STATISTICAL MACHINE TRANSLATION BEYOND CONTEXT-FREE GRAMMAR

INAUGURAL-DISSERTATION

zur Erlangung des Doktorgrades der Philosophie (Dr. phil.)
durch die Philosophische Fakultät der
Heinrich-Heine-Universität Düsseldorf

VORGELEGT VON

Miriam Käshammer aus Bühl (Baden)

BETREUERIN

Prof. Dr. Laura Kallmeyer

Düsseldorf, Mai 2018

D61

ABSTRACT

Statistical machine translation (SMT) has evolved from simple word-based models over feature-rich phrase-based approaches to tree-based methods. The latter employ synchronous grammars, usually some form of a Synchronous Context-Free Grammar (SCFG), to model hierarchical as well as translational relationships between language pairs.

In this thesis, an approach to tree-based SMT is explored which makes use of a grammar formalism beyond Context-Free Grammar (CFG). I define and implement the first SMT system based on Linear Context-Free Rewriting System (LCFRS), including training procedures and a cube-pruning decoder. At the same time, it is also the first hierarchical phrase-based system which allows for discontinuous phrases on the source as well as on the target side.

To that end, I define Synchronous Linear Context-Free Rewriting System (SLCFRS), a natural extension to SCFG. SLCFRS non-terminals may span more than one continuous block on each side of the bitext and can thus represent synchronous discontinuous constituents in a straightforward manner. In the domain of data-driven syntactic parsing, LCFRS is a well-studied formalism for modeling discontinuities while still being fairly efficient to handle. Experiments for translating from German to English demonstrate the feasibility of training and decoding with more expressive translation models such as SLCFRS and show a modest improvement over a context-free baseline.

The extension beyond context-freeness in the context of machine translation is motivated by a set of alignment configurations that are beyond the alignment capacity of current translation models based on SCFG. In quantitative and qualitative investigations, I show that an SCFG-based approach to translation modeling is not capable of deriving all alignments which occur in a wide range of manually aligned bilingual data sets, and that only very few of those configurations can be attributed to alignment errors. In order to not a priori exclude the corresponding translation options from the search space, a more expressive grammar formalism than SCFG is required in the context of tree-based translation.

ACKNOWLEDGMENTS

My deepest thanks go to my supervisor Laura Kallmeyer. It was your teaching at the University of Tübingen and the work in your project which introduced me to the world of parsing, grammar formalisms and beyond context-freeness. This dissertation would not have started and would not have finished without you. Thank you, especially for the support after I had already left and for the helpful comments on the pre-final version of this work. I would also like to thank Wiebke Petersen who kindly agreed to act as second examiner.

Wolfgang Maier has been my mentor in many academic and non-academic aspects. Thank you for cheering me up when I needed it, for the collaboration and scientific discussions, for reading my stuff, for motivating me to do things, for the Latex template, for making things more fun, for your advice, for your phone calls, and for beers. I am also grateful for the other nice, supportive and smart colleagues I had in Düsseldorf. Special thanks to Christian Wurm, Younes Samih, Simon Petitjean, Timm Lichte, Doris Gerland, Tim Seuchter, Thomas Gamerschlag and Christian Horn. From the many researchers I met at academic conferences around the world, I would like to especially thank Anders Søgaard, Gideon Maillette de Buy Wenniger and Peter Ljunglöf for the inspiring discussions. I am also thankful for the computational support and infrastructure provided by the ZIM and the IKM at the University of Düsseldorf.

A PhD is a long and winding road to take. I am extremely grateful to have had people around me who helped me stay sane. Thanks to Fabian, Elena, Nena, Markus, Katrin, Arne and Tim for being friends, not just flatmates, and making me feel home. Thanks to Sophie and Thomas for the distraction, moral support and survival cake. To my family, to Mama, Papa, Jule and Jonas, who provided endless love and unconditional support. Thank you for your belief in me and your patience. Lastly, to Lukas: you have been surviving that journey with me and made me move forward when I felt paralyzed. Thank you for staying by my side and making me happy.

Miriam Käshammer

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

1 INTRODUCTION

This thesis aims at transferring findings from formal grammar theory and parsing in the context of language modeling beyond Context-Free Grammar (CFG) to the field of statistical machine translation. In the following section, the background of these research topics will be outlined. A more detailed introduction to the individual concepts and topics will be presented later in chapter 2. Section 1.2 summarizes the contributions of this thesis. An overview of the following chapters is provided in section 1.3.

1.1 BACKGROUND

The goal of *machine translation* is to automatically translate an input text in a certain language into another language, as a human translator would perform the translation, e. g. translating the German example in (1a) to English; see (1b).

(1) a. Ich möchte die Präsidentschaft auch für ihre Arbeit loben.
    b. I also wish to praise the Presidency for its work.

The computer program performing the translation is called a *decoder*. In the case of *statistical machine translation (SMT)*, the translation task is formulated as a machine learning problem. The decoder operates on the basis of a model which defines a probability distribution $P(e|f)$ over all possible translations e for a given input f, and it is precisely the task of the decoder to find the most probable translation for f given the model. The most popular SMT models make use of a combination of different components (or features) which model different characteristics of the translation. Typically, at least a *translation model*, assigning probability to the correspondence between the source and the target sequence, and a *language model*, assessing the fluency of the target sequence, are used. The parameters of such statistical models are estimated from large amounts of (parallel) text. This work is exclusively concerned with translation modeling. The language model

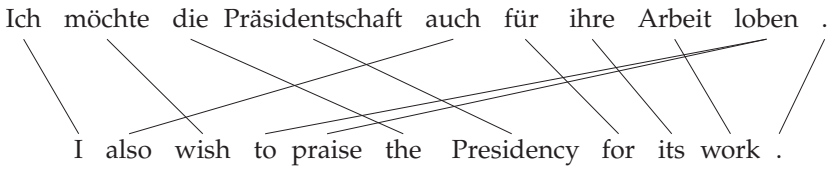


Figure 1: A word-based translation example. The links are word alignment links.

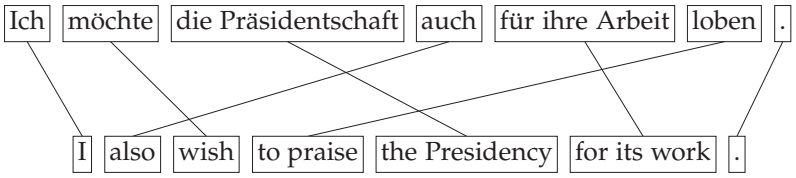


Figure 2: A phrase-based translation example. The links express correspondences between phrases.

will not be given further attention, except for the interaction of the language model and the translation model during decoding.

