eine lexikalische Regel erzeugt werden. Die Abbildung 4.20 zeigt eine mögliche lexikalische Regel für das Wort „System“.

$$\begin{matrix} N \\ \begin{bmatrix} NUM : sg \\ GEND : fem \\ PERS : 3 \\ KAS : nom \end{bmatrix} \end{matrix} \quad \rightarrow \quad \begin{matrix} System \\ \begin{bmatrix} NUM : sg \\ GEND : fem \\ PERS : 3 \\ KAS : nom \end{bmatrix} \end{matrix}$$

Abbildung 4.20.: Lexikalische Regel für das Wort „System“

Die Daten für das Lexikon stammen hauptsächlich aus SMOR, einem Werkzeug, das am IMS[2] entwickelt wurde und die Morphologie deutscher Wörter analysiert [SFH02]. Es können damit unter anderem Grundform, Kasus, Genus, Numerus und Person eines Wortes ermittelt werden. Außerdem funktioniert diese Analyse auch in die umgekehrte Richtung, d.h. wenn Grundform, Kasus, Numerus usw. bekannt sind, kann die entsprechende Wortform ausgegeben werden. Dadurch war es zum Beispiel möglich die Form eines Verbs in der dritten Person im Plural automatisch zu ermitteln.
Bei dem Lexikon handelt es sich um ein Vollformenlexikon, d. h. alle möglichen und notwendigen Formen eines Wortes können direkt nachgeschlagen werden. In anderen Lexika ist zum Beispiel bei den Verben nur der Infinitiv gespeichert und andere Formen müssen über Regeln aus diesem generiert werden.
Die Subkategorisierungsrahmen der Verben stammen aus der Arbeit von Schulte im Walde [Sch03b]. Diese und alle bisher genannten Daten standen für diese Diplomarbeit schon aufbereitet zur Verfügung. Im Rahmen dieser Arbeit wurden die Daten in ein einheitliches Format zusammengefasst und wenn möglich mit Übersetzungen aus den Daten von Hoedoro ergänzt.
In dieser Arbeit werden in den Subkategorisierungsrahmen nur Nominalphrasen, Präpositionalphrasen und Infinitvphrasen berücksichtigt. Eine NP wird im deutschen Subategorisierungsrahmen beschrieben durch „np:KAS“, wobei KAS das Kasus-Merkmal der

[2]Institut für maschinelle Sprachverarbeitung der Universität Stuttgart

Nominalphrase ist. Eine PP wird angegeben durch „pp:PREP_KAS“. PREP und KAS sind die Merkmale aus der Merkmalsstruktur. Eine Infintivphrase wird ohne zusätzlich Merkmale mit „ip:inf“ beschrieben.
Die Subkategorisierungsrahmen für die englischen Verben hingegen stammen aus einer Liste mit englischen Verben und deren mögliche Subkategorisierungsrahmen. Diese Liste wurde aus dem englischen Wikipedia mit dem BitPar[3] Parser im Rahmen des WordGraph[4] Projekts am IMS erstellt. Die Subkategorisierungsrahmen wurden statistisch extrahiert und sind daher nicht vollständig korrekt. Für einen repräsentativeren Datensatz werden nur die am häufigsten vorkommenden Subkategorisierungsrahmen für jedes Verb verwendet. Die Notation der englischen Subkategorisierungsrahmen wurde fast vollständig übernommen und nur teilweise an die Notation der deutschen Subkategorisierungsrahmen angepasst. Beispielsweise ein Subjekt wird in englischen Subkategorisierungsrahmen mit „subj“ und in deutschen mit „np:nom“ beschrieben. Eine Nominalphrase wird nur durch „np“angegeben und die Präpositionalphrase durch „pp:PREP“.

4.3. Transferbasierte Übersetzung

Ergebnis der Diskussion zur Auswahl eines Verfahrens für die maschinelle Übersetzung in Abschnitt 4.1 ist das transferbasierte Verfahren. Eine transferbasierte Übersetzung, wie sie in Kapitel 2.2.2 beschrieben wird, beinhaltet die Analyse des zu übersetzenden Satzes, den Transfer von einer quellsprachlichen in eine zielsprachliche Repräsentation und die Generierung des Satzes in der Zielsprache. In den folgenden Abschnitten soll auf das Konzept der einzelnen Schritte näher eingegangen werden.

4.3.1. Analyse

Die transferbasierte Übersetzung beginnt mit der Analyse des zu übersetzenden Satzes. Diese Analyse kann aufgeteilt werden, in die morphologische und die strukturelle Analyse. Auf eine morphologische Analyse kann im Moment verzichtet werden, da ein Vollformenlexikon zur Verfügung steht. Alle Wörter des Satzes werden direkt darin nachgeschlagen und müssen nicht erst auf ihre Grundform reduziert werden. Eine morphologische Analyse macht Sinn, um zum Beispiel die Größe des Lexikons zu reduzieren [HS92, S. 82]. Dies kann als Erweiterung für das Werkzeug in Frage kommen.
Die strukturelle Analyse übernimmt ein Earley Parser. Das Vorgehen des Earley Parsers wurde in Kapitel 2.4.2 beschrieben. Als Eingabe erhält dieser den zu übersetzenden Satz

[3] vgl. [Sch04]
[4] http://wiki.ims.uni-stuttgart.de/extern/WordGraph

und die deutsche Basisgrammatik aus Abschnitt 4.2.1. Die Basisgrammatik wird um lexikalische Regeln, durch das Nachschlagen der Wörter im Lexikon, erweitert. Dann kann der Satz analysiert werden.
Um zu vermeiden, dass nicht gewollte Sätze akzeptiert werden, müssen die Restriktionen bezüglich der Merkmale eingehalten werden. Außerdem wird durch zusätzliche Restriktionen (siehe S.49) überprüft, ob der Satz gemäß der Schablone geschrieben wurde.

In Kapitel 2.2.3 werden einige Probleme der maschinellen Übersetzung erörtert. Ein Teil dieser Probleme betrifft die Analyse, genauer sind dies die Probleme, die nur eine Sprache betreffen. Darunter fallen strukturelle und kategoriale Ambiguitäten auf Syntaxebene. Diese treten bei den Anforderungen, die von der Grammatik akzeptiert werden nicht auf, da die Grammatik eindeutig ist. Eine eindeutige Grammatik ist sinnvoll, da eindeutige Anforderungen verlangt werden. Ambiguitäten auf Morphologieebene treten auch nicht auf, denn in der Subsprache sind alle Wörter eindeutig definiert. Gleiches gilt für Ambiguitäten auf der Semantikebene, wie zum Beispiel Homographe oder Polyseme. Auch diese Wörter sind in der domänenspezifischen Sprache eindeutig. Die referentielle Ambiguität hingegen kann, wie in Kapitel 2.2.3 bereits erwähnt, bei einer satzweisen Übersetzung nicht aufgelöst werden.

4.3.2. Transfer

Der zweite Schritt der transferbasierten Übersetzung ist der Transfer der Repräsentation des Satzes in Quellsprache in eine Repräsentation des Satzes in Zielsprache. Nach Hutchins und Somers in [HS92][5] kann dieser Schritt in den lexikalischen und den syntaktischen Transfer aufgeteilt werden.
Der lexikalische Transfer überführt die Wörter des Satzes mittels eines bilingualen Lexikons von der Quellsprache in die Zielsprache. Die Einträge der Wörter, die im Rahmen der Analyse im Lexikon nachgeschlagen werden, enthalten auch immer eine Übersetzung, die hier verwendet wird.
Der strukturelle Transfer wird durch Transferregeln beschrieben. Er überführt die Strukturbeschreibung der Analyse, die abhängig ist von der Quellsprache, mittels der Transferregeln in eine Strukturbeschreibung in der Zielsprache. Der strukturelle Transfer lässt sich in mehrere Schritte aufteilen.
Zuerst werden schablonenspezifische Transferregeln angewandt. Im Moment beinhalten diese nur die in Abbildung 4.21 dargestellt Regel. Durch die schablonenspezifischen Transferregeln wird gewährleistet, dass die Struktur der deutschen Schablone in die Struktur der englischen Schablone überführt wird.

[5] vgl. [HS92] S. 113

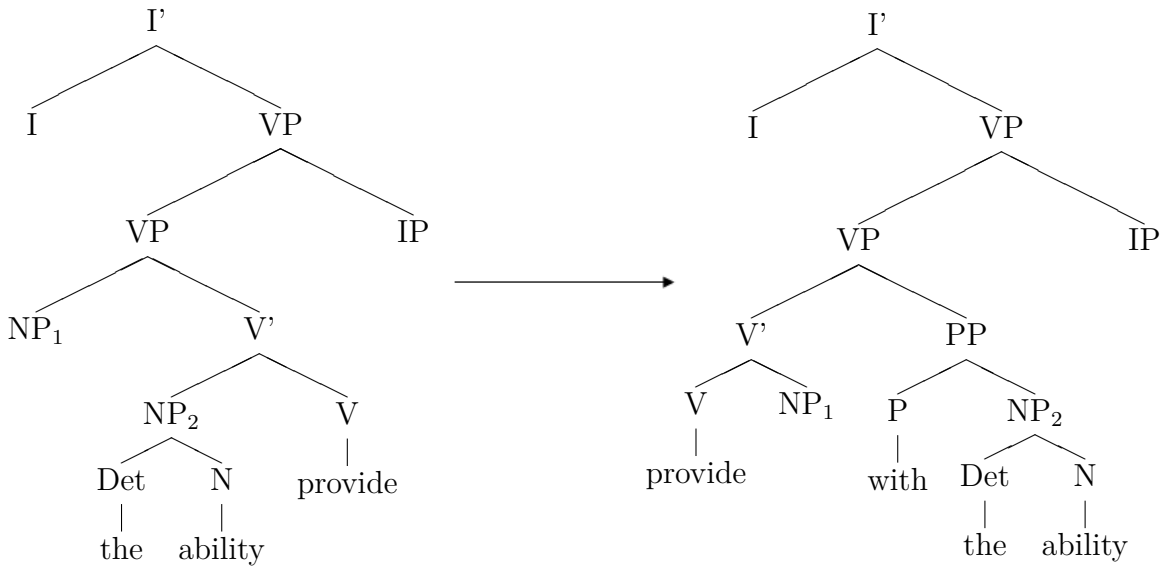

Abbildung 4.21.: Schablonenspezifische Transferregel

Die Transferregel in Abbildung 4.21 betrifft nur den Anforderungstyp „Benutzerinteraktion“. Die anderen beiden Anforderungstypen benötigen keinen speziellen Transfer und können mit den folgenden sprachspezifischen Transferregeln korrekt übersetzt werden. Auch der dargestellte Transfer für die Benutzerinteraktion könnte mit den folgenden Regeln transferiert werden. Es würden dadurch allerdings mehrere Übersetzungsmöglichkeiten (siehe S. 56) entstehen, wobei nur eine davon richtig ist, da die Übersetzung wegen der Schablone an dieser Stelle eindeutig ist. Mit den schablonenspezifischen Transferregeln wird diese eindeutige Übersetzung umgesetzt.
Vor dem Transfer mittels sprachspezifischen Regeln werden zunächst noch die Subkategorisierungsrahmen der Verben transferiert. Dieser Schritt soll anhand des Beispielsatzes „Das System muss dem Benutzer eine Antwort geben.“, dessen Phrasenstrukturbaum in Abbildung 4.22 dargestellt ist, erläutert werden.
Das Prozesswort „geben“ hat in diesem Fall den Subkategorisierungsrahmen <np:nom, np:dat, np:akk>. Als Übersetzung für „geben“ steht im Lexikon „give“ mit folgenden möglichen Subkategorisierungsrahmen: <subj, np>, <subj, np, np> und <subj, np, pp:to>. Es wäre natürlich ein großer Vorteil für die Übersetzung, wenn eine eindeutige Zuordnung zwischen den deutschen Subkategorisierungsrahmen und den englischen Subkategorisierungsrahmen vorliegen würde. Da für eine automatische Zuordnung keine Werkzeuge und Daten zur Verfügung stehen und die Entwicklung solcher den Rahmen dieser Arbeit sprengen würden, wurde auf eine eindeutige Zuordnung verzichtet. Eine manuelle Zuordnung ist aus Kosten- und Zeitgründen zu aufwändig.
Um „geben“ und die Elemente aus dessen Subkategorisierungsrahmen korrekt zu übersetzen, wird der dazugehörige englische Subkategorisierungsrahmen ermittelt. Dafür kommen nur jene in Frage, die gleich viele Elemente haben, wie der deutsche Subkategorisierungsrahmen. Subkategorisierungsrahmen mit mehr oder weniger Elemente können nicht verwendet werden, da das Werkzeug nicht fähig ist zu entscheiden, ob und wenn ja welche

Elemente weggelassen werden sollen und dem Werkzeug das Wissen fehlt, um ein neues Element hinzuzufügen. In dem Beispiel kommt nur der englische Subkategorisierungsrahmen <subj, np, pp:to> für den Transfer in Frage. Es ist generell möglich, dass auch mehrere Subkategorisierungsrahmen passen.

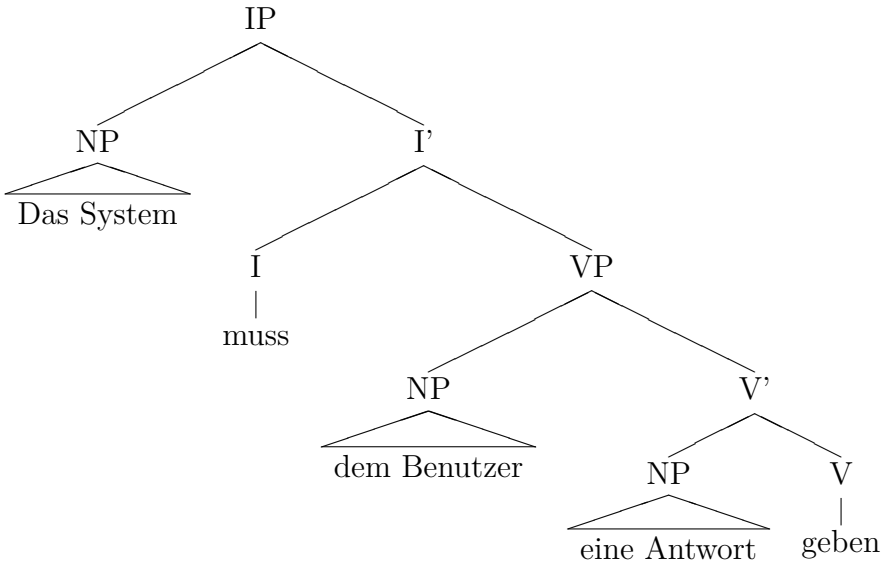

Abbildung 4.22.: Beispielsatz für einen Transfer der Subkategorisierungsrahmen

Für jeden möglichen Subkategorsierungsrahmen wird geprüft, ob der englische ebenso wie der deutsche ein Subjekt (*subj* bzw. *np:nom*) enthält. Da die Position des Subjekts im Satz feststeht und in der deutschen, so wie der englischen Schablone gleich ist, können diese Elemente der Subkategorisierungsrahmen als bearbeitet angesehen werden. Sie werden nicht weiter berücksichtigt. Es bleibt noch der Transfer zwischen <np:dat, np:akk> und <np, pp:to>.
Aufgrund der Kenntnisse über die Daten aus dem Lexikon wird davon ausgegangen, dass die Reihenfolge der Elemente im englischen Subkategorisierungsrahmen auch der Reihenfolge der Elemente im Satz entspricht. Deswegen werden den englischen Elementen die deutschen Elemente zugeordnet und nicht umgekehrt. Die Reihenfolge der englischen Elemente und die Reihenfolge der deutschen Elemente muss allerdings nicht übereinstimmen, d.h. alle möglichen Zuordnungen bestimmen die verschiedenen Übersetzungsmöglichkeiten. Bezogen auf die Subkategorisierungsrahmen aus dem Beispiel bedeutet dies, dass es sowohl Sätze gibt in denen *np:dat* mit *np* und *np:akk* mit *pp:to* als auch Sätze existieren in denen *np:dat* mit *pp:to* und *np:akk* mit *np* übersetzt wird (vgl. Abbildung 4.23). Dem Benutzer werden deswegen beide bzw. generell alle Übersetzungsmöglichkeiten angeboten. Das Übersetzungssystem ist nicht in der Lage zu entscheiden, welche der Möglichkeiten korrekt ist, da dazu notwendiges Wissen fehlt.

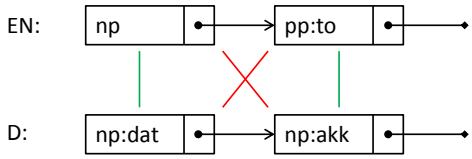

Abbildung 4.23.: Alle Möglichkeiten für die Zuordnung zwischen den Subkategorisierungsrahmen des Beispielsatzes

Bei dem Transfer der Merkmale der Subkategorisierungsrahmen wird von einem Phrasentyp in einen anderen transferiert. In der Basisgrammatik ist nur ein Transfer von NP in NP, PP in PP, NP in PP und PP in NP möglich. Der Transfer einer NP in eine NP erfordert keine strukturelle Änderung, die NP wird direkt übernommen. Bei einem Transfer von einer PP in eine PP ändert sich die Präposition. Diese wird bei einer PP im Subkategorisierungsrahmen immer mit angegeben. Strukturelle Änderungen tauchen bei einem Transfer von einer NP in eine PP oder bei einem Transfer in die umgekehrte Richtung auf. In Abbildung 4.24 ist die Struktur einer PP dargestellt. Bei einem Transfer einer NP in eine PP wird die zu transferierende NP an die Stelle der NP in der PP gehängt und die PP um die Präposition P aus dem Subkategorisierungsrahmen ergänzt. In der umgekehrten Richtung, also beim Transfer einer PP in eine NP, wird die NP der PP als NP genommen.

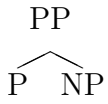

Abbildung 4.24.: Struktur einer Präpositionalphrase (PP) der Basisgrammatik

Mit dem Transfer des Subkategorisierungsrahmen ist die Strukturbeschreibung noch nicht vollständig transferiert. Es folgen als Letztes die schon erwähnten sprachspezifischen Transferregeln, dargestellt in Abbildung 4.25, 4.26 und 4.27. Diese Regeln sind abhängig von der Quell- und der Zielsprache. Bei einem Transfer von deutsch nach englisch wird die Verbstellung im Satz verändert. In der deutschen Basisgrammatik wird von einer Verbletztstellung, in der englischen Basisgrammatik von einer Verberststellung gesprochen. Durch die sprachspezifischen Transferregeln wird diese strukturelle Veränderung durchgeführt.

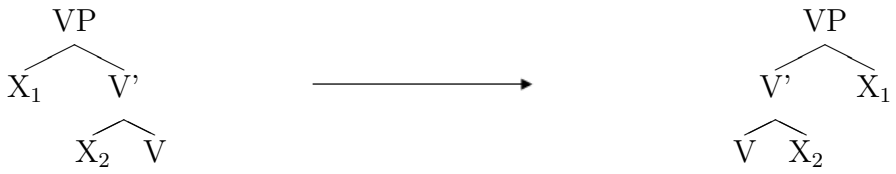

Abbildung 4.25.: Sprachspezifische Transferregel 1

Abbildung 4.26.: Sprachspezifische Transferregel 2

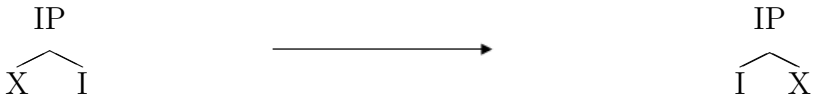

Abbildung 4.27.: Sprachspezifische Transferregel 3

Der Beispielsatz aus Abbildung 4.22 ist nach der deutschen Schablone eine selbstständige Systemaktivität, d.h. die schablonenspezifische Regel wird nicht angewandt. Nachdem alle anderen Transferregeln angewandt und damit der lexikalische, so wie auch der strukturelle Transfer beendet ist, sind die in Abbildung 4.28 und 4.29 dargestellten Phrasenstrukturbäume das Ergebnis der Übersetzung. Diese repräsentieren alle vorgeschlagenen Übersetzungsmöglichkeiten des Beispielsatzes.

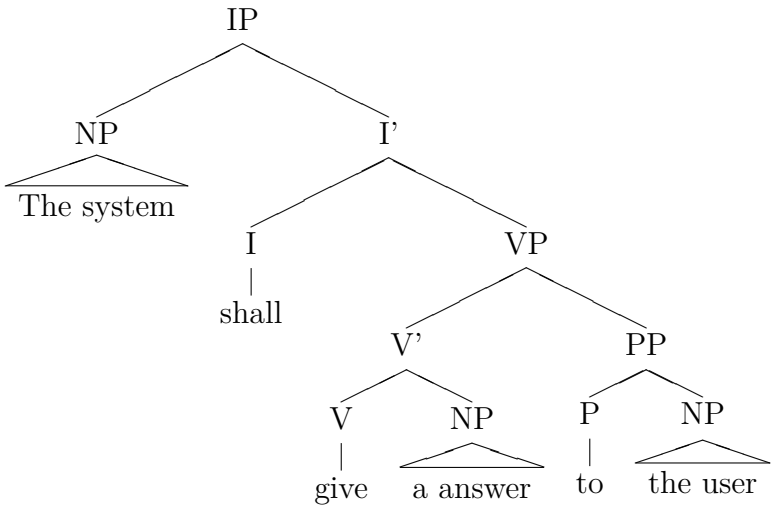

Abbildung 4.28.: Erste Übersetzungsmöglichkeit des Beispielsatzes

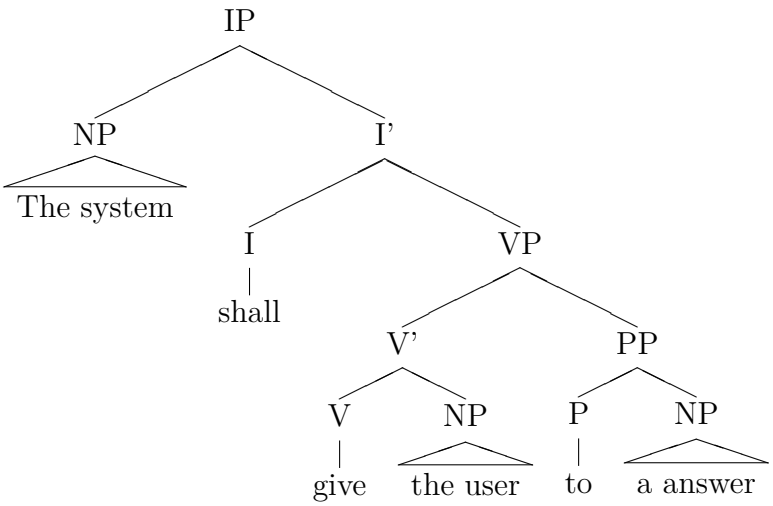

Abbildung 4.29.: Zweite Übersetzungsmöglichkeit des Beispielsatzes

Die Reihenfolge der Transferschritte muss eingehalten werden, da sonst die Übersetzung nicht korrekt ist. Werden beispielsweise die sprachspezifischen vor den schablonenbasierten Transferregeln angewandt, kann die linke Seite der Regel aus Abbildung 4.21 im Strukturbaum einer Benutzerinteraktion nicht wieder gefunden werden. Der schablonenbasierte Transfer würde also nicht ausgeführt werden und der korrekte Transfer zwischen der deutschen und der englischen Schablone wäre nicht gewährleistet.

Wie die Analyse betreffen die Probleme der maschinellen Übersetzung aus Kapitel 2.2.3, auch den Transferschritt. Dies sind Probleme, die bei der Übersetzung zwischen zwei Sprachen auftreten. Dazu gehören die lexikalischen Lücken (lexical gaps), die fehlenden Entsprechungen (mismatches), die unterschiedliche Granularität, die unterschiedliche syntaktische Struktur und das Head Switching. Das erste Problem, die lexikalischen Lücken, wird ganz konsequent behandelt. Findet sich eine Lücke im Lexikon, gibt es keine Übersetzung. Es kann als Erweiterung eine Benutzerinteraktion eingebaut werden, welche dem Benutzer die Möglichkeit bietet, eine eigene Übersetzung einzugeben. Gleich behandelt werden fehlende Entsprechungen. Lexikalische Lücken und fehlende Entsprechungen kommen, wenn überhaupt, nur als Ausnahmefall vor, da eine domänenspezifische Sprache verwendet wird. Gleiches gilt für eine unterschiedliche Granularität der Wörter. In einer domänenspezifischen Sprache ist die Übersetzung meist eindeutig. Falls dieses Problem denoch auftritt kann dem Benutzer die Wahl gelassen werden, welche der möglichen Übersetzungen er wählt. Das Problem der unterschiedlichen syntaktischen Strukturen oder des Head Switchings wird, wenn es nicht schon durch die Verwendung der Schablonen ausgeschlossen ist, über die Transferregeln oder den Subkategorisierungsrahmen bewältigt.

4.3.3. Generierung

Die Generierung ist der letzte Schritt in der transferbasierten Übersetzung. Diese kann ebenso, wie die beiden anderen Schritte, in zwei Teile, die syntaktische und die morphologische Generierung, aufgeteilt werden [HS92, S. 133].
Zuerst wird die syntaktische Generierung durchgeführt. Alle Strukturbeschreibungen, die Ergebnis des Transferschritts sind, werden Inorder durchlaufen, um den englischen Satz zu extrahieren. Inorder bedeutet, dass für jeden Knoten des binären Strukturbaums zuerst der linke Teilbaum des Knotens betrachtet wird, anschließend der Knoten selbst und zuletzt der rechte Teilbaum. Der Satz besteht aus den Einträgen der Blätter des Baums in der Reihenfolge in der sie angetroffen werden, wenn der Baum inorder durchlaufen wird.
Eine morphologische Generierung ist anschließend nicht notwendig, da die Wörter im Analyseschritt nicht auf ihre Grundform reduziert werden und daher die richtige Form im Transferschritt direkt übersetzt wird.

5. Implementierung

Im vorliegenden Kapitel 5 wird die Implementierung des Werkzeugs und die praktische Umsetzung des Konzepts aus Kapitel 4 betrachtet. Der Aufbau dieses Kapitels ähnelt deswegen dem des Kapitel 4. Nach Einführung der grundlegenden Systeminformationen in Abschnitt 5.1 werden die Analyse, der Transfer und die Generierung einer transferbasierten Übersetzung in den Abschnitten 5.2, 5.3 und 5.4 nochmals beleuchtet. Der Abschnitt 5.5 behandelt die Benutzeroberfläche des Werkzeugs.

5.1. Systeminformationen

Das Werkzeug wurde in der Programmiersprache Java mithilfe der Entwicklungsumgebung Eclipse[1] implementiert. Für die Wahl der Programmiersprache gab es mehrere Gründe. Der wichtigste Grund ist, dass Java in der Softwareentwicklung allgemein, aber besonders auch bei der Daimler AG sehr verbreitet ist. Deshalb ist unter anderem eine einfache Anbindung an andere Werkzeuge, wie z. B. das Werkzeug von Hoedoro [Hoe11] in das die Übersetzungsfunktion integriert und welches bereits in Java implementiert wurde, möglich. Aus denselben Gründen wird auch die SWT(Standard Widget Toolkit)-Bibliothek[2] für die Erweiterung der Benutzeroberfläche verwendet.
Um das Werkzeug möglichst erweiterbar zu gestalten, wird unter anderem die Grammatik in eine XML[3]-Datei ausgelagert (siehe Abschnitt 5.2.1). XML ist eine Sprache, die relativ leicht erlernbar ist und auch ohne tiefere Programmierkenntnisse verwendet werden kann. So muss beispielsweise zur Erweiterung der Grammatik die Implementierung des Werkzeugs nicht geändert werden.
Um eine XML-Datei mit Java zu verarbeiten wird JDOM[4] verwendet, eine API mit der XML-Dokumente eingelesen, verarbeitet und erzeugt werden können.
Damit das Debuggen des Werkzeugs bezüglich einer möglichen Weiterentwicklung erleichtert wird, gibt es toString-Funktionen für die verschiedenen Klassen. Dadurch ist es zum Beispiel möglich eine Chart oder einen Syntaxbaum als String auszugeben.

[1] http://www.eclipse.org/
[2] http://www.eclipse.org/swt/
[3] http://www.w3.org/XML/
[4] http://www.jdom.org

5.2. Analyse

Zur Vorbereitung der Analyse wird die Grammatik und das Lexikon eingelesen. Anschließend nutzt der Chart-Parser beides, um den eingegebenen Satz zu analysieren und bezüglich der Restriktionen zu prüfen.

5.2.1. Datenstruktur und implementeller Aufbau der Grammatik

Die Grammatik aus Kapitel 4.2.1 liegt, wie in Abschnitt 5.1 motiviert, in XML vor. Ein Auszug aus der XML-Datei wird in Abbildung 5.1, die dazugehörige Regel in Abbildung 5.2 gezeigt.

```
<rule name="IP">
	<child name="NP">
		<attribute>KAS</attribute>
	</child>
	<child name="I">
		<attribute>FIN</attribute>
		<attribute>ZU</attribute>
		<attribute>SUBKAT</attribute>
	</child>

	<attribute name="FIN">
		<assignment>
			<child name="I" pos="2" attribute="FIN"/>
		</assignment>
	</attribute>
	<attribute name="ZU">
		<assignment>
			<child name="I" pos="2" attribute="ZU"/>
		</assignment>
	</attribute>
	<attribute name="SUBKAT">
		<assignment>
			<subkat>
				<child name="I" pos="2" attribute="SUBKAT"/>
				<child name="NP" pos="1"/>
			</subkat>
		</assignment>
	</attribute>

	<restriction>
		<subkat>
			<child name="I" pos="2" attribute="SUBKAT"/>
			<child name="NP" pos="1"/>
		</subkat>
	</restriction>
</rule>
```

Abbildung 5.1.: Auszug aus der Grammatik-XML-Datei

$$\begin{array}{ccccc} IP & & \rightarrow & NP & I \\ \begin{bmatrix} FIN:F \\ ZU:Z \\ SUBKAT:X_1X_2 \end{bmatrix} & & & \begin{bmatrix} KAS:K \end{bmatrix} & \begin{bmatrix} FIN:F \\ ZU:Z \\ SUBKAT:X_1(np:K)X_2 \end{bmatrix} \end{array}$$

Abbildung 5.2.: Beschriebene Regel in Abbildung 5.1

Für die Grammatik sind verschiedene XML-Elemente vordefiniert, die das Werkzeug interpretieren kann. Eine Regel in der Grammatik steht zwischen den Tags <rule> und </rule>. Die Regel muss einen Namen haben, dieser enspricht der linken Seite der Regel (siehe Abbildung 5.3). Innerhalb der <rule>-Tags werden die Merkmale der linken Seite, die rechte Seite und die Restriktionen beschrieben.

```
<rule name="IP">
        ...
</rule>
```

Abbildung 5.3.: Auszug aus der Grammatik-XML-Datei - Rule

Die Elemente der rechten Seite der Regel bzw. im weiteren auch Kinder genannt, werden durch die <child>-Tags beschrieben. Die Reihenfolge dieser <child>-Tags im XML-Dokument entspricht der Reihenfolge der Elemente in der Regel. Die Regel in Abbildung 5.2 hat, wie auch in Abbildung 5.4 wieder erkannt werden kann, als erstes Kind eine NP und als zweites Kind ein I. Für jedes Element der rechten Seite sind dessen Merkmale durch <attribute>-Tags mit angegeben.

```
<rule name="IP">
        ....
        <child name="NP">
                <attribute>KAS</attribute>
        </child>
        <child name="I">
                <attribute>FIN</attribute>
                <attribute>ZU</attribute>
                <attribute>SUBKAT</attribute>
        </child>
        ...
</rule>
```