The first successful probabilistic translation models made use of translational correspondences on the word or phrase level. Examples are shown in figure 1 and 2 respectively. The translation process itself is performed sequentially from left to right. The past decades have however shown a trend towards more structural models, which allow to translate recursively. Such *tree-based* (also called *syntax-based*) translation models employ synchronous grammars that derive hierarchical structure on the source and the target side in parallel. Figure 3 depicts an example. Nodes which carry the same index have been derived synchronously. Translation is usually performed by parsing the input sentence with the monolingual source projection of the grammar using standard parsing algorithms, thereby inducing the derivation on the target side in parallel. The most prominent formalism is Synchronous Context-Free Grammar (SCFG), originally known as Syntax-Directed Transduction Grammar (SDTG), a generalization of CFG that generates pairs of strings instead of just strings. Figure 4 shows sample SCFG rewriting rules to derive the synchronous tree in figure 3. These rules do not only encode lexical and phrasal translational correspondence, as the word-based and phrase-based models do as well, such as *die Präsidentschaft* translates to *the Presidency* (rule r_4). They also

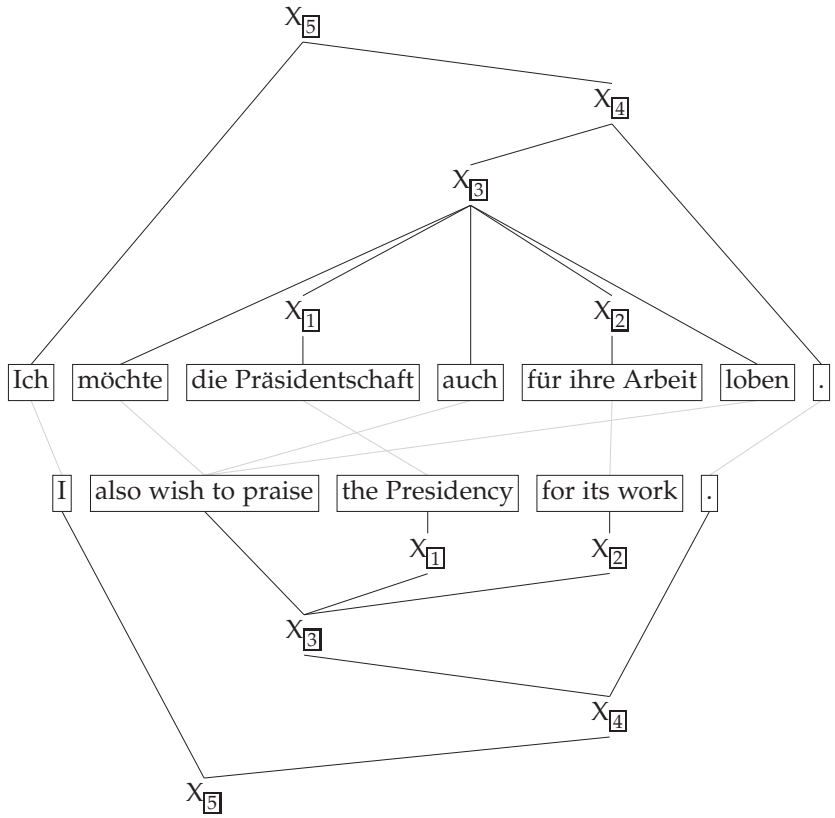


Figure 3: A tree-based translation example. The gray links represent phrasal alignments which are induced by the synchronous derivation.

contain reordering information, for instance that in German two (potentially hierarchical) phrases $X_{[1]}$ and $X_{[2]}$ intervene between the verbs *möchte* and *loben*, whereas in English they are placed after the verbal elements (rule r_3). Such information is particularly beneficial for language pairs with substantially different word orders.

There exist a variety of different flavors of SCFG translation models and also approaches for learning them from bitext. *Hierarchical phrase-based (Hiero)* translation grammars (Chiang, 2007) are syntactic only in a formal sense. They use a single non-terminal category, as in figure 3 and 4, and do not apply any notion of linguistics at all. Usually, Hiero grammars are extracted from automatically word-aligned text, applying a set of heuristic constraints to limit the type and number of extracted rules. Non-parametric approaches to infer such rules have also been proposed (e. g. Levenberg et al., 2012). Furthermore, a vari-

r_1 : $X \rightarrow \langle \text{Ich } X_{\boxed{1}} \text{ , I } X_{\boxed{1}} \rangle$

r_2 : $X \rightarrow \langle X_{\boxed{1}} \text{ . , } X_{\boxed{1}} \text{ .} \rangle$

r_3 : $X \rightarrow \langle \text{möchte } X_{\boxed{1}} \text{ auch } X_{\boxed{2}} \text{ loben , also wish to praise } X_{\boxed{1}} X_{\boxed{2}} \rangle$

r_4 : $X \rightarrow \langle \text{die Präsidentschaft , the Presidency} \rangle$

r_5 : $X \rightarrow \langle \text{für ihre Arbeit , for its work} \rangle$

Figure 4: Sample SCFG rules

ety of translation models that indeed make use of monolingual syntactic information and corresponding labels on the source and/or the target side have been worked out (e. g. Galley et al., 2004; Zollmann and Venugopal, 2006; Hoang and Koehn, 2010). As an example of the type of linguistic information which is integrated, consider figure 6.

The concept of a *word alignment* - a set of links between the source and target words of a sentence or paragraph that represent translational equivalence - is a central concept in SMT. Consider figure 1 as an example. As already mentioned, phrase-based and tree-based translation models are commonly learned on the basis of word alignments, or they are induced jointly. Unsupervised machine learning approaches to create word alignments are typically based on sequence models and the original word-based translation models of Brown et al. (1993). Synchronous grammars can also be viewed in the context of bitext parsing and alignment: When applying a synchronous rule, alignments between the terminals, i. e. the words of the parallel text, are induced. For instance, in figure 3 and 4, when applying rule r_1 to translate *Ich*, it becomes aligned to *I*.

However, the space of alignments that can be generated with SCFGs, and hence the space of translation options during decoding, is limited. Some alignment configurations, e. g. *inside-out (IO) alignments* (Wu, 1997) and certain discontinuous translation units with multiple gaps, are beyond the alignment capacity of SCFGs of rank 2.[1] *Cross-serial discontinuous translation units (CDTUs)* and *bonbon* configurations can be induced by neither SCFGs of any rank nor phrase-based translation systems (Søgaard and Kuhn, 2009; Wellington et al., 2006). It is thereby

1 The rank of an SCFG is the maximal number of non-terminals in the right-hand side (RHS) of the rules in the source or target projection. The grammar in figure 4 has rank 2. Grammars are often restricted to rank 2 for parsing efficiency.