Abbildung 5.4.: Auszug aus der Grammatik-XML-Datei - Child

Auch die linke Seite der Regel besitzt Merkmale (siehe Abbildung 5.5). Diese sind ebenso wie die Merkmale der Kinder durch <attribute>-Tags angegeben. Erforderlich für ein Merkmal ist ein Name und bei den Merkmalen für die linke Seite auch eine Zuweisungsregel. Einem Merkmal auf der linken Seite kann direkt ein Wert über <value>-Tags oder der Wert eines Merkmals eines Elements der rechten Seite über <child>-Tags zugewiesen werden. Eine Besonderheit ist das <subkat>-Tag. Steht dieses innerhalb <assignment>-Tags und hat zwei Kinder (eines mit dem Attribut „attribute“) wird dem Merkmal der Wert des Subkategorisierungsrahmen des einen Kindes ohne den Anteil des zweiten Kindes zugewiesen. Hat zum Beispiel das Merkmal SUBKAT des Elements I aus Abbildung 5.2 den Wert <np:nom#np:akk> und das Element NP das Merkmal KAS mit dem Wert „akk“, dann wird dem Merkmal SUBKAT des Elements IP der Wert <np:nom> zugewiesen.
Abbildung 5.2 zeigt beispielsweise, dass der Wert des Merkmals ZU des Kindes I dem Merkmal ZU der IP zugewiesen wird (Erklärung siehe Kapitel 2.3.3). Da der betreffende Schritt in der Analyse bottom-up (siehe Kapitel 2.4.1) arbeitet werden die Werte der Merkmale der rechten Seite den Merkmalen der linken Seite der Regel zugewiesen. In dem Auszug aus der XML-Datei in Abbildung 5.5 kann diese Zuweisung wieder erkannt werden. Bei der Definition des Merkmals ZU wird die Zuweisungsregel durch ein <child>-Tag angegeben. Dies kann später ausgewertet werden, d.h. der Wert des Merkmals des Kindes wird ermittelt und dem Merkmal ZU der linken Seite der Regel zugewiesen.

```
<rule name="IP">
        ....
        <attribute name="FIN">
                <assignment>
                        <child name="I" pos="2" attribute="FIN"/>
                </assignment>
        </attribute>
        <attribute name="ZU">
                <assignment>
                        <child name="I" pos="2" attribute="ZU"/>
                </assignment>
        </attribute>
        <attribute name="SUBKAT">
                <assignment>
                        <subkat>
                                <child name="I" pos="2" attribute="SUBKAT"/>
                                <child name="NP" pos="1"/>
                        </subkat>
                </assignment>
        </attribute>
        ...
</rule>
```

Abbildung 5.5.: Auszug aus der Grammatik-XML-Datei - Attribute

Die Restriktionen werden innerhalb von <restriction>-Tags angegeben. Es gibt zwei verschiedene Typen von Restriktionen. Diese beiden Typen sind in Abbildung 5.6 dargestellt, gehören allerdings zu keiner spezifischen Regel. Der erste Typ wird durch <subkat>-Tags

angegeben und wird ähnlich interpretiert, wie die <subkat>-Tags bei den Zuweisungsregeln. Der Unterschied besteht darin, dass bei den Restriktionen kein Wert weitergegeben wird, sondern lediglich geprüft wird, ob das eine Kind im Subkategorisierungsrahmen des anderen Kindes vorkommt. Da in der Basisgrammatik nur Nominalphrasen, Präpositionalphrasen und Infinitivphrasen in den Subkategorisierungsrahmen vorkommen können, müssen die Elemente, die darin wieder gefunden werden sollen NP, PP oder IP heißen (Beschreibung der Subkategorisierungsrahmen siehe Seite 51 in Kapitel 4.2.2)
Die zweite Möglichkeit eine Restriktion anzugeben ist mittels <equal>-Tags. Innerhalb dieser <equal>-Tags können beliebig viele Argumente mit <argument>-Tags angegeben werden. Ein <argument>-Tag kann ein <child>-Tag oder ein <value>-Tag beinhalten. Bei <value>-Tags wird der Inhalt direkt als Wert verwendet, bei <child>-Tags wird über die Position des Kindes und ein Merkmal der Wert bestimmt. Die Angabe der Position ist obligatorisch. Es reicht nicht nur den Namen des Kindes anzugeben, da es mehrere Kinder mit dem gleichen Namen geben kann. Alle <argument>-Tags werden ausgewertet und die Ergebnisse miteinander verglichen. Sind diese ungleich ist die Restriktion nicht erfüllt.

```
<restriction>
	<subkat>
		<child name="I" pos="2" attribute="SUBKAT"/>
		<child name="NP" pos="1"/>
	</subkat>

	<equal>
		<argument>
			<child name= "IP" pos="2" attribute="SUBKAT"/>
		</argument>
		<argument>
			<value>np:nom</value>
		</argument>
	</equal>

	<equal>
		<argument>
			<child name="Det" pos="1" attribute="NUM"/>
		</argument>
		<argument>
			<child name="N" pos="2" attribute="NUM"/>
		</argument>
	</equal>
</restriction>
```

Abbildung 5.6.: Beispiele für Restriktionen

Eine XML-Datei, die nach eben beschriebenen Kriterien aufgebaut ist, kann von dem Werkzeug mit Hilfe von JDOM eingelesen und alle wichtigen Informationen daraus entnommen werden um eine Grammatik aufzubauen. Die Datenstruktur einer Grammatik innerhalb des Werkzeugs ist in Abbildung 5.7 dargestellt. Die Klassendiagramme enthalten aus Gründen der Übersichtlichkeit nur die wichtigsten Attribute und Methoden.

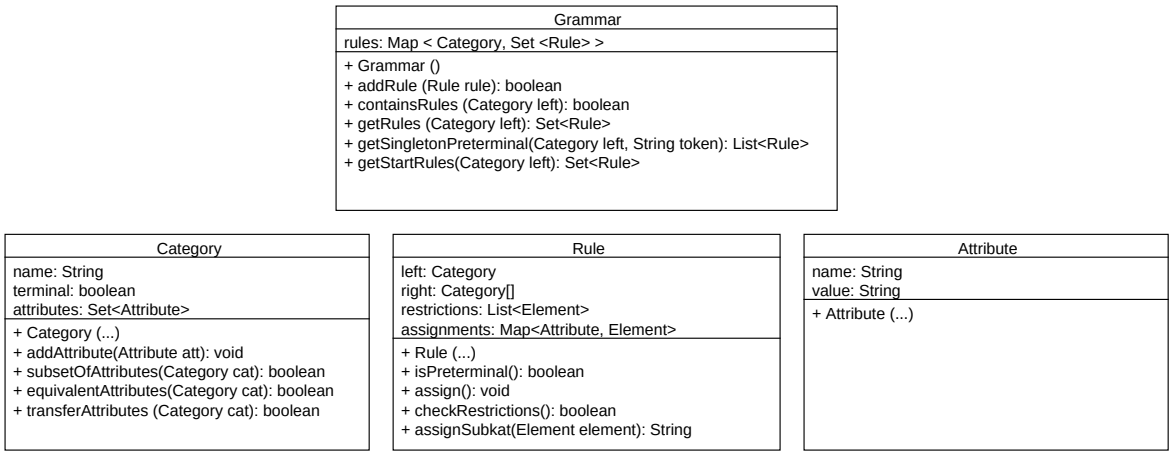

Abbildung 5.7.: Klassendiagramme der Klassen einer Grammatik

Die Grammatik besteht aus einer Klasse *Grammar*, die eine Hash-Tabelle (in Java HashMap) mit Regeln enthält. Eine Hash-Tabelle ist ideal um viele Elemente zu speichern und über einen Schlüssel, hier die Kategorie der linken Regelseite, schnell wieder zu finden. Die Klasse *Grammar* ist in Abbildung 5.8 dargestellt.

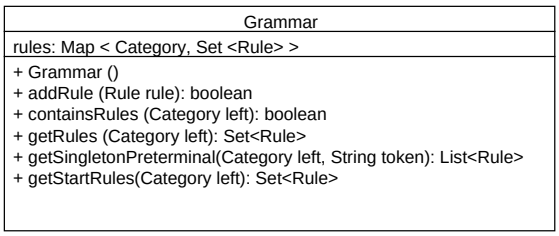

Abbildung 5.8.: Klassendiagramme der Klasse *Grammar*

Eine Kategorie wird implementiert durch die Klasse *Category*, die die Attribute *name*, *terminal* und *attributes* besitzt (siehe Abbildung 5.9). *Name* gibt den Namen der Kategorie an, *terminal* nimmt einen boolschen Wert an und ist „true“, wenn es sich bei der Kategorie um ein Terminal handelt. Das Attribut *attributes* ist ein HashSet, das die Merkmale der Kategorie speichert. Ein Set ist eine ungeordnete Menge von Elementen, wobei jedes Element nur einmal darin vorkommen darf.

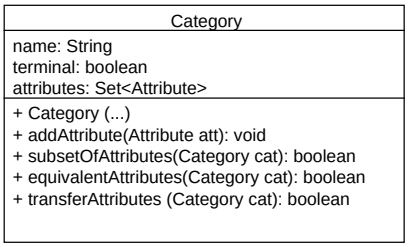

Abbildung 5.9.: Klassendiagramme der Klasse *Category*

Ein Merkmal ist gegeben durch ein Objekt der Klasse *Attribute* und besteht aus einem Namen und einem Wert. Die Klasse Attribute wird in Abbildung 5.10 gezeigt.

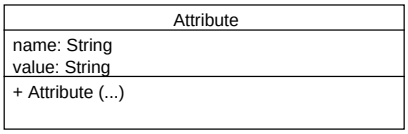

Abbildung 5.10.: Klassendiagramm der Klasse *Attribute*

Die Regeln der Grammatik werden implementiert durch die Klasse *Rule*. Diese ist in Abbildung 5.11 abgebildet und besteht aus dem Attribut *left* für die linke Kategorie der Regel, einem Array *right* mit den Kategorien der rechten Seite, einer Liste von Restriktionen und einer Hash-Tabelle in der zu jedem Merkmal aus *attributes* eine Zuweisungsregel gespeichert ist.

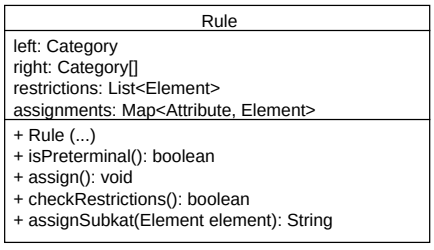

Abbildung 5.11.: Klassendiagramm der Klasse *Rule*

Mit dieser Datenstruktur stehen alle notwendigen Informationen über eine Grammatik zur Verfügung.

5.2.2. Realisierung des Lexikons

Zur Vollständigkeit der Grammatik, die aus der XML-Datei aufgebaut wurde (siehe Abschnitt 5.2.1), fehlen noch die lexikalischen Regeln. Da die Grammatik immer gleich bleibt, wird diese beim Start des Werkzeugs nur einmal eingelesen und aufgebaut, ebenso wie das Lexikon. Die lexikalischen Regeln werden für jeden eingegeben Satz aus dem Lexikon erzeugt und zur Basisgrammatik hinzugefügt. Mit der daraus entstandenen Grammatik wird dann der Satz analysiert.
Das Lexikon nur beim Start des Werkzeugs einzulesen und dann dauerhaft in einer Datenstruktur zu speichern, benötigt zwar mehr Speicherplatz, dafür ist dann die Generierung der Übersetzung schneller. Im Moment hat das Lexikon noch eine Größe bei der es möglich ist das Lexikon in einer Datenstruktur zu speichern, wird es erweitert muss ab einer gewissen Größe (je nach System auf dem das Werkzeug ausgeführt wird) eine andere Lösung gesucht werden. Wie im Kapitel 4.3 angesprochen ist eine Möglichkeit den Speicherbedarf zu reduzieren, der Verzicht auf ein Vollformenlexikon und die Einführung einer morphologischen Analyse. In dieser Arbeit ist aber die aktuelle Lösung legitim.
Das Lexikon ist in mehreren Textdateien gespeichert. Ein Eintrag im Lexikon entspricht einer Zeile in einer dieser Dateien. Jede Datei beinhaltet eine andere Wortart, jede Zeile enthält ein Wort und abhängig von der Wortart wichtige Merkmale. Diese Informationen werden aus den Dateien gelesen und in der Datenstruktur, die in Abbildung 5.12 dargestellt ist, gespeichert.
Die Klasse *DictionaryTranslation* beinhaltet für jede Wortart eine Liste. Jede Wortart wird durch eine eigene Klasse repräsentiert und enthält die Merkmale, die wichtig sind für die Merkmalsstrukturen aus Kapitel 4.2.2 und eine eindeutige Übersetzung.

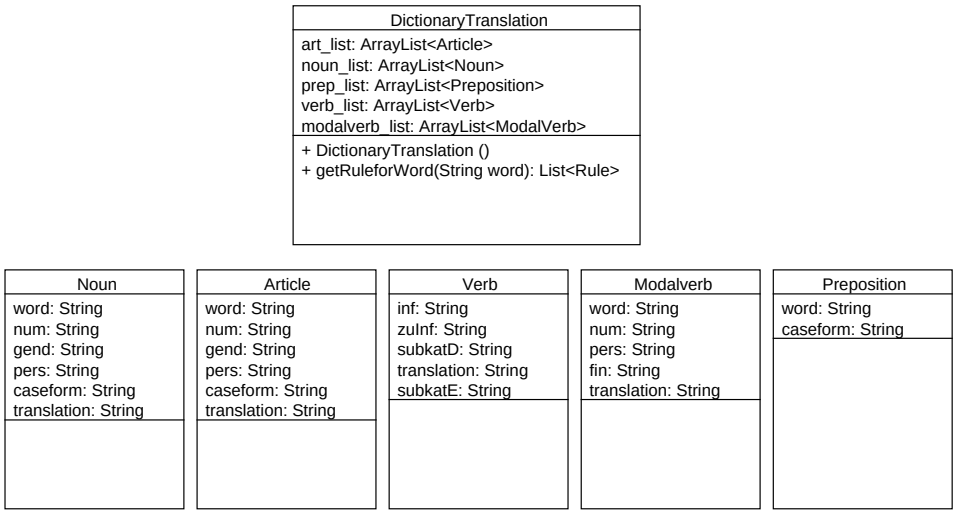

Abbildung 5.12.: Klassendiagramme des Lexikons

Wenn ein Satz übersetzt werden soll, muss, wie erwähnt, die Basisgrammatik um die notwendigen lexikalischen Regeln erweitert werden. Dazu wird der eingegebene Satz zunächst in die einzelnen Tokens unterteilt. Eine Besonderheit dabei ist die Erkennung der Wortkombination „fähig sein“ und „zu“ gefolgt von einem Verb im Infinitiv. Diese Wortkombinationen werden auch als ein Token aufgefasst. Für jedes Token wird die Methode *getRuleforWord (String word)* der Klasse *DictionaryTranslation* aufgerufen. Innerhalb dieser Methode werden die Wörterlisten nach dem Token durchsucht und für jeden gefundenen Eintrag ein Objekt der Klasse *Rule* erzeugt. Die Kategorie der linken Seite dieser Regel wird bestimmt durch die Liste aus der der Treffer stammt. Die rechte Seite der Regel ist eine terminale Kategorie, die nach dem Token benannt ist und mit den Merkmalen aus dem Eintrag der Wörterliste ergänzt wird. Wurden alle Listen durchsucht, wird eine Liste mit den erzeugten Regeln zurückgegeben und zur Basisgrammatik hinzugefügt.

5.2.3. Chart Parser

Zur Analyse des Satzes mit der Grammatik, die aus der Basisgrammatik und den lexikalischen Regeln besteht, wird ein Chart Parser, genauer ein Earley-Parser (siehe Kapitel 2.4.2), verwendet. Die Implementierung des Earley-Parsers ist angelehnt an die Implementierung von Scott [Sco07]. Der Earley-Parser von Scott verwendet allerdings keine Merkmalsstrukturen oder Restriktionen. Die Datenstruktur des Earley-Parsers ist in Abbildung 5.13 dargestellt.