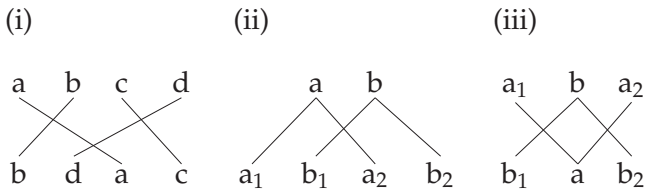


Figure 5: (i) IO alignment, (ii) CDTU, (iii) bonbon alignment

assumed that a *translation unit*, containing the transitive closure of some set of nodes of the bipartite alignment graph, represents minimal translational equivalence, and therefore that an adequate translation grammar should be able to generate each translation unit separately. Figure 5 schematically depicts some of the aforementioned alignment configurations.

Several studies addressing the empirical alignment capacity of various formalisms have found that those alignment configurations occur relatively frequently in manually aligned data. For instance, Wellington et al. (2006) report that 5% of their English-Chinese sentence pairs contain IO alignments, and Søgaard and Kuhn (2009) report that, on average, about every second sentence pair in their English-Spanish data contains a CDTU. Such findings have consequently put the empirical adequacy of SCFG translation models into question. A few efforts towards more powerful translation models in order to capture the described alignment configurations have appeared in the literature (Søgaard, 2008b; Galley and Manning, 2010).

In the parsing and formal linguistics community, a related development has taken place. For many years, CFG has been the grammar formalism of choice for describing the hierarchical structure of the syntax of languages in constituency-based frameworks (Chomsky, 1956). For illustration, consider the tree in figure 6 which shows a CFG analysis for an English sentence. Different parsing strategies were developed for CFG (e. g. Cocke and Schwartz, 1970; Earley, 1970; Rosenkrantz and Lewis, 1970). Data-driven probabilistic parsing has concentrated as well on approaches for CFG parsing (e. g. Charniak, 1997; Collins, 1999; Charniak, 2000; Klein and Manning, 2001; Petrov, 2009).

However, it is known since the 80's that CFG does not provide enough expressivity to model all natural language phenomena. In particular, CFG is only able to model *continuous* constituents, i. e. the words in the yield of a constituent always form a continuous sequence.

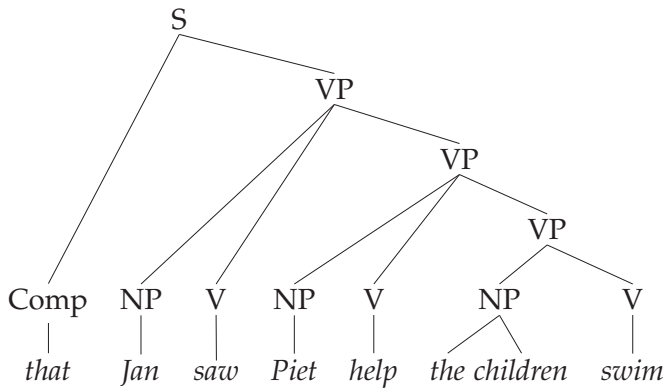


Figure 6: Constituency structure

Yet, the constituency analysis of certain phenomena, such as cross-serial dependencies in Dutch and Swiss-German, unbounded scrambling of constituents in German and, generally, non-local dependencies, requires *discontinuous* constituents. For illustration, consider example (2) (Bresnan et al., 1982), which is a literal Dutch translation of the English sentence in figure 6, and its analysis in figure 7.

(2) dat Jan Piet de kinderen zag helpen zwemmen
that Jan Piet the children saw help swim
'[...] that Jan saw Piet help the children swim'

Both in the English and the Dutch tree, each verb is grouped with its respective arguments under one verb phrase (VP). The discontinuous constituents in figure 7 can be easily spotted by paying attention to *crossing branches*. The two embedded lower VPs are discontinuous because their yields have a gap. They are interrupted by *zag helpen* and *zag* respectively. Such a tree cannot be derived with a CFG.

In order to characterize the amount of context-sensitivity which is required beyond context-freeness in order to describe natural language adequately, the notion of *mild context-sensitivity* was introduced (Joshi, 1985). Roughly, an adequate formalism includes at least CFGs, it is able to describe a limited amount of cross-serial dependencies, it is polynomially parsable and the lengths of the words of its languages grow linearly. Subsequently, several grammar formalisms which fall into the category of generating mildly context-sensitive languages and corresponding parsing algorithms were developed. The most influen-

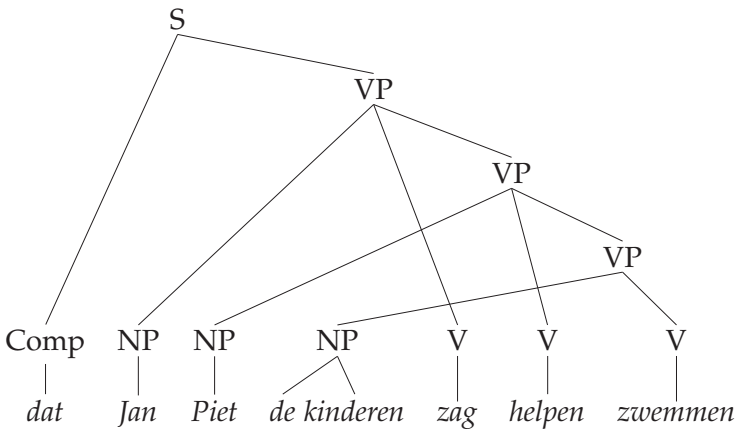


Figure 7: Constituency structure with discontinuous constituents. Adapted from Kallmeyer (2010, figure 1.2).

tial formalisms probably are, with increasing expressivity, Tree-Adjoining Grammar (TAG) (Joshi et al., 1975) and its variants, Linear Context-Free Rewriting System (LCFRS) (Vijay-Shanker et al., 1987) and Multiple Context-Free Grammar (MCFG) (Seki et al., 1991), and (restricted forms of) Range Concatenation Grammar (RCG) (Boullier, 2000).

LCFRS can be considered a generalization of CFG. Intuitively, the rules are still context-free, but instead of rewriting continuous strings, tuples of continuous strings are rewritten.[2] The length v of a tuple of strings which is spanned by a non-terminal is termed its fan-out. LCFRSs are parsable in polynomial time, but the degree depends directly on the fan-out v of the grammar. Several symbolic parsing algorithms for LCFRS and equivalent formalisms have been published (e. g. Seki et al., 1991; Burden and Ljunglöf, 2005; Kallmeyer and Maier, 2009). LCFRS derivations can be interpreted as parse trees with crossing branches. Figure 8 shows LCFRS rules which can be used to generate the tree in figure 7.[3] The third VP rule, for instance, explains that a discontinuous VP constituent may consist of an NP and a V

2 The original formulation of LCFRS is more general, in the sense that other objects, such as trees or graphs, can be rewritten as well. Within this work, we will only consider string-rewriting LCFRS. We furthermore use the notation of RCG.