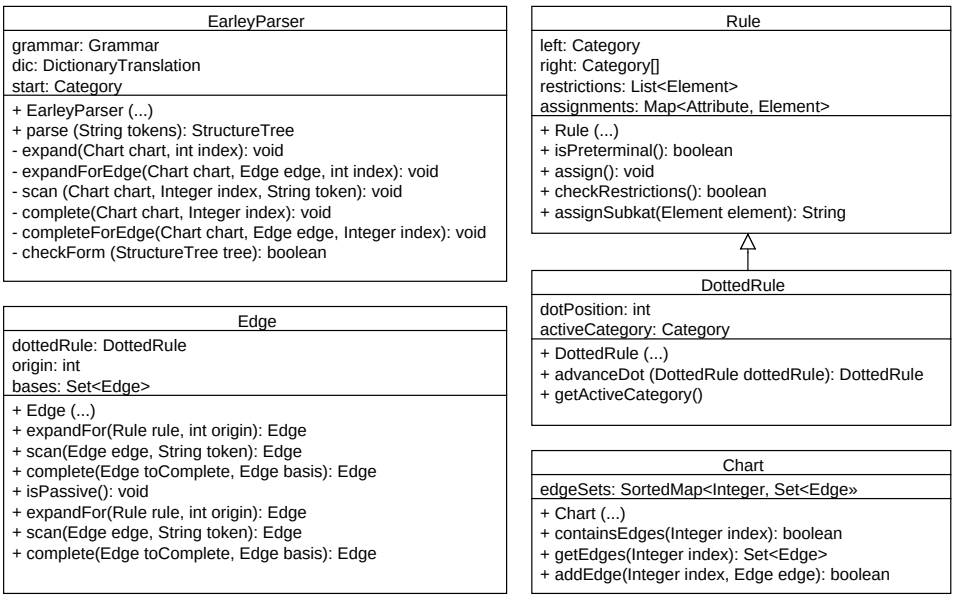

Abbildung 5.13.: Klassendiagramme des Earley-Parser

Die Klasse *EarleyParser* beinhaltet Attribute, wie das Lexikon, die Grammatik und die Startkategorie. Die Startkategorie wird in der XML-Datei der Grammatik durch das Attribut *start=„true“* gekennzeichnet. Die einzige sichtbare Methode der Klasse *EarleyParser* ist die Methode *parse(String tokens)* zum Parsen eines Satzes. Die anderen Methoden werden nur intern verwendet.

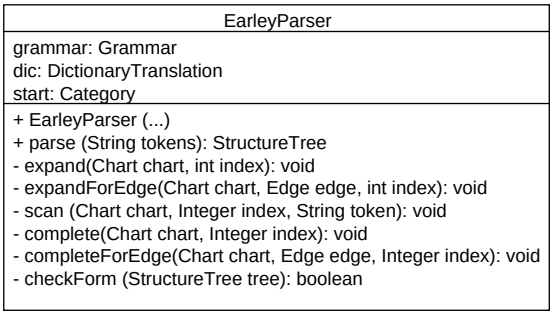

Abbildung 5.14.: Klassendiagramm der Klasse *EarleyParser*

Beim Parsen des Satzes wird eine Chart aufgebaut, diese wird durch die Klasse *Chart* implementiert. Die Chart wird durch eine SortedMap realisiert, dies kann auch als eine Art Adjazenzliste eines azyklischen Graphen aufgefasst werden. Eine Adjazenzliste ist eine Liste, in der für jeden Knoten des Graphen dessen Nachfolgeknoten gespeichert werden oder wie in diesem Fall, werden für jeden Knoten, repräsentiert durch eine Zahl, die Kanten gespeichert die zu diesem Knoten führen.

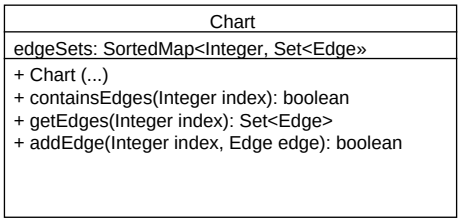

Abbildung 5.15.: Klassendiagramm der Klasse *Chart*

Eine Kante ist ein Objekt der Klasse *Edge*, unter anderem mit dem Attribut *origin*, durch das nachvollziehbar ist, welcher Knoten der Startknoten dieser Kante ist. Zu jeder Kante gehört außerdem die im Analyseschritt entstandene Regel und eine Referenz auf die Kanten aufgrund derer diese entstanden ist. Diese Referenz ist wichtig, um eine Strukturbeschreibung des geparsten Satzes extrahieren zu können.

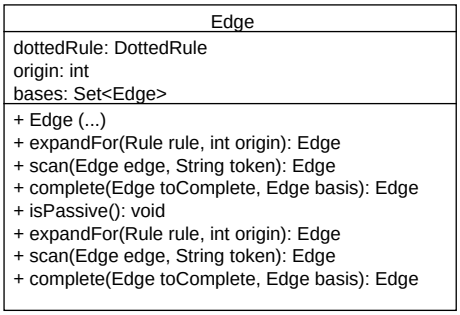

Abbildung 5.16.: Klassendiagramm der Klasse *Edge*

Bei den Regeln handelt es sich um Objekte der Klasse *DottedRule*. Diese erbt von der Klasse *Rule* und hat die zusätzlichen Attribute *dotPosition*, um eine Unterteilung in aktiven und passiven Teil der rechten Seite zu ermöglichen und *activeCategory*, welches die aktive Kategorie rechts neben dem „dot“ angibt.

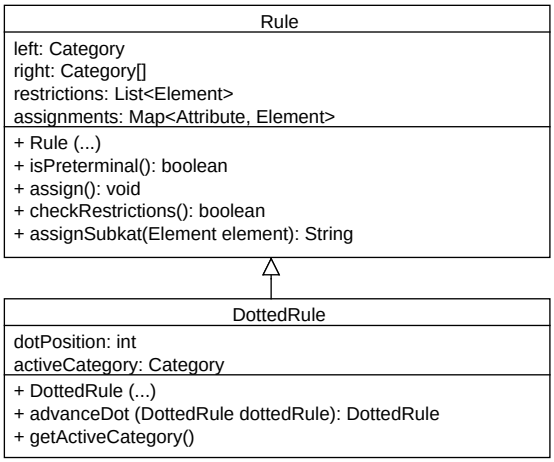

Abbildung 5.17.: Klassendiagramm der Klasse *DottedRule*

Ein Item wie es in Kapitel 2.4.2 beschrieben wird, besteht in dieser Datenstruktur aus einem Knoten und einer Kante, die bei diesem endet.

Die Implementierung des Earley-Algorithmus soll durch die Abbildung 5.18 verdeutlicht werden.

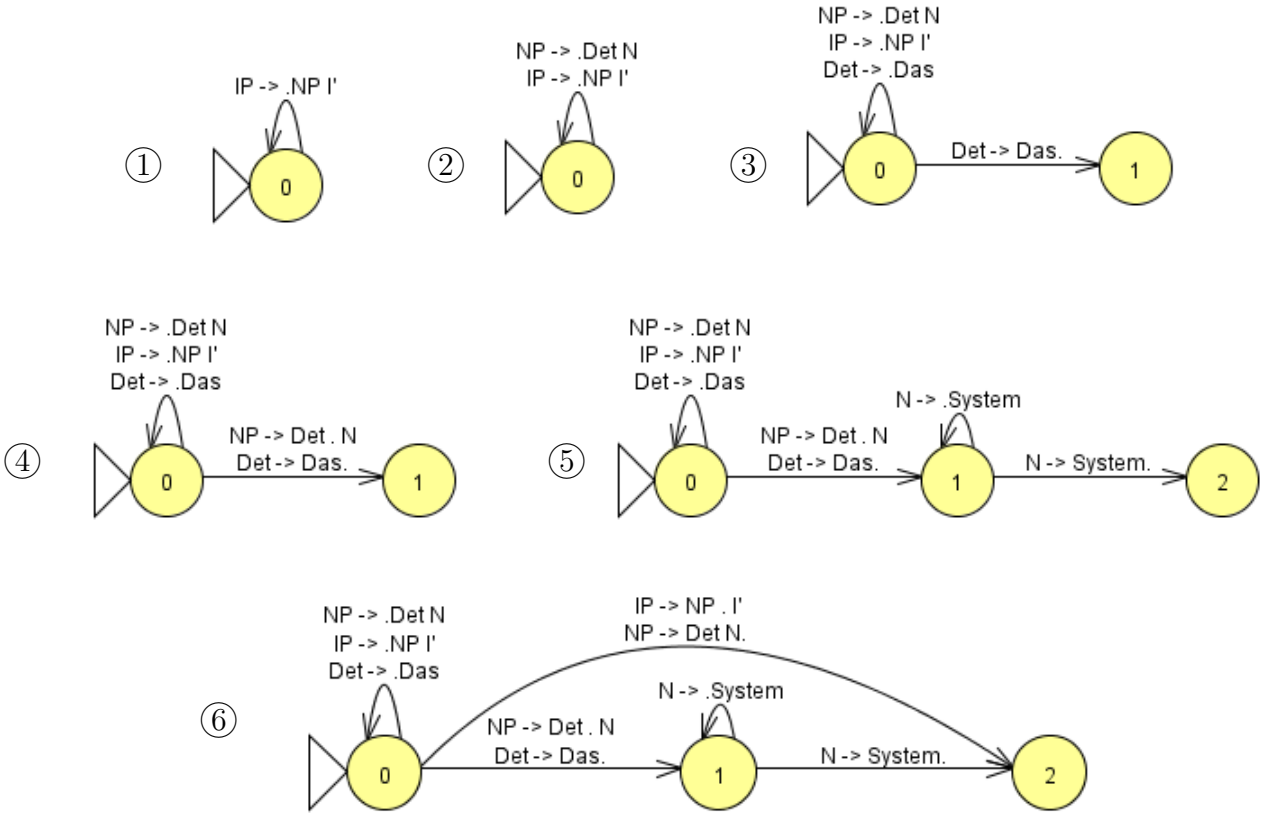

Abbildung 5.18.: Ablauf des Aufbaus einer Chart

Analysiert wird die Phrase „Das System“ mit der Basisgrammatik aus Kapitel 4.2.1. Die Knoten des Graphen entsprechen den Zwischenräume der Wörter. Es ergibt sich folgender Aufbau: ${}_0Das_1System_2$
Als Initialisierung des Parsers wird, wie in Abbildung 5.19 dargestellt, eine Kante mit der Startregel in den Graph eingefügt. Die Regel der Kante ist aktiv, startet und endet am Knoten 0, d.h. es wurde noch kein Wort erkannt.

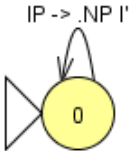

Abbildung 5.19.: Aufbau der Chart - Schritt 1

Auf diesem Graph bzw. dieser Chart werden nun die Methoden *Expand*, *Scan* und *Complete* ausgeführt, solange bis keine neue Kante mehr hinzugefügt wird[5].
Die Abbildung 5.20 zeigt den Graph, nachdem zum ersten Mal die Methode *Expand* ausgeführt wurde. Eine neue Kante mit der Regel NP → • Det N wurde hinzugefügt.

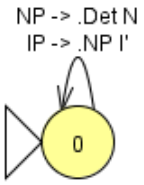

Abbildung 5.20.: Aufbau der Chart - Schritt 2

Als nächstes wird die Methode *Scan* ausgeführt, in der zunächst für jede lexikalische Regel mit dem Wort „Das“ auf der rechten Seite, eine Kante, die von und zum aktuell betrachteten Knoten geht, zum Graph hinzugefügt wird. Für jede dieser Kanten wird danach eine weitere Kante mit der selben Regel aber eine um rechts bzw. ans Ende der rechten Seite verschobene „dot“-Position, hinzugefügt. In Abbildung 5.21 wird exemplarisch für die lexikalischen Regeln mit verschiedenen Merkmalsstrukturen eine Regel eingefügt.

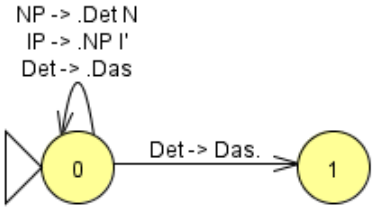

Abbildung 5.21.: Aufbau der Chart - Schritt 3

Wird eine Regel passiv, d.h. der „dot“ steht am Ende der rechten Seite, werden die Merkmale der Kategorie der linken Seite mittels der dazugehörigen Zuweisungsregeln mit Werten der rechten Seite versehen. Eine Regel wird entweder im *Scan*- oder *Complete*-Schritt passiv. Innerhalb des *Scan*-Schritts wurde das Wort „Das“ erkannt und der nächste Knoten wird bearbeitet.

[5]Definition der Methoden findet sich in Kapitel 2.4.2

Der folgende Schritt ist die Ausführung der Methode *Complete*. Die Kante mit der Regel NP → • Det N kann mittels der Kante mit der passiven Regel Det → Das • um einen Schritt vervollständigt werden. Es wird die Kante mit der Regel NP → Det • N, die von Knoten 0 bis zum Knoten 1 geht, eingefügt. Intern wird für die vervollständigte Kante ein Verweis auf die passive Kante mit der sie vervollständigt wurde, gespeichert. Dieser Schritt wird in Abbildung 5.22 dargestellt.

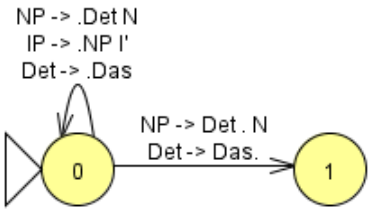

Abbildung 5.22.: Aufbau der Chart - Schritt 4

In Abbildung 5.23 wird der Graph nach Ausführung der Methoden *Expand* und *Scan* gezeigt. Während der Methode *Expand* wurden keine neuen Kanten hinzugefügt und im Schritt *Scan* die Kanten mit den Regeln N → • System und N → System •, das Vorgehen ist das selbe, wie beim „scannen“ des Wortes „Das“ in Abbildung 5.21.

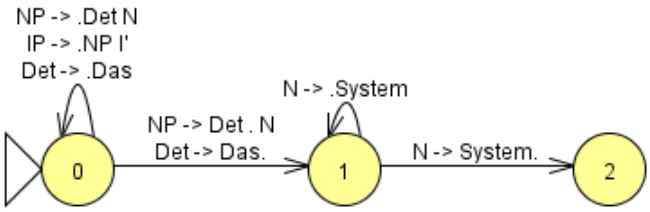

Abbildung 5.23.: Aufbau der Chart - Schritt 5

Die Abbildung 5.24 zeigt die Ausführung der Methode *Complete* für den Knoten 2. Die Kante mit der Regel NP → Det • N kann mit der Kante deren Regel N → System • lautet, vervollständigt werden. Da innerhalb des *Complete*-Schritts nun eine Regel passiv wird, werden die Merkmale der linken Seite mit Werten versehen, zuvor wird allerdings geprüft, ob die Regel auch tatsächlich vervollständigt werden kann, indem die Restriktionen bezüglich der Merkmalsstrukturen dieser Regel geprüft werden. Die Regel NP → Det N hat die in Abbildung 5.25 dargestellten Restriktionen.

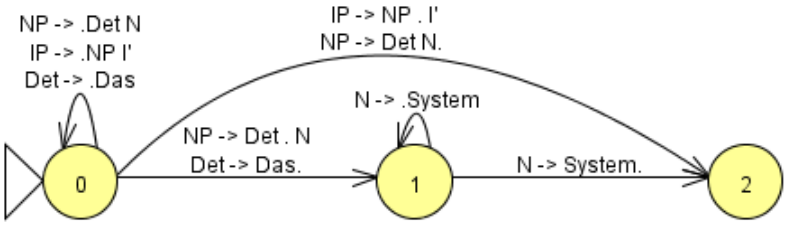

Abbildung 5.24.: Aufbau der Chart - Schritt 6

$$\underset{\begin{bmatrix} NUM:N \\ PERS:P \\ KAS:K \\ GEND:G \end{bmatrix}}{NP} \rightarrow \underset{\begin{bmatrix} NUM:N \\ PERS:P \\ KAS:K \\ GEND:G \end{bmatrix}}{Det} \quad \underset{\begin{bmatrix} NUM:N \\ PERS:P \\ KAS:K \\ GEND:G \end{bmatrix}}{N}$$

Abbildung 5.25.: Regel NP → Det N mit Merkmalsstruktur und Restriktionen

Die Merkmalsstruktur für das Wort „Das“ und die Merkmalsstruktur für das Wort „System“ können wie in Abbildung 5.26 aussehen. In den Restriktionen wird verlangt, dass Merkmale mit dem gleichen Namen auch den gleichen Wert haben. Dies trifft hier zu, d.h. die Restriktionen sind erfüllt und die Regel kann vervollständigt und als Kante in den Graph eingefügt werden.

$$\begin{bmatrix} \text{NUM} & \text{sg} \\ \text{GEND} & \text{neut} \\ \text{PERS} & 3 \\ \text{KAS} & \text{nom} \end{bmatrix}$$

Abbildung 5.26.: Mögliche Merkmalsstruktur für das Wort „Das“ und das Wort „System“

Die Kante mit der Regel IP → NP • I’ kann daraufhin auch um einen Schritt vervollständigt werden. Die Restriktionen werden allerdings nicht geprüft, da die Regel durch die Vervollständigung nicht passiv wird.

Die Tabelle 5.1 zeigt den Aufbau, der in Abbildung 5.18 beschriebenen Chart, als Tabelle. Diese ist aufgebaut, wie die Chart für das Beispiel in Kapitel 2.4.2 auf Seite 31. Die Repräsentation als Graph und die Repräsentation als Tabelle lassen sich ineinander überführen. Für die praktische Umsetzung innerhalb des Werkzeugs ist allerdings die Repräsentation als Graph geeigneter.

Nr.	**Item**		**Begründung**
1	0 0	IP → • NP I'	Initialisierung
2	0 0	NP → • Det N	EXPAND 1.
3	0 0	Det → • Das	EXPAND 2.
4	0 1	Det → Das •	SCAN 3.
5	0 1	NP → Det • N	COMPLETE 2. mit 4.
6	1 1	N → • System	EXPAND 5.
7	1 2	N → System •	SCAN 6.
8	0 2	NP → Det N •	COMPLETE 5. mit 7.
9	0 2	IP → NP • I'	COMPLETE 1. mit 8.

Tabelle 5.1.: Chart, dargestellt als Tabelle, für das Phrase „Das System“

5.2.4. Extraktion des Syntaxbaums

Für die Übersetzung des Satzes wird eine Strukturbeschreibung des Satzes benötigt. Nachdem die Chart mittels des Earley-Algorithmus vollständig aufgebaut ist, kann diese daraus extrahiert werden.

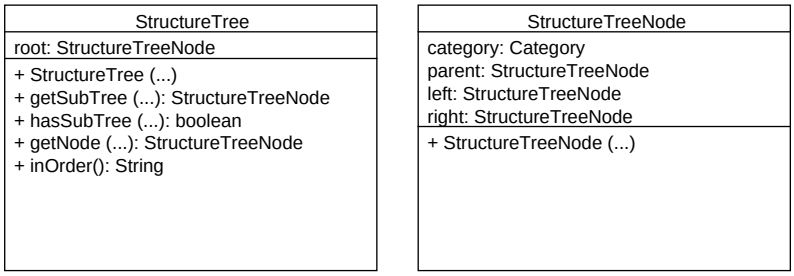

Abbildung 5.27.: Klassendiagramme der Strukturbeschreibung

Die Datenstruktur der Strukturbeschreibung, welche einem binären Syntaxbaum entspricht, ist in Abbildung 5.27 dargestellt. Sie besteht aus den Klassen *StructureTree* und *StructureTreeNode*. *StructureTree* ist die Klasse zur Verwaltung der Strukturbeschreibung. Sie besitzt ein Attribut *root*, welches ein Zeiger auf den Wurzelknoten des Syntaxbaums

ist. Die Klasse *StructureTreeNode* entspricht einem Knoten im Syntaxbaum. Als Attribute besitzt die Klasse Zeiger auf einen rechten und einen linken Teilbaum, sowie auf den Elternknoten. Außerdem entspricht jeder Knoten im Baum einer Kategorie und besitzt deswegen ein Attribut *category* vom Typ *Category*.

Um die Strukturbeschreibung aus der Chart extrahieren zu können, muss die Kante mit der passiven Startregel, die vom ersten bis zum letzten Knoten verläuft, gefunden werden. Diese Kante ist der Startpunkt für die Extraktion der Strukturbeschreibung. Für jede Kante wurde gespeichert, mit welchen anderen passiven Kanten die Regel vervollständigt wurde. Damit kann der Analyseweg zurückverfolgt werden. Die Kante mit der Regel NP → Det N • wurde in der Abbildung 5.18 beispielsweise mit den Kanten deren Regeln Det → Das • und N → System • sind, vervollständigt. Für jede Regel wird ein Teilbaum erzeugt, der linke Teil der Regel ist der Elternknoten, die rechte Seite entspricht den Kindsknoten. Ist die Regel keine lexikalische Regel werden die Kindsknoten des rechten und linken Kindes mit Hilfe der gemerkten Kanten ergänzt. Es ergibt sich für die Kante mit der Regel NP → Det N • der Teilbaum in Abbildung 5.28.

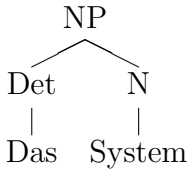

Abbildung 5.28.: Syntaxbaum für „Das System“

5.2.5. Restriktive Überprüfung des Anforderungstyps

Die Strukturbeschreibung des zu übersetzenden Satzes wird mit Hilfe der in Kapitel 4.2.1 auf Seite 49 beschriebenen Restriktionen überprüft, ob diese einer der drei Anforderungstypen, die durch die Schablone aus Kapitel 2.1.2 beschrieben werden, zugeordnet werden kann. Wenn dies nicht der Fall ist, wird der Satz nicht automatisiert übersetzt.

Die XML-Datei der Restriktionen beschreibt Teilbäume. Es gibt positiv und negativ gekennzeichnete Teilbäume. Die negativ gekennzeichneten Teilbäume sind durch umschließende <not>-Tags gekennzeichnet. Wird einer der in der XML-Datei beschriebenen positiven Bäume als Teilbaum im Syntaxbaum des Satzes gefunden, ist der Anforderungstyp bestimmt. Findet sich keiner dieser Bäume im Syntaxbaum wieder und kommt auch einer der mit <not>-Tags umschlossenen Teilbäume nicht vor, ist der Anforderungstyp auch bekannt.

Ein Auszug aus der XML-Datei ist in Abbildung 5.29 dargestellt. Ein Teilbaum wird beschrieben durch ein hierarchisches Konstrukt aus öffnenden und schließenden Tags. Die Knoten werden durch das <node>-Tag und die Angabe des Namens der Kategorie beschrieben. Außerdem können sie Kindsknoten enthalten. Diese werden wiederum duch <node>-Tags innerhalb der <node>-Tags des Elternknoten beschrieben. Die Reihenfolge der <node>-Tags der Kindsknoten beschreiben auch die Reihenfolge der Kinder im Baum. Die Blätter des Teilbaums müssen nicht unbedingt, wie die Blätter des Syntaxbaums, terminale Knoten sein.

```
<restriction>
	<node name="IS">
		<node name = "I"/>
		<node name = "VP">
			<node name = "VP">
				<node name = "V">
					<node name = "f\"ahig sein"/>
				</node>
			</node>
			<node name = "IP"/>
		</node>
	</node>

	<not>
		<node name="IS">
			<node name = "I"/>
			<node name = "VP">
				<node name = "VP"/>
				<node name = "IP"/>
			</node>
		</node>
	</not>
</restriction>
```

Abbildung 5.29.: Auszug aus der XML-Datei für die Restriktionen bezüglich der Schablone

Die XML-Datei mit den Restriktionen wird eingelesen und für jeden Teilbaum eine Datenstruktur mit Hilfe der Klassen *StructureTree* und *StructureTreeNode* aufgebaut. Die Klasse *StructureTree* stellt die Methode *getSubTree (StructureTreeNode subtree)* zur Verfügung, welche einen Zeiger auf den Knoten im Syntaxbaum zurückgibt, der Wurzelknoten des Teilbaums ist (siehe Abbildung 5.30).
Für die Teilbaumsuche wird der Syntaxbaum in Preorder durchlaufen. Preorder bedeutet, dass zuerst der Knoten selbst betrachtet wird und anschließend der linke und dann der rechte Teilbaum. Wird der Wurzelknoten des Teilbaums gefunden, wird die Rekursion abgebrochen. Beim Betrachten eines Knoten wird für diesen die Methode *hasSubTree(...)* aufgerufen. Die Methode *hasSubTree* prüft, ob der aktuell betrachtete Knoten des Syntaxbaums dem Wurzelknoten des Teilbaums entspricht. Rekursiv werden bei Übereinstimmung die Kinder der linken und rechten Teilbäume der beiden Bäume miteinander

verglichen. Stimmen auch diese überein, ist der betrachtete Knoten der Wurzelknoten des zu suchenden Teilbaums und es wird „true“ zurückgegeben.

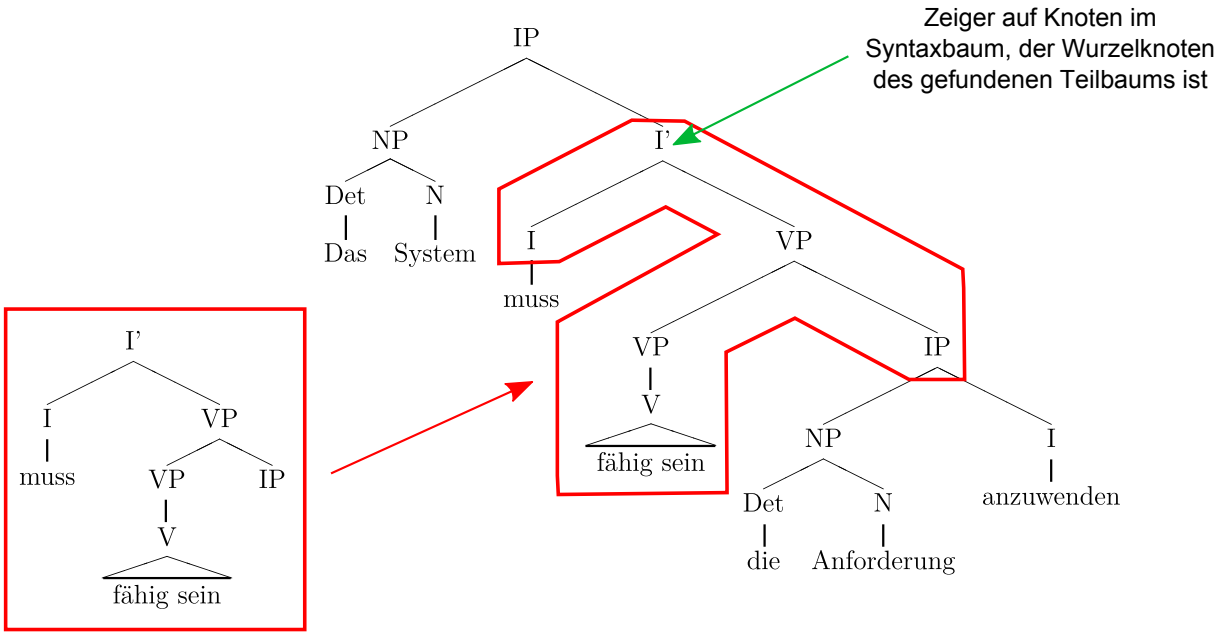

Abbildung 5.30.: Teilbaumsuche

Ist der Rückgabewert der Methode *hasSubTree* „true“, wird der aktuell betrachtete Knoten als Rückgabewert der Methode *getSubTree* zurückgegeben. Falls der Rückgabewert „false“ ist, wird die Methode *getSubTree* rekursiv mit dem linken Teilbaum des aktuellen Knoten und dem Wurzelknoten des Teilbaums aufgerufen. Ist der Rückgabewert des rekursiven Aufrufs mit dem linken Kind nicht der Null-Zeiger, wird der Rückgabewert auch als Ergebnis dieses *getSubTree* Aufrufs zurückgegeben. Wenn der Rückgabewert des rekursiven Aufrufs der Methode *getSubTree* allerdings ein Null-Zeiger ist, wird diese auch rekursiv für den rechten Kindsknoten und den Wurzelknoten des Teilbaums aufgerufen. Der Rückgabewert dieses Aufrufs ist dann endgültig auch Rückgabewert des ursprünglichen Aufrufs der Methode *getSubTree*. Der Programmcode der Methode *getSubTree* ist zum besseren Nachvollziehen des Ablaufs in Abbildung 5.31 dargestellt.

Wurde die Analyse bis hierhin durchgeführt ist der Satz korrekt geparst, d.h. sämtliche Restriktionen bezüglich der Merkmalsstrukturen wurden beachtet und überprüft, die Strukturbeschreibung wurde extrahiert und auf Richtigkeit bezüglich der Schablone kontrolliert.

```
StructureTreeNode getSubTree (StructureTreeNode subtree, StructureTreeNode node)
{
	// Teilbaum leer?
	if (subtree == null) {
		return node;
	}
	else if (subtree != null && node != null) {
		if (hasSubTree(subtree, node)) {
			return node;
		}
		else {
			if (node.left != null) {
				// rekursiver Aufruf fuer linkes Kind
				StructureTreeNode left = getSubTree (subtree, node.left);
				if (left != null) {
					return left;
				}
				else {
					if (node.right != null) {
						// rekursiver Aufruf fuer rechtes Kind
						return getSubTree(subtree, node.right);
					}
				}
			}
			else {
				if (node.right != null) {
					// rekursiver Aufruf fuer rechtes Kind
					return getSubTree(subtree, node.right);
				}
			}
		}
	}

	// Teilbaum nicht gefunden
	return null;
}
```

Abbildung 5.31.: Methode getSubTree der Klasse StructureTree

5.3. Transfer

Beim Transfer wird die aus der Analyse entstandene Strukturbeschreibung, in diesem Werkzeug ein Syntaxbaum, des eingegebenen Satzes von der Repräsentation in Quellsprache in eine Repräsentation in der Zielsprache überführt. Dabei heißt Quellsprache oder Zielsprache nicht nur, dass die Blätter des Syntaxbaums in der jeweiligen Sprache vorliegen, sondern auch, dass die Strukturbeschreibung in diesem Fall entweder durch die deutsche oder englische Basisgrammatik beschrieben wird.
Der Transfer, der in Kapitel 4.3.2 beschrieben ist, ist in mehrere Schritte aufgeteilt. Zunächst werden schablonenspezifische Transferregeln angewandt, anschließend die Subkategorisierungsrahmen der Verben übersetzt und zum Schluss die sprachspezifischen Transferregeln eingesetzt. Da das Prinzip des Transfers mittels schablonenspezifischer und sprachspezfischer Transferregeln gleich ist, wird beides in Abschnitt 5.3.1 beschrieben. Die Be-

schreibung des Transfers der Subkategorisierungsrahmen folgt in Abschnitt 5.3.2.
Trotz des gleichen Prinzips ist es wichtig, dass zuerst der schablonenspezifischen Transfer, anschließend der Transfer der Subkategorisierungsrahmen und als letztes der sprachspezifische Transfer angewandt wird (siehe Seite 58 des Kapitels 4.3.2).
Der Transfer wird von der Klasse *Transfer* gesteuert. Dazu wird dieser Klasse bei der Initialisierung die Strukturbeschreibung, die Ergebnis der Analyse ist, übergeben. Die Klasse *Transfer* besitzt eine Methode *transferTree(...)*, deren Ergebnis eine Liste von Strukturbeschreibungen, die die Übersetzungsmöglichkeiten darstellen, ist.