3 In fact, the LCFRS rules have been generated from the tree in figure 7 using the approach in Maier and Søgaard (2008).

$$
\begin{aligned}
&\mathrm{S}(X_1X_2) && \rightarrow \mathrm{Comp}(X_1)\ \mathrm{VP}(X_2)\\
&\mathrm{VP}(X_1X_2X_3X_4) && \rightarrow \mathrm{NP}(X_1)\ \mathrm{VP}(X_2,X_4)\ \mathrm{V}(X_3)\\
&\mathrm{VP}(X_1X_2,X_3X_4) && \rightarrow \mathrm{NP}(X_1)\ \mathrm{VP}(X_2,X_4)\ \mathrm{V}(X_3)\\
&\mathrm{VP}(X_1,X_2) && \rightarrow \mathrm{NP}(X_1)\ \mathrm{V}(X_2)\\
&\mathrm{Comp}(\mathit{dat}) && \rightarrow \varepsilon\\
&\mathrm{NP}(\mathit{Jan}) && \rightarrow \varepsilon\\
&\mathrm{NP}(\mathit{Piet}) && \rightarrow \varepsilon\\
&\mathrm{NP}(\mathit{de\ kinderen}) && \rightarrow \varepsilon\\
&\mathrm{V}(\mathit{zag}) && \rightarrow \varepsilon\\
&\mathrm{V}(\mathit{helpen}) && \rightarrow \varepsilon\\
&\mathrm{V}(\mathit{zwemmen}) && \rightarrow \varepsilon
\end{aligned}
$$

Figure 8: Sample LCFRS rules

constituent and that there is a gap between the material coming from the NP and the material from the V.

In the last decade, LCFRS has attracted increased interest in the parsing community, in particular in the context of data-driven probabilistic parsing. Such parsers are trained on *treebanks*, which are text collections with a manually annotated syntactic analysis for each sentence. Some treebank annotation schemes, e. g. the one of the German Negra and Tiger treebanks (Skut et al., 1997; Brants et al., 2002), account for discontinuous constituents by directly annotating trees with crossing branches, as the one in figure 7. LCFRS has been established as an appropriate formalism for the modeling of discontinuous structures as they occur in treebanks (Maier and Lichte, 2011; Kuhlmann and Satta, 2009). In the Negra treebank, approximately 25% of all sentences have at least one discontinuous constituent (Maier and Lichte, 2011). Treebank trees with crossing branches can be directly interpreted as LCFRS derivation trees and wide-coverage probabilistic LCFRS grammars can be extracted in a natural way (Maier and Søgaard, 2008; Kallmeyer and Maier, 2013). Other annotation mechanisms to account for discontinuity in treebanks, such as the trace and co-indexation annotation of the English Penn Treebank (Marcus et al., 1994), which otherwise annotates context-free trees, can be transformed to trees with crossing branches first (Evang and Kallmeyer, 2011). It has been shown that data-driven probabilistic parsing with LCFRS is feasible

and gives acceptable results (Maier, 2010; Evang and Kallmeyer, 2011; van Cranenburgh, 2012; Maier et al., 2012; Kallmeyer and Maier, 2013).

For data-driven probabilistic CFG parsing, it is common practice to simply discard the additional information about the non-local dependencies in Penn Treebank style annotations, and to convert trees with crossing branches to CFG trees using a lossy conversion algorithm. This means that the correct syntactic dependencies get lost and that such CFG parsers are not able to produce trees which contain the same level of information as represented in the original constituency structures of the treebanks.[4] The direct parsing of discontinuous constituents with LCFRS remedies this deficiency.

1.2 CONTRIBUTIONS

1.2.1 *What this Thesis is About*

The focus of this thesis is translation modeling, in particular translation modeling beyond SCFG, and its application in an SMT system. It seems timely to transfer recent advances of the parsing community concerning parsing with mildly context-sensitive grammar formalisms to tree-based SMT. This step is motivated by the limitations which SCFG imposes on the space of alignments and the resulting questionable adequacy of SCFG translation models. Our major contributions are the following:

LCFRS FOR TRANSLATION MODELING We propose LCFRS for the modeling of translation in order to be able to generate the alignment configurations which SCFG-based and phrase-based translation models cannot generate, and, to the best of our knowledge, we are the first to do so. By using a grammar formalism which is more powerful than SCFG for that purpose, we follow the line of work of Søgaard (2008b). LCFRS, the primary representative of the class of mildly context-sensitive grammar formalisms, is a promising candidate for translation modeling. As it is a direct generalization of CFG, many concepts and techniques can be adapted in a straight-forward fashion. Further-

4 Approaches which enhance a probabilistic CFG parser with pre- and/or post-processing techniques and which use machine learning to recover discontinuous constituents have been published as well (Johnson, 2002; Dienes and Dubey, 2003; Levy and Manning, 2004).

more, LCFRS has proven to be successful in the context of data-driven probabilistic parsing. We consequently introduce a novel synchronous grammar formalism, the synchronous version of LCFRS, namely *Synchronous Linear Context-Free Rewriting System (SLCFRS)*. We establish SLCFRS as a formalism for translation modeling beyond CFG and provide a deduction system for bitext parsing.

EXTENSIVE INVESTIGATION WITH RESPECT TO ALIGNMENT CONFIGURATIONS BEYOND SCFG In order to thoroughly motivate the move from SCFG to SLCFRS for translation modeling, we contribute an extensive investigation of the empirical alignment capacity of SCFG and the alignment configurations beyond. To that end, we investigate manual alignments of 16 language pairs in 19 data sets using a bottom-up hierarchical aligner based on SLCFRS, for which we contribute the deduction system as well as the actual implementation. We confirm previous similar experiments of a smaller scale in that SCFG of rank 2 is not expressive enough to cover all alignment configurations which occur in natural data. Moreover, we extend previous experiments by investigating the amount of discontinuity which is required in order to derive the manual alignments. In addition, we conduct a manual investigation of the alignment configurations beyond SCFG for four data sets in order to preempt arguments towards these alignment configurations being annotation errors. We also contribute the discovery of a class of alignment configurations beyond SCFG of rank 2 which have not been reported in the literature before. We call them inside-out discontinuous translation units (IO-DTUs).