5.3.1. Schablonen- und sprachspezifischer Transfer

Die Transferregeln für den schablonen- bzw. sprachspezifischen Transfer wurden zur leichteren Erweiterbarkeit, wie schon in Abschnitt 5.1 motiviert, in eine XML-Datei ausgelagert. Einen Ausschnitt aus der XML-Datei der sprachspezifischen Transferregeln zeigt die Abbildung 5.32.

```
<transfer>
    <rule>
       <from>
            <node name = "VP" index = "1">
               <node name = "_" index = "2"/>
               <node name = "V" index = "3"/>
            </node>
       </from>
       <to>
            <node name = "VP" index = "1" transfer="true">
               <node name = "V" index = "3" transfer="true"/>
               <node name = "_" index = "2" transfer="true"/>
           </node>
       </to>
    </rule>
     ...
</transfer>
```

Abbildung 5.32.: Auszug aus der XML-Datei der sprachspezifischen Transferregeln

Eine Transferregel besteht aus einer Beschreibung des Teilbaums, der transferiert werden soll und einer Beschreibung des transferierten Teilbaums. Über eindeutige Indizes kann eine Zuordnung zwischen den beiden Teilbäumen gemacht werden.
Der zu transferierende Teilbaum (Startbaum) wird innerhalb von <from>-Tags definiert und ist, wie schon die Bäume in Abschnitt 5.2.5, über geschachtelte <node>-Tags aufgebaut. Zusätzlich zum Namen der Kategorie eines Knoten wird ein eindeutiger Index angegeben.
Der transferierte Baum (Zielbaum) wird innerhalb von <to>-Tags definiert. Der Index eines Knoten muss schon aus dem zu tranferierenden Teilbaum bekannt sein. Werden neue

Knoten eingeführt haben diese keinen Index. Eine Transferregel besteht immer aus einem Teilbaum innerhalb von <from>-Tags und einem Teilbaum innerhalb von <to>-Tags. In Abbildung 5.32 existiert ein Knoten mit Namen „_“. Dieses Zeichen dient als Platzhalter für eine beliebige Kategorie, die an dieser Stelle stehen kann. In Abbildung 5.33, die auch die Transferregel darstellt, steht statt einem Unterstrich ein X als Platzhalter.

Abbildung 5.33.: Prinzip des sprachspezifischen und schablonenspezifischen Transfers

Der Transfer an sich kann als Umordnung der Knoten angesehen werden. Durch die Indizes ist die Position des Knotens nach dem Transfer bekannt. Knoten können an ihrer Position bleiben, die Position ändern, wegfallen oder hinzu kommen. Abbildung 5.33 zeigt die Umordnung der Knoten nach Vorgabe durch die Transferregel.

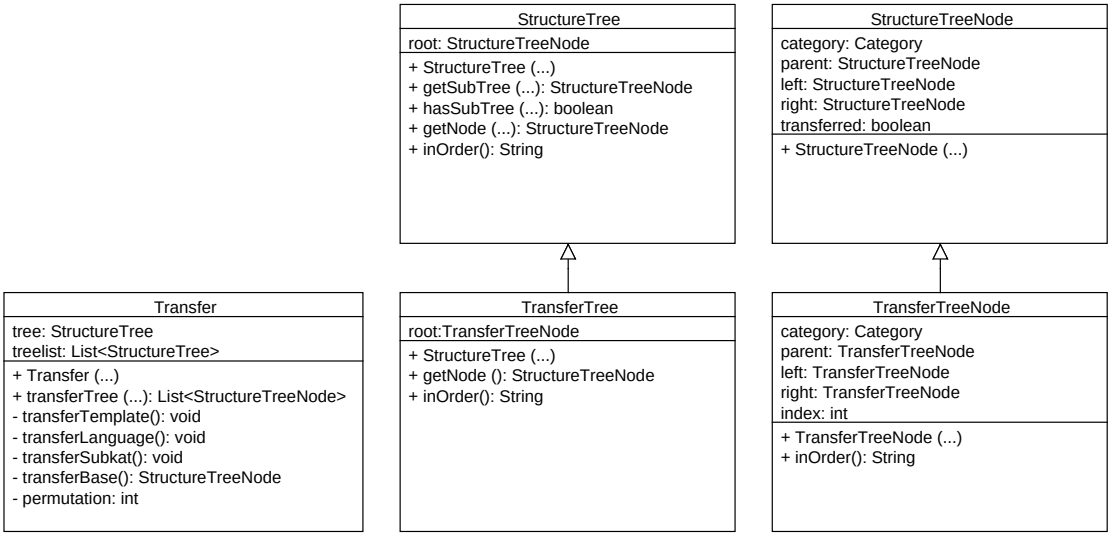

Abbildung 5.34.: Klassendiagramme des Transfers

Die Datenstruktur für den Transfer ist in Abbildung 5.34 dargestellt. Die Klasse *Transfer* steuert den Transfer der Strukturbeschreibung. Dazu bietet sie die Methode *transferTree* an, innerhalb derer nacheinander die Methoden *transferTemplate()*, *transferSubkat()* und *transferLanguage()* aufgerufen werden. In den Methoden *transferTemplate()* und *transferLanguage()* wird als erstes die eben beschriebene XML-Datei mit den schablonenspezifischen bzw. sprachspezifischen Transferregeln eingelesen. Für jede Regel wird ein Startbaum und ein Zielbaum aufgebaut. Diese Bäume sind Objekte der Klasse *TransferTree*,

die ein Attribut *root* enthält, das auf den Wurzelknoten, ein Objekt der Klasse *TransferTreeNode*, zeigt.
Die Klasse *TransferTree* erbt von der Klasse *StructureTree* und die Klasse *TransferTreeNode* von der Klasse *StructureTreeNode*. Um den Index, der jedem Knoten in der XML-Datei zugewiesen wird, zu speichern, bekommt die Klasse *TransferTreeNode* ein zusätzliches Attribut *index*. Mit der Methode *getSubTree* der Klasse *StructureTree* wird der Startbaum als Teilbaum im Syntaxbaum gesucht (siehe Abschnitt 5.2.5). Rückgabe der Methode ist ein Zeiger auf den Wurzelknoten des Teilbaums im Syntaxbaum. Dieser Teilbaum soll nun transferiert werden, falls er nicht schon als transferiert markiert wurde (Attribut *transferred*). Für den Transfer eines Teilbaums wird die Methode *transferSubTree* der Klasse *Transfer* aufgerufen. Dieser Methode wird unter anderem der Startbaum, der Zielbaum und der Teilbaum des Syntaxbaums übergeben.

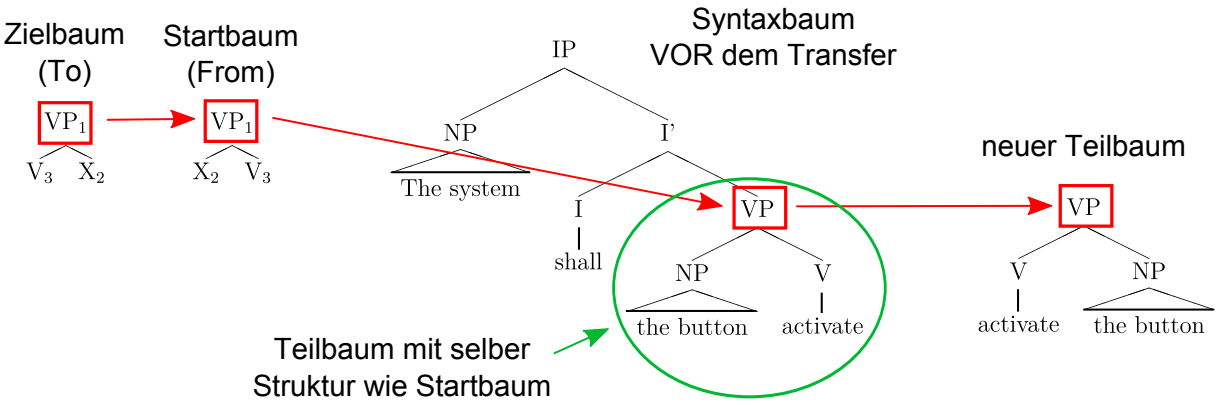

Abbildung 5.35.: Ablauf der Methode *transferSubTree*

Mit *transferSubTree* wird der Zielbaum in Preorder durchlaufen (Erklärung für Preorder siehe Seite 78). Parallel zum Zielbaum wird ein äquivalenter Baum aufgebaut allerdings werden die Knoten mit den Informationen aus dem Syntaxbaum gefüllt. Für jeden Knoten des Zielbaums wird mit Hilfe des Index der zugehörige Knoten im Startbaum gesucht. Da der Startbaum die selbe Struktur hat, wie der ermittelte Teilbaum des Syntaxbaums, ist auch der Knoten im Syntaxbaum bekannt. Aus diesem werden nun die Informationen z.B. die Kategorie und Merkmale für die Erzeugung eines neuen Knotens genommen. Die Kinder werden zunächst auch übernommen, allerdings kann es sein, dass diese im weiteren Verlauf des Transferschritts verändert, also zum Beispiel durch einen neuen Knoten ersetzt werden.
Hat der Knoten des Zielbaums keinen Index wird ein neuer Knoten erzeugt. Er wird gefüllt mit den Informationen des Knotens aus dem Zielbaum. Rückgabe der Methode *transferSubTree* ist der neu aufgebaute und damit transferierte Teilbaum (siehe Abbildung 5.35). Dieser muss dann noch innerhalb von *transferLanguage* oder *transferTemplate* an der richtigen Stelle im ursprünglichen Syntaxbaum eingehängt werden (siehe Abbildung 5.36).

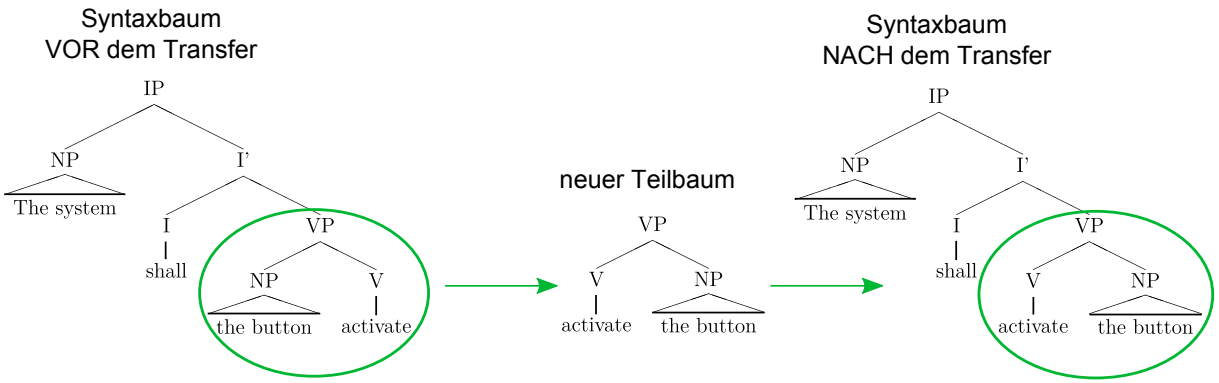

Abbildung 5.36.: Ablauf der Methoden *transferLanguage* oder *transferTemplate*

5.3.2. Transfer der Subkategorisierungsrahmen

Der Transfer der Subkategorisierungsrahmen wird zwischen dem schablonenspezifischen und sprachspezifischen Transfer durchgeführt. Das Prinzip des Transfers der Subkategorisierungsrahmen wurde bereits in Kapitel 4.3.2 erläutert. Im Gegensatz zu den beiden anderen Transferschritten gibt es hierfür keine expliziten Transferregeln, die in eine XML-Datei ausgelagert werden können. Für den Transfer der Subkategorisierungsrahmen wird nur die Strukturbeschreibung, der deutsche Subkategorisierungsrahmen, sowie die dazugehörigen englischen Subkategorisierungsrahmen jedes Verbs benötigt.
Zunächst muss ein Verb, das noch nicht transferiert wurde, im Syntaxbaum gefunden werden. Dazu wird der Baum in Preorder durchlaufen und nach terminalen Knoten gesucht, die das Merkmal „SUBKAT“ (deutscher Subkategorisierungsrahmen) und „SUBKAT E“ (englische Subkategorisierungsrahmen) besitzen. Dann werden für jedes Verb die passenden englischen Subkategorisierungsrahmen, für die eine Übersetzung generiert werden soll, ermittelt, dies sind alle Subkategorisierungsrahmen mit gleich vielen Elementen, wie die Anzahl der Elemente im deutschen Subkategorisierungsrahmen. Für jeden passenden englischen Subkategorisierungrahmen wird eine Übersetzung erzeugt. Dazu werden die Elemente des deutschen und englischen Subkategorisierungsrahmen in je einer Liste gespeichert. Nachdem aus der englischen Liste das Element „subj“ und aus der deutschen Liste das Element „np:nom“ gelöscht wurde, wird für jede Zuordnungsmöglichkeit zwischen den verbleibenden Elementen eine Übersetzung generiert [6]. Bevor dies jedoch möglich ist, müssen erst noch die Elemente des deutschen Subkategorisierungsrahmen im Syntaxbaum gesucht werden. Dies wird durch die Anwendung der Zuweisungsregeln auf dem Baum, in denen die Elemente der Subkategorisierungsrahmen identifiziert werden, realisiert. Die Zeiger auf die Elemente des deutschen Subkategorisierungrahmen werden den Elementen, die in der deutschen Liste gespeichert wurden, zugewiesen. So ist die Position jedes Elements im Baum bekannt.

[6] Beschreibung der Subkategorisierungsrahmen siehe Seite 51 in Kapitel 4.2.2

Die Zuordnungsmöglichkeiten zwischen den beiden Subkategorisierungsrahmen mit je n Elementen, entspricht der Anordnungsmöglichkeit von n Elementen, da die Position der englischen Elemente fix ist und diesen die deutschen Elemente in jeder möglichen Kombination zugeordnet werden. Damit keine der n! Möglichkeiten vergessen wird, wird eine Matrix mit allen Permutationsmöglichkeiten der deutschen Elemente zur Hilfe genommen. Die Matrix enthält die permutierten Indexpositionen der deutschen Elemente. Abbildung 5.37 zeigt eine solche Matrix für zwei Subkategorisierungsrahmen mit je drei Elementen. Die Nummerierung der Spalten entspricht den Indizes der englischen Elemente, die Zahlen in der jeweiligen Zeile entsprechen den dazugehörigen Indizes der deutschen Elemente.

$$\begin{pmatrix} 1 & 2 & 3 \\ 1 & 3 & 2 \\ 2 & 1 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 2 \\ 3 & 2 & 1 \end{pmatrix}$$

Abbildung 5.37.: Matrix mit Permutation der Zahlen 1 bis 3

Abbildung 5.38 zeigt die beiden Listen des Wortes „geben“ aus dem Beispiel aus Kapitel 4.3.2 mit den Elementen der Subkategorisierungsrahmen und die Zuordnung bei entsprechendem Matrixeintrag. Ist die Zeile in der Matrix beispielsweise der Eintrag „(2 1)„, wird dem ersten Element des englischen Subkategorisierungrahmen das zweite Element des deutschen Subkategorisierungsrahmen und dem zweiten englischen Element das erste deutsche Element zugeordnet. Ist die Zuordnung zwischen den beiden Listen bestimmt, muss ein Basistransfer zwischen den Elementpaaren durchgeführt werden. Dies ist im Werkzeug fest implementiert und kann in allen Kombinationen zwischen einer NP und einer PP durchgeführt werden. Das neu erzeugte Element wird im Baum an der entsprechenden Stelle, die bereits über die Zuweisungsregeln bestimmt wurde, eingehängt.
Für jede Zuordnungsmöglichkeit wird ein neuer Syntaxbaum erzeugt, dieser stellt dann eine Übersetzungsmöglichkeit dar.

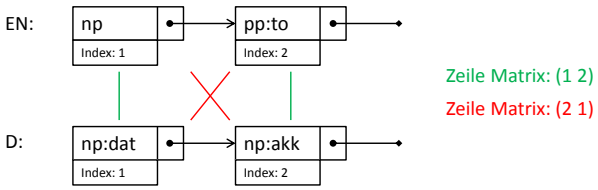

Abbildung 5.38.: Zuordnung der Elemente zweier Subkategorisierungsrahmen mit Hilfe von Permutationen

5.4. Generierung

Die Generierung ist der letzte Schritt der transferbasierten Übersetzung. Jede Strukturbeschreibung, die Ergebnis des Transfers ist, also eine Übersetzungsmöglichkeit darstellt, wird in Inorder durchlaufen, um den Satz daraus zu generieren. Die Reihenfolge in der die Knoten betrachtet werden, ist in Abbildung 5.39 dargestellt. Der resutierende Satz besteht aus den Blättern des Baums, die in der Reihenfolge, in der sie angetroffen werden, ausgegeben werden.

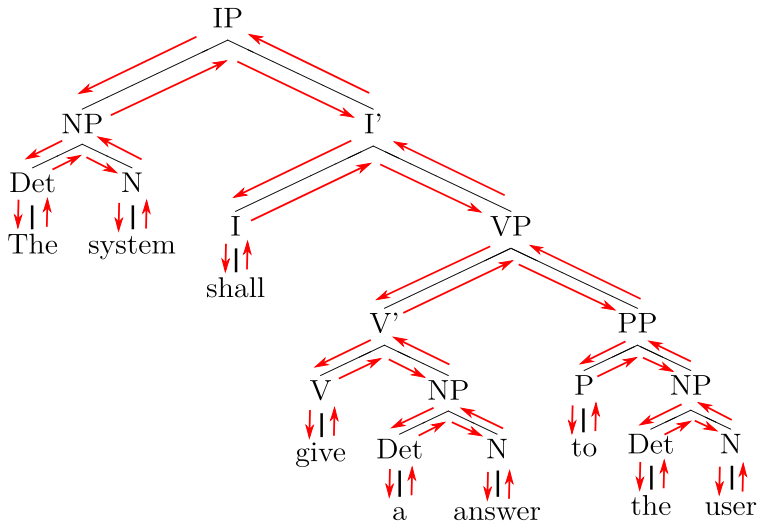

Abbildung 5.39.: Inorder-Durchlauf der Strukturbeschreibung

5.5. Benutzeroberfläche

Die Benutzeroberfläche des Werkzeugs basiert auf dem Werkzeug von Hoedoro [Hoe11], wobei dieses an die neue Funktion angepasst wurde. Der Benutzer gibt in das Textfeld des Anforderungswidgets eine Anforderung ein. Darüber wird, wie in Abbildung 5.40 gezeigt, während der Eingabe die Struktur der Schablone eingeblendet. Das System von Hoedoro erkennt nach welcher Schablone der Satz geschrieben wird. Hoedoros Werkzeug erkennt sowohl Sätze nach der Schablone aus Abbildung 2.5 als auch Sätze nach der Schablone aus Abbildung 2.6. Im Vorschauwidget werden für jedes begonnene Wort Vorschläge zur Vervollständigung angeboten. Diese Funktionalität war schon durch das Werkzeug gegeben. Erweitert wurde es um die Übersetzungsfunktion, die allerdings nur Übersetzungen von Sätzen, die nach der in Abbildung 2.5 gezeigten Schablone geschrieben wurden, anbietet.

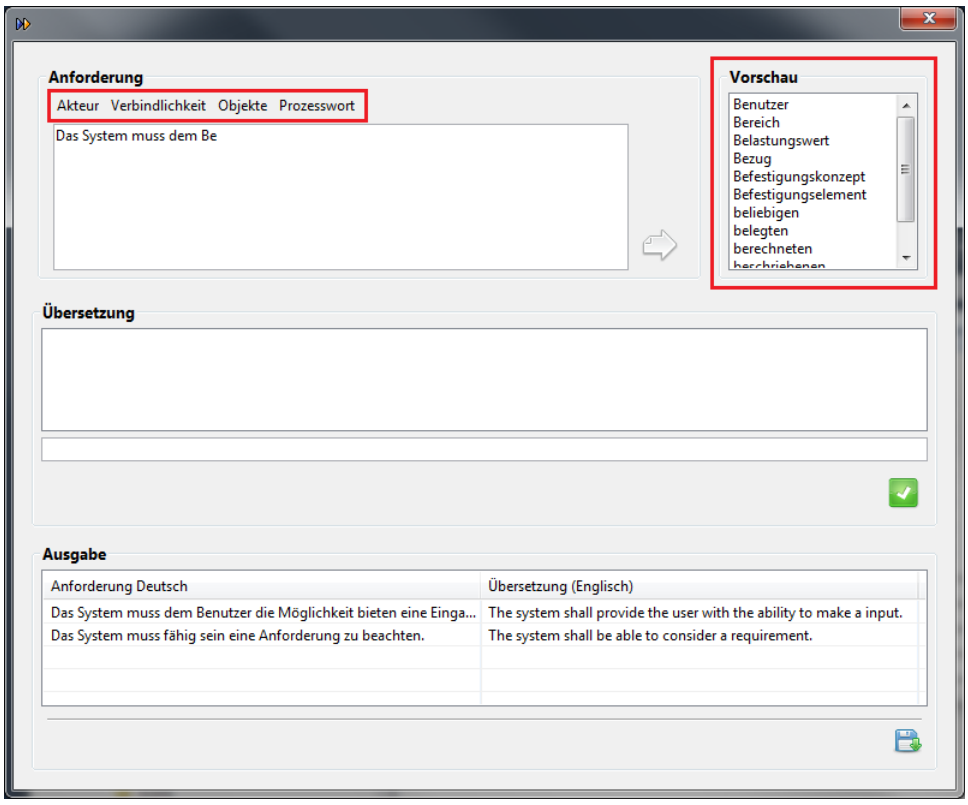

Abbildung 5.40.: Screenshot des Werkzeugs - Anforderung- und Vorschaufenster

Für die Übersetzung wurde zur Benutzeroberfläche ein neues Widget hinzugefügt. Wird in das Textfeld des Anforderungswidgets eine Anforderung eingegeben und anschließend auf den Button mit dem Pfeil, der sich direkt daneben befindet, geklickt, werden alle Übersetzungsmöglichkeiten für diesen Satz generiert und in einer scrollbaren Liste angezeigt. Die

Übersetzungsmöglichkeiten können durch anklicken ausgewählt werden. Die ausgewählte Übersetzung wird in das Textfeld unter der Liste mit den Übersetzungsmöglichkeiten eingefügt. Hier kann diese vom Benutzer noch verbessert oder ergänzt werden.

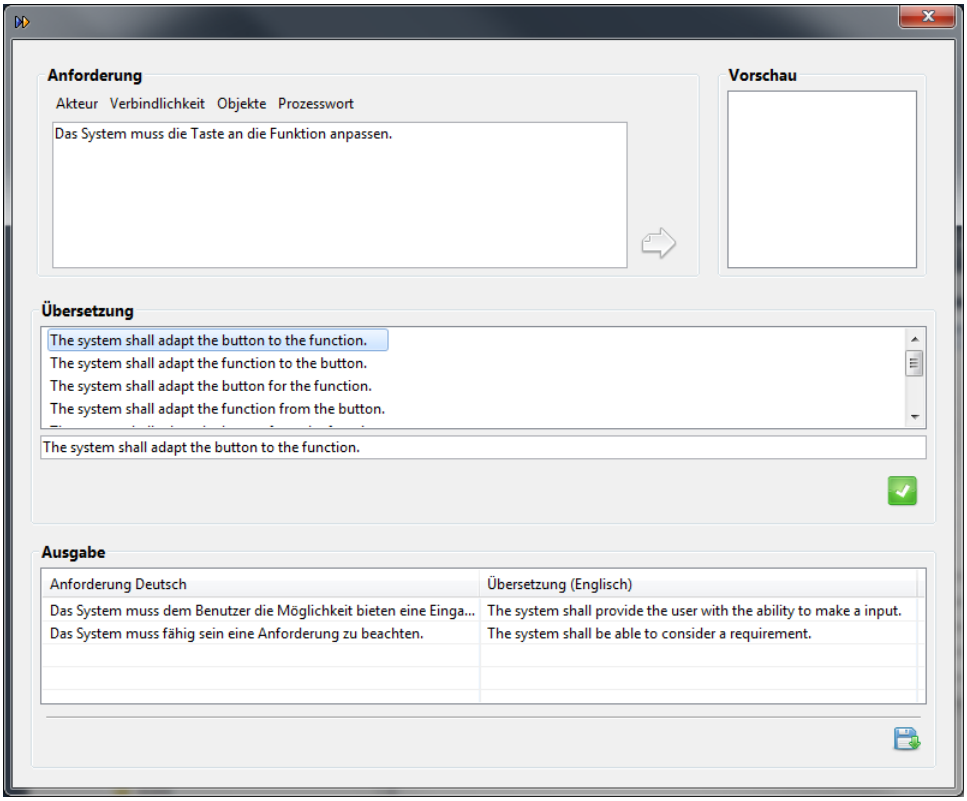

Abbildung 5.41.: Screenshot des Werkzeugs - Übersetzungsmöglichkeiten

Ist der Benutzer mit der Übersetzung zufrieden, kann er auf den Button rechts unter dem Textfeld mit der Übersetzung klicken, um den Satz und die Übersetzung in seine Ausgabe zu übernehmen. Im unteren Bereich der Benutzeroberfläche befindet sich eine Tabelle, in der sowohl die deutsche Anforderung als auch die dazugehörige Übersetzung angezeigt wird. Mit einem Klick auf das Diskettensymbol können alle Anforderungen und ihre Übersetzungen in einer Textdatei gespeichert werden. Die Benutzeroberfläche wird dann wieder zurückgesetzt, d.h. alle Tabellen, Listen und Textfelder sind leer.
Kann für einen Satz keine Übersetzung erzeugt werden, wird im Übersetzungswidget, wie in Abbildung 5.42 abgebildet, der Satz „Die Anforderung konnte nicht übersetzt werden.“ angezeigt. Dieser Satz kann nicht als Übersetzung ausgewählt werden. Es kann aber generell immer eine eigene Übersetzung in das entsprechende Textfeld eingegeben werden. Falls keine Übersetzung eingegeben wird bzw. von den angebotenen Übersetzungsmöglichkeiten keine ausgewählt wird, kann der Satz nicht in die Ausgabetabelle übernommen werden.

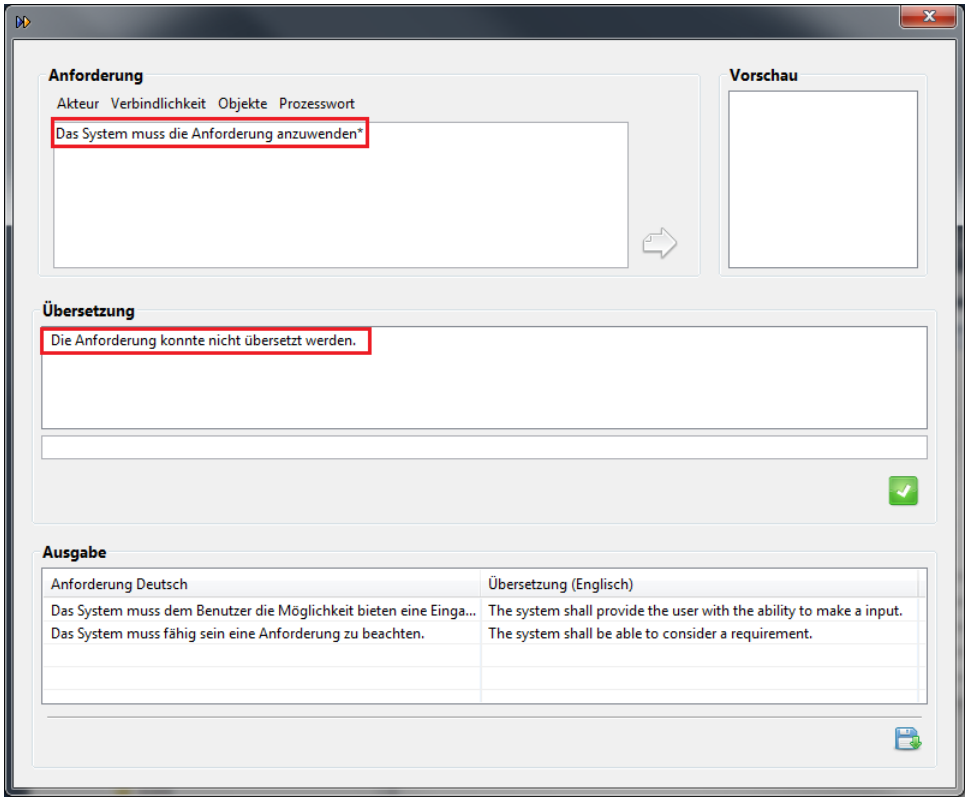

Abbildung 5.42.: Screenshot des Werkzeugs - Übersetzungsfunktion

Ein Satz, der nicht korrekt nach der Schablone geschrieben wurde bzw. dessen Wörter nicht aus dem Lexikon bekannt sind, wird nachdem der Button zur Erzeugung einer Übersetzung angeklickt wurde, mit einem Sternchen am Ende markiert. Diese Funktion hat Hoedoro implementiert.

6. Schlussbetrachtung

6.1. Zusammenfassung

In dieser Arbeit wurde eine Methodik zur maschinellen Übersetzung von Anforderungen vorgestellt, die in einem Werkzeug umgesetzt wurde. Mit dem Werkzeug können Anforderungen, die nach einer Schablone geschrieben wurden, von Deutsch nach Englisch übersetzt werden. Durch die Schablone wird die Syntax des Satzes auf drei verschiedene Anforderungstypen eingeschränkt. Es handelt sich damit um eine kontrollierte Sprache, die leichter übersetzt werden kann. Zusätzlich wird die Übersetzung durch die Verwendung eines Lexikons mit einer Subsprache vereinfacht.
Für eine maschinelle Übersetzung gibt es verschiedene Ansätze. In diesem Fall hat sich das transferbasierte Verfahren als am geeignetsten herausgestellt. Eine transferbasierte Übersetzung besteht aus Analyse, Transfer und Generierung.
Zur Analyse wird eine um Merkmalsstrukturen und Restriktionen ergänzte Phrasenstrukturgrammatik verwendet. Merkmalsstrukturen und Restriktionen werden benötigt, weil die Phrasenstrukturgrammatik übergeneriert. Die lexikalischen Regeln der Grammatik werden für jedes Token des zu übersetzenden Satzes aus dem Lexikon erzeugt und zur Grammatik hinzugefügt. Diese Grammatik ist dann Grundlage für einen Earley-Parser, der als Analyseergebnis eine Chart liefert, aus der nach erfolgreichem Parsen des Satzes ein Syntaxbaum extrahiert werden kann. Für den extrahierten Syntaxbaum wird mittels Restriktionen geprüft, ob er einem der drei Anforderungstypen entspricht.
Der nächste Schritt ist der Transfer. Der Syntaxbaum, der nach der deutschen Grammatik aufgebaut wurde und Ergebnis der Analyse ist, wird über Transferregeln in einen Syntaxbaum, der durch die englische Grammatik beschrieben wird, transferiert. Der Transfer ist in mehrere Schritte aufgeteilt. Als erstes wird die deutsche Schablonenstruktur, die im Baum enthalten ist, in die englische Schablonenstruktur überführt. Dann wird das Wissen über die Subkategorisierungsrahmen der Verben ausgenutzt um den deutschen Subkategorisierungsrahmen in die passenden englischen Subkategorisierungsrahmen zu überführen. Es entstehen unter Umständen mehrere Übersetzungsmöglichkeiten, da es keine eindeutige Zuordnung zwischen den deutschen und englischen Subkategorisierungsrahmen gibt. Es kann wegen fehlendem Wissen auch keine der Übersetzungsmöglichkeiten ausgeschlossen werden. Der Transfer wird beendet durch die Anwendung von sprachspezifischen Regeln auf die verschiedenen Syntaxbäume des vorhergehenden Transferschritts. Die sprachspezifischen Regeln korrigieren die Verbstellung, die in deutschen und englischen Sätzen unterschiedlich ist.