THE FIRST SMT SYSTEM BASED ON SLCFRS Finally, we contribute the first SMT system which is based on weighted SLCFRS, that is the first hierarchical phrase-based machine translation system with discontinuous phrases on the source and the target side. Our model includes features that are specific to translation models which allow for discontinuities such as the source and target gap degree of the rules. Decoding is performed with an extended LCFRS bottom-up chart parser which generates possibly discontinuous partial translations. The decoding process also includes language model scoring and applies cube pruning to narrow down the translation search space. We also provide the first evaluation of this SMT system and show

modest, but significant improvements over an SCFG-based baseline for German-to-English translation.

1.2.2 *What this Thesis is not About*

Neural Machine Translation

During writing up this thesis, a new machine translation paradigm emerged. While *traditional*[5] SMT approaches model the distribution $P(e|f)$ with a log-linear model that combines several feature functions (see section 2.2.1, in particular equation (2.7)), *neural machine translation (NMT)* uses a deep artificial neural network for this task (Sutskever et al., 2014; Cho et al., 2014; Bahdanau et al., 2015). Currently, it is state of the art to use a recurrent neural network encoder-decoder setup with a bidirectional encoder, an additional attention mechanism and a softmax layer as the output layer which provides a probability distribution over the target vocabulary. To find the best translation, beam search with a small beam size is performed over the output states. This approach has been shown to outperform phrase-based and tree-based SMT approaches for many language pairs by a wide margin (e. g. Junczys-Dowmunt et al., 2016).

We would like to emphasize that the main points of this thesis were published in the years 2013 to 2015, i. e. before NMT emerged and became accepted as a disruptive technology and state of the art. Thus, even though this thesis is published at a time where deep learning and NMT dominate the field, it is still located in *the old world*, and deep artificial neural networks will not be a topic anymore after this section. If one is not merely interested in achieving high translation quality, but also in the process of translation generation and in the structures and alignments used therein, then the work at hand will nevertheless provide insights.

Furthermore, there is the chance that the two approaches to machine translation will eventually benefit from each other. Finally, we see opportunities to position language and translation modeling using grammar formalisms beyond CFG in the context of deep learning, e. g. by using a tree-structured network on the encoder side for NMT. Working out those ideas will be left for future work.

5 What is called *traditional* in this section, is called *state of the art* elsewhere in this thesis.

Dependency Structure

Besides constituency-based descriptions of the hierarchical structure of language, *dependency-based* frameworks provide another approach to syntactic description, in which the words themselves are nodes (of a tree) and the edges between them represent relations, so-called *dependencies*. Dependency and constituent structures are closely related, as CFG is equivalent to the class of *projective* dependency structures (Gaifman, 1965). The equivalent of discontinuous constituents in the dependency-based framework are *non-projective dependencies*.

Different parsing strategies for data-driven dependency parsing have been presented in the literature, including parsers which also generate non-projective dependencies. A few translation models have been developed specifically for dependency structures (Quirk et al., 2005; Menezes and Quirk, 2007, 2008; Shen et al., 2010). With the exception of Carreras and Collins (2009) who use a variant of TAG as a proxy to model dependencies, it seems that none of them deal with non-projective dependencies. In this thesis, we focus on constituency-based approaches to tree-based SMT. In favor of a clear scope, we will not further discuss dependency-based models.

1.3 OVERVIEW

Chapter 2 represents the foundation of all later chapters. We present basic definitions and notations, terminology and concepts. The chapter is divided into two parts. The first one concentrates on the monolingual modeling of language and introduces grammar formalisms and parsing. The second part is dedicated to SMT. It explains the general framework and largely concentrates on tree-based translation models.

Chapter 3 provides the motivation for using translation models beyond CFG. Besides presenting the alignment configurations for which phrase-based and SCFG-based translation models are not powerful enough, we investigate to which extend manual word alignments are affected. We furthermore present a qualitative analysis of affected alignment configurations.

In chapter 4, we show that discontinuous constituents are required in order to generate the alignment configurations beyond SCFG and introduce the novel synchronous grammar formalism SLCFRS to that end. We present a CYK-style bitext parsing algorithm for SLCFRS and

extend the investigation of chapter 3 in order to quantify the number of gaps which are required to generate the manual word alignments at hand.

Chapter 5 is dedicated to the SLCFRS-based SMT system, i. e. to hierarchical machine translation with discontinuous phrases on both the source and the target side. We present the model itself, grammar learning, and decoding including integration of the language model.

In chapter 6, the hierarchical machine translation approach with discontinuous phrases is evaluated. Besides automatic metrics on a German-to-English translation task, we present the results of a manual evaluation and individual translation examples.

Chapter 7 concludes this thesis. We furthermore provide perspectives for future work.

Related work is presented throughout this thesis in the corresponding relevant context.

2

FOUNDATIONAL MATTERS

The work at hand combines research and findings from formal language theory and parsing with the world of statistical machine translation (SMT). This chapter introduces the notions and notations, concepts and approaches which the later chapters build on. Section 2.1 covers grammar formalisms and syntactic parsing, while section 2.2 focuses on SMT.

2.1 GRAMMAR FORMALISMS AND SYNTACTIC PARSING

Parsing is one of the core tasks of computational linguistics. It refers to the automatic syntactic analysis of a natural language sentence. In this thesis, we focus on constituency-based[1] syntactic structures, which describe the hierarchical relationship between constituent phrases (*constituents*) of a sentence. A constituent comprises all words which form a (linguistically) meaningful unit. Its label usually denotes a linguistic type of an underlying linguistic theory, e. g. VP for *verb phrase*, S for *sentence* or NP for *noun phrase*.[2] A *constituency structure* is a hierarchical structure of constituents.

Grammar formalisms are used as mathematical devices to precisely formulate linguistic and, in particular, syntactic theories about natural languages. They allow to define *grammars*, which basically are finite sets of rules. The set of strings that can be generated with a grammar is called the *language* of the grammar. A *derivation* is a sequence of rule applications whose results can be visualized as a *parse tree*. Constituency grammars, i. e. grammars which describe constituency structures, are also known under the name of *phrase structure grammar*, as they were originally called by N. Chomsky.

In this section, first, basic concepts which will be used throughout this thesis are defined. Those are trees for syntactic modeling and the grammar formalisms Context-Free Grammar (CFG) and Linear Con-

1 As opposed to dependency-based descriptions, see also p. 12.

2 Arguing for or against particular linguistic theories is beyond the scope of this thesis.

text-Free Rewriting System (LCFRS). Then the concept of parsing as deduction and the relation between parsing and hypergraphs is introduced. Those concepts also carry over to tree-based SMT. We will then move on to the modeling of natural language syntax, motivate why CFG, the prevalent formalism in natural language processing (NLP), is not suitable, and provide an overview over grammar formalisms beyond CFG. Finally, the direct parsing of discontinuous constituents with LCFRS will be described.

This section can only provide selective insights into parsing and syntactic description of natural language. For a detailed overview of formal language theory, see Hopcroft and Ullman (1979), for parsing in general, see Grune and Jacobs (2008), and for parsing beyond CFG, see Kallmeyer (2010).