Ergebnis des Transfers ist mindestens ein, eventuell auch mehrere, Syntaxbäume. Jeder dieser Bäume wird in Inorder durchlaufen um den übersetzten Satz daraus zu generieren. Das Werkzeug bietet eine Eingabemaske für Anforderungen, übersetzt diese nach dem beschrieben Verfahren und stellt, falls nötig, dem Benutzer mehrere Übersetzungsmöglichkeiten als Auswahl zur Verfügung.

6.2. Fazit und Ausblick

Mit dem entwickelten Werkzeug wird demonstriert, dass eine maschinelle Übersetzung für Anforderungen sinnvoll und umsetzbar ist. Sind alle Tokens des Satzes als Einträge im Lexikon vorhanden und entspricht der Satz der Schablone kann auch eine Übersetzung generiert werden. Um das Werkzeug allerdings ausgiebig zu testen, wäre eine Benutzerstudie notwendig. Die Kandidaten müssten sich sowohl fachlich mit den Anforderungen auskennen, als auch am besten englische Muttersprachler sein. Eine andere Evaluierung wäre subjektiv und daher nicht repräsentativ.
Die Abbildung 6.1 zeigt eine Anregung für Fragen, die in einer Evaluierung beantwortet werden könnten. Durch diese Fragen könnte eine Bewertung des Werkzeugs erfolgen.

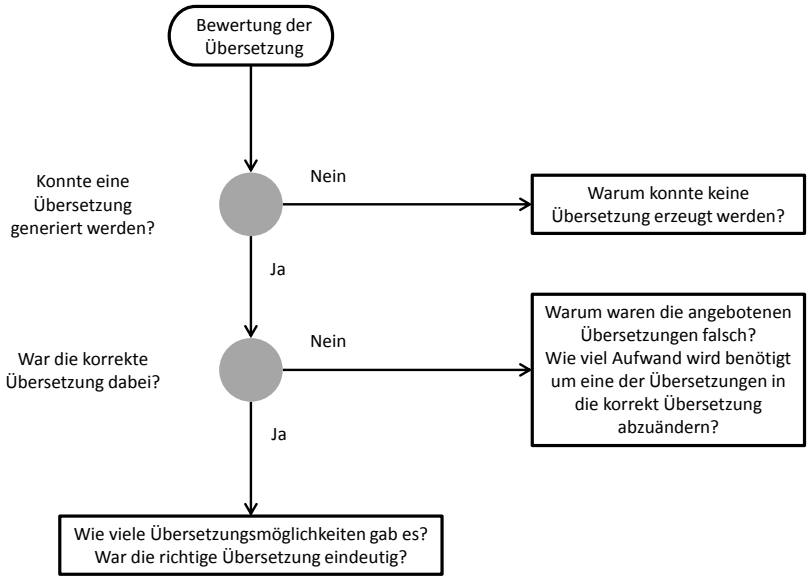

Abbildung 6.1.: Bewertung einer Übersetzung

Dennoch soll im Folgenden ein Eindruck über den bisherigen Stand und mögliche Erweiterungen gegeben werden.

Das Werkzeug ist interaktiv und erlaubt eine Postedition. Dies gibt dem Benutzer die Möglichkeit eine korrekte Übersetzung zu wählen oder die Übersetzung zu verbessern. Dadurch wird fehlendes Wissen ausgeglichen und die Übersetzungsqualität angehoben. Zusätzlich nutzt das Werkzeug zur Verbesserung der Übersetzung eine kontrollierte Sprache, gegeben durch die Schablone und eine Subsprache, gegeben durch das Lexikon.
Da das Werkzeug schon interaktiv gestaltet wurde, könnte eine zusätzliche Funktionalität sein, den Benutzer beim Start des Werkzeugs auswählen zu lassen in welcher Domäne, z. B. Fahrerassistenzsysteme oder Motoren, er seine Anforderungen schreiben möchte. Dies macht das Lexikon flexibler und nutzt die Tatsache, dass die Übersetzung der Wörter in einer kleineren Domäne häufig eindeutig ist.
Es hat sich gezeigt, dass das Einbeziehen der syntaktischen Merkmale aus dem Lexikon, vor allem der Subkategorisierungsrahmen, für eine korrekte Übersetzung besonders wichtig ist. Dadurch wurde erreicht, dass wenn eine Übersetzung erzeugt werden kann, meist auch die korrekte dabei ist. Dieser Eindruck ist natürlich rein subjektiv und das Werkzeug wurde daraufhin noch nicht ausreichend getestet. Allerdings kann schon jetzt festgestellt werden, dass das Lexikon vor allem bezüglich der Übersetzungen und Subkategorisierungsrahmen noch stark verbesserungswürdig ist.
Das System wurde an vielen Stellen, zum Beispiel durch das Auslagern der Grammatik in eine XML-Datei, erweiterbar gestaltet. Die maschinelle Übersetzung ist ein interdisziplinäres Gebiet, in dem nicht nur Informatiker arbeiten. Experten, wie zum Beispiel Computerlinguisten, können das Werkzeug mit ihrem Wissen erweitern, um zum Beispiel eine größere Abdeckung der Anforderungstypen zu ermöglichen. Der Grammatik fehlen unter anderem Relativsätze und Vorbedingungen. Auch das Lexikon wurde in Textdateien ausgelagert. Dadurch können fehlende Begrifflichkeiten ergänzt oder neue Domänen eingeführt werden. Alternativ kann als Verbesserung das Lexikon von einem Vollformenlexikon in ein Lexikon mit einer morphologischen Analyse transferiert werden. Dies würde vor allem Speicherplatz sparen und mehr Wörter abdecken. Schnellere Zugriffszeiten könnten durch die Verwaltung des Lexikons als SQL-Datenbank realisiert werden.
Wichtig wäre noch eine Überprüfung der Werte der Merkmale eines Wortes. Der Kasus kann beispielsweise nur die Werte Nominativ, Dativ, Genitiv und Akkusativ annehmen. Andere Werte sind nicht zulässig.
Mit einer zusätzlichen Logik bezüglich der Subkategorisierungrahmen könnten fehlerhafte Übersetzungsmöglichkeiten ausgeschlossen werden. Hat zum Beispiel ein deutsches Verb den Subkategorisierungrahmen <np:nom, np:akk, pp:auf_dat> und ein passender englischer Subkategorisierungsrahmen wäre <subj, np, pp:on>, dann ist eine Zuordnung der beiden PPs und der beiden NPs zueinander am wahrscheinlichsten. Die andere Möglichkeit der Zuordnung könnte ausgeschlossen werden. Die beiden Subjekte *np:nom* und *subj* werden immer einander zugeordnet und werden schon jetzt separat betrachtet.
Ein weiterer erheblicher Vorteil der maschinellen Übersetzung ist die einheitliche Struktur der Übersetzungen. Dies kann natürlich bei einer anschließenden Editierung durch den Benutzer nicht gewährleistet werden. Das Werkzeug könnte aber dahingehend erweitert werden, dass nachdem der Benutzer eine Übersetzung editiert oder eingegeben hat, diese nochmals auf Korrektheit bezüglich der Schablone überprüft wird.
Ein letzter Vorschlag zur Verbesserung des Werkzeugs ist das transferbasierte System zu

einem Hybridsystem zu erweitern. Es könnte beispielsweise mit einem wissensbasierten oder statistischen Verfahren optimiert oder mit künstlicher Intelligenz zu einem lernenden System ausgebaut werden. Die neusten Systeme sind fast ausschließlich Hybridsysteme. Die maschinelle Übersetzung hat Zukunft im Bereich der Anforderungsübersetzung, denn die Möglichkeiten sind noch nicht ausgeschöpft und die Aussichten optimistisch. Diese Arbeit ist definitv ein Anfang in diesem Bereich.

Literaturverzeichnis

[Bal09] H. Balzert. *Lehrbuch Der Softwaretechnik: Basiskonzepte und Requirements Engineering*. Spektrum Lehrbücher der Informatik. Springer-Verlag New York Inc, 2009.

[BEW06] J. Butt, C. Eulitz, and Wiemer. Vorlesung: Einführung in die Linguistik - Syntax 3. `http://ling.uni-konstanz.de/pages/allgemein/study/introling06/einf_syntax3-print.pdf`, 2006.

[Brä06] H. Bräuer. Generative Grammatik. `http://tu-dresden.de/die_tu_dresden/fakultaeten/philosophische_fakultaet/iph/thph/braeuer/lehre/putnam_bedeutung/Generative%20Grammatik.pdf`, 2006.

[Bre01] J. Bresnan. *Lexical-Functional Syntax*. Blackwell Textbooks in Linguistics. Blackwell, 2001.

[Car02] A. Carnie. *Syntax: a generative introduction*. Introducing linguistics. Blackwell Publishers, 2002.

[CEE+04] K. Carstensen, C. Ebert, C. Endriss, S. Jekat, R. Klabunde, and H. Langer. *Computerlinguistik und Sprachtechnologie - Eine Einführung*. Spektrum Akademischer Verlag, 2. edition, 2004.

[Cho56] N. Chomsky. Three models for the description of language. *IRE Transactions on Information Theory*, 2, 1956. `http://www.chomsky.info/articles/195609--.pdf`.

[Ebe09] K. Eberle. Integration von regel- und statistikbasierten Methoden in der maschinellen Übersetzung. *LDV Forum*, 24(3):37–70, 2009.

[Ela11] H. Elazhary. Translation of Software Requirements. *International Journal of Scientific and Engineering Research*, 2, 2011.

[Gol11] J. Goll. *Methoden und Architekturen der Softwaretechnik*. Vieweg Studium. Vieweg+teubner Verlag, 2011.

[Hei04] U. Heid. Vorlesung: Maschinelle Übersetzung 1, 2004.

[Hoe11] N. Hoedoro. Entwicklung eines interaktiven Werkzeugs zum Verfassen natürlichsprachlicher Spezifikationen, 2011.

[HS92] H. Hutchins and H. Somers. *An introduction to machine translation*. Academic Press, 1992.

[IEE90] IEEE Standard Glossary of Software Engineering Terminology. *IEEE Std 610.12-1990*, 1990.

[NL94] S. Naumann and H. Langer. *Parsing - eine Einführung in die maschinelle Analyse natürlicher Sprache*. Teubner, 1994.

[Par10] H. Partsch. *Requirements-Engineering systematisch: Modellbildung für softwaregestützte Systeme*. Springer, 2010.

[Ram09] M. Ramlow. *Die maschinelle Simulierbarkeit des Humanübersetzens*. Frank&Timme, 2009.

[Rei09] K. Reif. *Automobilelektronik: Eine Einführung für Ingenieure*. Atz/Mtz-Fachbuch. Vieweg+teubner Verlag, 2009.

[RN04] S. Russell and P. Norvig. *Künstliche Intelligenz: Ein moderner Ansatz*. Pearson Studium, 2004.

[Rup09] C. Rupp. *Requirements-Engineering und -Management*. HANSER, 2009.

[Sah06] S. Sahel. Vorlesung Theorien und Modelle 1 - X-Bar-Syntax. `http://www.uni-bielefeld.de/lili/personen/ssahel/theorien_modelle1/x-bar-syntax.pdf`, 2006.

[Sah09] S. Sahel. Struktur der deutschen Sprache 2: Der Satz. `http://wwwedit.uni-bielefeld.de/lili/personen/ssahel/struktur2_ss09/x-bar_theorie.pdf`, 2009.

[Sch03a] U. Schöning. *Theoretische Informatik - kurzgefasst*. Spektrum Akademischer Verlag, 4. edition, 2003.

[Sch03b] S. Schulte im Walde. *Experiments on the Automatic Induction of German Semantic Verb Classes*. PhD thesis, Institut für Maschinelle Sprachverarbeitung, Universität Stuttgart, 2003. Published as AIMS Report 9(2).

[Sch04] H. Schmid. Efficient Parsing of Highly Ambiguous Context-Free Grammars with Bit Vectors. *Proceedings of the 20th International Conference on Computatio-*

nal Linguistics (COLING 2004), 2004. http://www.ims.uni-stuttgart.de/projekte/gramotron/PAPERS/COLING04/BitPar.pdf.

[Sco07] M. Scott. Pep 0.4: Pep is an Earley parser. http://www.ling.ohio-state.edu/~scott/projects/pep/docs/api/, 2007.

[SFH02] H. Schmid, A. Fitschen, and U. Heid. SMOR: A German Computational Morphology Covering Derivation, Composition and Inflection. *Proceedings of the 19th International Conference on Computational Linguistics (COLING 2002)*, 2002. www.ims.uni-stuttgart.de/projekte/gramotron/PAPERS/LREC04/smor.pdf.

A. Auszug aus dem Lexikon

Artikel

Wort	Grundorm	Definitheit	Numerus	Genus	Kasus	Übersetzung
alle	alle	+	pl	masc/fem/neut	nom/akk	all
der	die	+	pl	masc/fem/neut	gen	the
der	die	+	sg	fem	dat/gen	the
der	die	+	sg	masc	nom	the

Tabelle A.1.: Auszug aus der Artikel-Datei des Lexikons

Nomen

Wort	Grundorm	Numerus	Genus	Kasus	Übersetzung
Beschreibung	Beschreibung	sg	fem	gen	description
Beschreibung	Beschreibung	sg	fem	nom/dat/akk	description
Beschreibungen	Beschreibung	pl	fem	dat	descriptions
Beschreibungen	Beschreibung	pl	fem	nom/akk/gen	descriptions
Information	Information	sg	fem	gen	information
Information	Information	sg	fem	nom/dat/akk	information
System	System	sg	neut	nom/dat/akk	system

Tabelle A.2.: Auszug aus der Nomen-Datei des Lexikons

Modalverben

Wort	Grundorm	Numerus	Person	Finitheit	Übersetzung
muss	müssen	sg	3	+	shall
soll	sollen	sg	3	+	should
wird	werden	sg	3	+	will
müssen	müssen	pl	3	+	shall
sollen	sollen	pl	3	+	should
werden	werden	pl	3	+	will

Tabelle A.3.: Auszug aus der Modalverben-Datei des Lexikons

Präposition

Wort	Kasus
an	dat/akk
mittels	dat/gen
nach	dat
trotz	dat/gen

Tabelle A.4.: Auszug aus der Präpositionen-Datei des Lexikons

Verben

Grundform	Hilfswort	3. Person Singular	3. Person Plural	Partizip
aktivieren	haben	aktiviert	aktivieren	aktiviert
anbringen	sein	bringt an	bringen an	angebracht
passen	haben	passt	passen	gepasst
wechseln	haben	wechselt	wechseln	gewechselt

Verbpartikel	zu-Infinitiv	Subaktegorisierungsrahmen D
-	zu aktivieren	np:nom#np:akk/np:nom#np:akk#pp:bei_dat
+	anzubringen	np:nom#np:akk/np:nom#np:akk#pp:an_dat
-	zu passen	np:nom#pp:in_akk/np:nom/np:nom#np:dat
-	zu wechseln	np:nom#np:akk/np:nom/np:nom#pp:in_akk

Übersetzung	Subkategorisierungsrahmen E
activate	subj#np
attach	subj#np/subj#np#pp:to
fit	subj#np/subj#np#pp:with
change	subj/subj#np/subj#np#pp:to

Tabelle A.5.: Auszug aus der Verben-Datei des Lexikons

Erklärung

Hiermit versichere ich, diese Arbeit
selbständig verfasst und nur die
angegebenen Quellen benutzt zu haben.

(Nadine Siegmund)