2.1.1 *Basic Definitions*

This section provides definitions for basic mathematical concepts in the context of syntactic description and the grammar formalisms CFG and LCFRS.

Trees

Information about syntax is usually encoded by means of trees. Trees are a particular kind of graphs. The presentation here loosely follows Kallmeyer (2010, sec. 1.5.4).

DEFINITION 2.1 (Directed Graph).

1. A *directed graph* is a pair $\mathcal{G} = \langle V, E \rangle$ where V is a finite set of vertices (or nodes) and $E \subseteq V \times V$ is a set of edges.

2. E^+ is the transitive closure of E, and E^* is the reflexive transitive closure of E.

3. For every $v \in V$, the *out-degree* $f_{out}(v)$ is defined as

   $$|\{u \mid \langle v, u \rangle \in E\}|$$

   and the *in-degree* $f_{in}(v)$ is defined as

   $$|\{u \mid \langle u, v \rangle \in E\}|\,.$$

4. $\mathcal{G}$ is *acyclic* iff it does not contain cycles, i. e. there is no $v \in V$ such that $\langle v, v \rangle \in E^+$. □

DEFINITION 2.2 (Labeling (Graph)). Let $\mathcal{G} = \langle V, E \rangle$ be a directed graph. $\mathcal{G}$ is *(node) labeled* iff it has a node labeling over an alphabet L. A *(node) labeling* over an alphabet L is a function $l : V \to L$. □

DEFINITION 2.3 (Tree). A *tree* is a triple $\mathcal{D} = \langle V, E, r \rangle$ such that

1. $\langle V, E \rangle$ is a directed, acyclic graph,
2. r is a distinguished *root* node with $f_{in}(r) = 0$,
3. every vertex $v \in V$ is accessible from r, i. e. $\langle r, v \rangle \in E^*$, and
4. for all $v \in V \setminus \{r\}$, $f_{in}(v) = 1$. □

DEFINITION 2.4 (Properties of trees). Let $\mathcal{D} = \langle V, E, r \rangle$ be a tree.

1. $\mathcal{D}$ is (node) labeled iff $\langle V, E \rangle$ is (node) labeled.
2. For all vertices $v \in V$, if $f_{out}(v) = 0$, then v is a *leaf*, otherwise v is an *internal node*.
3. For all vertices $v_1, v_2 \in V$, v_1 *directly dominates* v_2 iff $\langle v_1, v_2 \rangle \in E$, and v_1 *dominates* v_2 iff $\langle v_1, v_2 \rangle \in E^*$. If v_1 directly dominates v_2, then v_1 is referred to as the *mother* or *parent node* of v_2, and v_2 is called a *daughter* or *child node* of v_1.
4. $\mathcal{D}$ is *ordered* if it has a linear precedence relation relation $\prec \subseteq V \times V$ which is antisymmetric, irreflexive and transitive. □

Syntactic constituency structures are ordered labeled trees. The ordering is defined on the basis of the natural ordering of the leaf nodes. They correspond to the words of the natural language sentence and are labeled with integers which denote the position of the respective word. The internal nodes are ordered by the label of the respective leftmost leaf that they dominate.

Our definitions of syntactic structures and related notions like yield, yield block set, gap and block degree and well-nestedness closely follow the presentation in Maier (2013, sec. 2.1.2, sec. 5.2.1).

DEFINITION 2.5 (Syntactic constituency structure). Let $w = w_1 \dots w_n$ with $n \in \mathbb{N}$ be a string, and let $\mathcal{D} = \langle V, E, r \rangle$ be an ordered tree with a node labeling l. $\mathcal{D}$ is a *syntactic constituency structure* for w if $L = \{1, \dots, n\} \cup N$, with N being a set of syntactic category labels disjoint from $\{1, \dots, n\}$, and l is such that

1. for all $1 \leq i \leq n$, there is exactly one $v_i \in V$ with $f_{out} = 0$ (i. e. a leaf) and $l(v_i) = i$, and
2. for all $v \in V$ with $f_{out} > 0$, $l(v) \in N$.

Furthermore, $\prec$ is such that

1. for all $v_1, v_2 \in V$ with $l(v_1), l(v_2) \in \mathbb{N}$ and $l(v_1) < l(v_2)$: $v_1 \prec v_2$, and
2. for all $v_1, v_2 \in V$ with $l(v_1), l(v_2) \in N$: $v_1 \prec v_2$ iff $\min(\{l(v) \mid v \in V \text{ and } l(v) \in \mathbb{N} \text{ and } \langle v_1, v \rangle \in E^+\}) < \min(\{l(v) \mid v \in V \text{ and } l(v) \in \mathbb{N} \text{ and } \langle v_2, v \rangle \in E^+\})$. □

When depicting the graphical representation of a syntactic structure, we will usually represent internal nodes with their label $l(v) \in N$ and leaves with the word w_i of the sentence that corresponds to their label $l(v) = i$. See ex. 2.6. Internal nodes of constituency structures are also referred to as *constituents*.

EXAMPLE 2.6 (Constituency structure). We consider again (2), repeated here as (3), and figure 9, which shows the graphical representation of the syntactic constituency structure of the sentence.

(3) dat Jan Piet de kinderen zag helpen zwemmen
that Jan Piet the children saw help swim
'[...] that Jan saw Piet help the children swim'

We would like to point out the following:

- By convention, trees are drawn such that all edges point downwards. We therefore omit the arrowhead.
- Each leaf is labeled with its index as well as the word which corresponds to the index. In other graphical representations of constituency structures throughout this thesis, we usually omit the indices. See figure 7 for example.

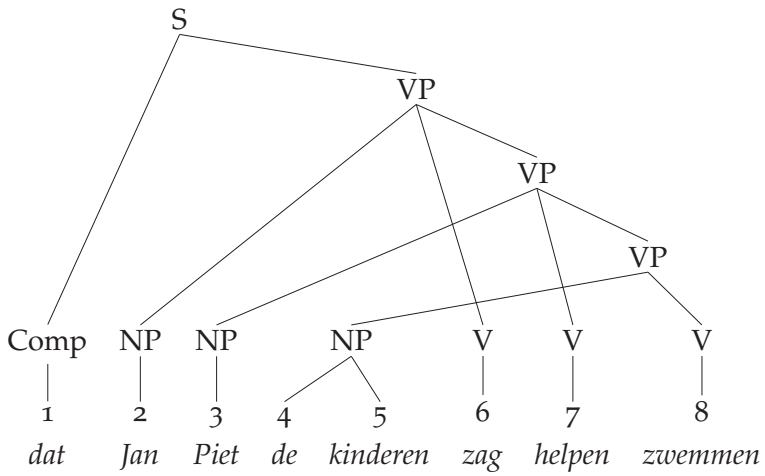


Figure 9: Syntactic structure for ex. 2.6

- The leaf nodes are depicted according to their ordering. However, we refrain here and throughout the thesis to depict the internal nodes of trees according to the ordering for the sake of readability. As an example consider the children $\mathrm{NP}_2, \mathrm{V}_6$ and VP_3 of the highest VP node. The indices indicate the positions according to which the nodes are ordered (i. e. their leftmost terminal child). The ordering thus is $\mathrm{NP}_2 \prec \mathrm{VP}_3 \prec \mathrm{V}_6$, even though this is not depicted in figure 9. □

We will later define the yield of a node in a syntactic structure as a set of natural numbers. Sets of natural numbers will also occur elsewhere in this thesis. In such sets, *blocks* are sequences of continuous integers, they have an ordering, and in between the blocks, there are *gaps*.

Definition 2.7 (Block set). Let $S \subset \mathbb{N}$. The *block set* Ω_S is a partition of S given by the equivalence relation D on S. For all $i, j \in S$, $i \sim_D j$ iff

1. $i < j$ and $i + 1 \in S$ and $i + 1 \sim_D j$, or
2. $i > j$ and $i - 1 \in S$ and $i - 1 \sim_D j$, or
3. $i = j$.

We call any $o \in \Omega_S$ a *block* of S. □

Definition 2.8 (Block set ordering). Let $S \subset \mathbb{N}$. The following holds for the block set Ω_S.

1. Ω_S is ordered according to a strict total order $\prec$ called *block ordering* such that for all $o_1, o_2 \in \Omega_S$, $o_1 \prec o_2$ iff $min(o_1) < min(o_2)$.
2. Any $o \in \Omega_S$ is the *ith block* of S iff $|\{o' \in \Omega_S \mid o' \prec o\}| = i - 1$. □

Definition 2.9 (Gap). Let $S \subset \mathbb{N}$. A gap in S is a pair (i_1, i_2) with $i_1, i_2 \in S$ and $i_1 + 1 < i_2$ and for all i in $i_1 < i < i_2$ it holds that $i \notin S$. The size of the gap (i_1, i_2) is $i_2 - i_1 - 1$. □

The *yield* of a node in a syntactic tree are the labels of all leaves which are dominated by this node.

Definition 2.10 (Yield). Let $\mathcal{D} = \langle V, E, r \rangle$ be a syntactic constituency structure. The *yield* function $\pi : V \to \mathcal{P}(\mathbb{N})$ is defined such that for all $v \in V$, $\pi(v) = \{i \in \mathbb{N} \mid \text{there is a } v' \in V \text{ with } \langle v, v' \rangle \in E^* \text{ and } l(v') = i\}$. □

Definition 2.11 (Yield block set). Let $\mathcal{D} = \langle V, E, r \rangle$ be a syntactic constituency structure. For all $v \in V$, the *yield block set* Ω_v of v is defined as the block set $\Omega_{\pi(v)}$ of the yield of v. □

All other notions related to block sets naturally extend to yield block sets. We call a gap in $\pi(v)$ a gap of v.

To quantify the amount of discontinuity in a syntactic structure, *gap degree* and *block degree* have been defined (Kuhlmann, 2007; Maier and Lichte, 2011).

Definition 2.12 (Gap degree). Let $\mathcal{D} = \langle V, E, r \rangle$ be a syntactic constituency structure. For all $v \in V$, the *gap degree* of v is the number of gaps in $\pi(v)$. The gap degree of $\mathcal{D}$ is the maximal gap degree of any of its nodes. □

Definition 2.13 (Block degree). Let $\mathcal{D} = \langle V, E, r \rangle$ be a syntactic constituency structure. For all $v \in V$, the *block degree* is a function $dim : V \to \mathbb{N}$. dim yields the cardinality of the block set of v. □

It is obvious that the gap degree of a node v equals $dim(v) - 1$. Furthermore, a node v with at least one gap in its yield, or equally $dim(v) > 1$, is *discontinuous*, and a constituency structure with a gap degree > 0 is *discontinuous*.

Another characteristic of syntactic structures is *well-nestedness* as opposed to *ill-nestedness* (Maier and Lichte, 2011). Intuitively, two nodes of a syntactic structure that do not dominate each other are well-nested if their yields do not interleave.

DEFINITION 2.14 (Well-Nestedness). Let $\mathcal{D} = \langle V, E, r \rangle$ be a syntactic constituency structure.

1. A pair of nodes $v_1, v_2 \in V$ with $\pi(v_1) \cap \pi(v_2) = \emptyset$ is *well-nested* iff there are no $i_1, i_2 \in \pi(v_1)$ and $j_1, j_2 \in \pi(v_2)$ such that $i_1 < j_1 < i_2 < j_2$.
2. $\mathcal{D}$ is *well-nested* iff all pairs of nodes in V with disjoints yields are well-nested. □

We define the *depth* of a gap of v as the length of the upward path from v to the lowest node which dominates v and fills its gap.

DEFINITION 2.15 (Gap filler, gap depth). Let $\mathcal{D} = \langle V, E, r \rangle$ be a syntactic constituency structure and (i_1, i_2) a gap of some $v_0 \in V$. A node v_1 is called a *gap filler* of (i_1, i_2) iff for all $i_1 < i < i_2$ it holds that $i \in \pi(v_1)$. The *depth* of (i_1, i_2) is defined as $n \in \mathbb{N}$ being the length of the path $(v_n, \ldots, v_0)$, $v_i \in V$ for all $0 \leq i \leq n$, for which the following holds:

1. $\langle v_j, v_{j-1} \rangle \in E$ for all $0 < j \leq n$ (i. e. $\langle v_n, v_0 \rangle \in E^+$),
2. v_n is a gap filler of (i_1, i_2),
3. no v_i for all $0 < i < n$ is a gap filler of (i_1, i_2). □

EXAMPLE 2.16 (Yield, block set, block degree, gap degree, gap depth, well-nestedness). Let us consider again the syntactic structure $\mathcal{D}$ in figure 9, in particular the node labeled S. Its yield is $\{1, 2, 3, 4, 5, 6, 7, 8\}$, its yield block set is $\{\{1, 2, 3, 4, 5, 6, 7, 8\}\}$ and its block degree is $|\{\{1, 2, 3, 4, 5, 6, 7, 8\}\}| = 1$. Its yield does not have any gaps.

Now consider the central VP node, the grandchild of the node labeled S. Its yield is $\{3, 4, 5, 7, 8\}$. Its yield block set is $\{\{3, 4, 5\}, \{7, 8\}\}$ where $\{3, 4, 5\}$ is the first block and $\{7, 8\}$ is the second block. Its yield has one gap, which is $(5, 7)$. The depth of this gap is 1. The block degree of this node is $|\{\{3, 4, 5\}, \{7, 8\}\}| = 2$ and its gap degree is 1.

The lower VP node also has gap degree 1. The depth of the gap $(5, 8)$ is 2. All other nodes in $\mathcal{D}$ have gap degree 0. Accordingly, $\mathcal{D}$ has gap degree 1. $\mathcal{D}$ is furthermore well-nested. □

Context-Free Grammar (CFG)

CFG is the prevalent formalism in natural language processing, and its synchronous variant and corresponding parsing techniques are also

applied in SMT. We loosely follow Hopcroft and Ullman (1979) in the presentation of the definition of CFG and related notions.

DEFINITION 2.17 (Context-Free Grammar). A *Context-Free Grammar (CFG)* is a tuple $G = (N, T, P, S)$ where

1. N and T are finite, disjoint sets of non-terminal and terminal symbols,
2. $S \in N$ is a distinguished *start symbol*, and
3. P is a finite set of *(rewriting) rules* of the form $A \to \alpha$ with $A \in N$ and $\alpha \in (N \cup T)^*$. □

Generally, in rewriting rules, the part left of the rewriting symbol $\to$ is called its *left-hand side (LHS)*, and the part right of the rewriting symbol is called its *right-hand side (RHS)*, i. e. in a CFG rule $A \to \alpha$, A is the LHS and α constitutes the RHS. Furthermore, we can call a rule $A \to \alpha \in P$ a *terminal rule* if its RHS consists of terminal symbols only, i. e. $\alpha \in T^*$, and we can call it a *mixed rule* if its RHS consists of at least one terminal and one non-terminal symbol.

DEFINITION 2.18 (Rank (CFG)). Let $G = (N, T, P, S)$ be a CFG. The *rank* of a rule $A \to \alpha \in P$ is the number of non-terminals occurring in α. The rank of G is the maximal rank of any of its rules $r \in P$. G is called a u-CFG if it has rank u. □

In order to formally define the language of a CFG, we first provide the *derives* relation.

DEFINITION 2.19 (Derivation (CFG)). Let $G = (N, T, P, S)$ be a CFG.

1. $\Rightarrow_G \subseteq (N \cup T)^+ \times (N \cup T)^*$ is a relation called *derives* between two strings. It is defined as follows: $\beta \Rightarrow_G \beta'$ iff there is a $A \to \alpha \in P$ and $\beta = \gamma A \gamma'$ and $\beta' = \gamma \alpha \gamma'$ with $\gamma, \gamma' \in (N \cup T)^*$. If G is given by the context, we can use $\Rightarrow$ instead of $\Rightarrow_G$. To make the applied rule explicit, we can use $\Rightarrow_G^{A \to \alpha}$.
2. $\overset{*}{\Rightarrow}_G$ is the reflexive transitive closure of $\Rightarrow_G$.
3. Let $\beta_1, \ldots, \beta_m \in (N \cup T)^*, m \in \mathbb{N}$. $\beta_1 \Rightarrow_G \ldots \Rightarrow_G \beta_m$ is a *derivation* of length m. □

DEFINITION 2.20 (Language (CFG)). Let $G = (N, T, P, S)$ be a CFG. The *(string) language* of G is $\mathcal{L}(G) = \{w \mid w \in T^* \text{ and } S \overset{*}{\Rightarrow}_G w\}$. □

Derivations can also be represented as trees, called *derivation* or *parse trees*. Recall the definition of a tree (def. 2.3, p. 17) and its properties (def. 2.4, p. 17).

DEFINITION 2.21 (Derivation tree (CFG)). Let $G = (N, T, P, S)$ be a CFG. A labeled ordered tree $\mathcal{D} = \langle V, E, r \rangle$ is a *derivation tree* for G iff the following conditions hold.

1. The order $\prec$ is such that for all $v_1, v_2 \in V$ with $\langle v_1, v_2 \rangle \notin E^*$ and $\langle v_2, v_1 \rangle \notin E^*$,
   a) either $v_1 \prec v_2$ or $v_2 \prec v_1$, and
   b) if there is either a $\langle v_3, v_1 \rangle \in E$ with $v_3 \prec v_2$ or a $\langle v_4, v_2 \rangle \in E$ with $v_1 \prec v_4$, then $v_1 \prec v_2$, and
   c) nothing else is in $\prec$.
2. The node labeling l over $N \cup T \cup \{\varepsilon\}$ is such that
   a) for all $v \in V$, if $f_{out}(v) = 0$ (i. e. v is a *leaf*), then $l(v) \in T \cup \{\varepsilon\}$, otherwise $l(v) \in N$, and
   b) for all $v, v_1, \ldots, v_n \in V$, $n \in \mathbb{N}$ where $v_1, \ldots, v_n$ are the only and all children of v and $v_j \prec v_{j+1}$ for $1 \leq j < n$, there exists a rule $l(v) \to l(v_1) \ldots l(v_n) \in P$.

$\mathcal{D}$ is called a *partial derivation tree* if $l(r) \neq S$. □

Reading off the labels of the leaves according to the ordering of the tree provides the *yield* of the derivation tree (cf. def. 2.10, p. 20). If β is the yield of some derivation tree for $G = (N, T, P, S)$, then $S \overset{*}{\Rightarrow}_G \beta$.

Normal forms impose restrictions on the form of the rules of the grammar. The availability of a grammar in normal form can be useful for parsing.

DEFINITION 2.22 (Chomsky Normal Form). Let $G = (N, T, P, S)$ be a CFG with $\varepsilon \notin \mathcal{L}(G)$. G is in *Chomsky Normal Form (CNF)* iff all rules $r \in P$ have either the form $A \to BC$ or $A \to a$ with $A, B, C \in N$ and $a \in T$. □

DEFINITION 2.23 (ε-free CFG). Let $G = (N, T, P, S)$ be a CFG. A rule $A \to \alpha \in P$ is called an ε-rule if $\alpha = \varepsilon$. G is ε-free if either P does not contain any ε-rules or the only ε-rule it contains is the rule $S \to \varepsilon$ and S does not appear in any RHS of the other rules in P. □

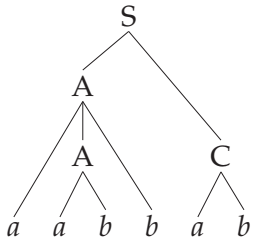


Figure 10: CFG derivation tree

For each context-free language L that does not generate ε, there is a CFG G in CNF with $L = \mathcal{L}(G)$, and every CFG G can be transformed into a weakly equivalent ε-free CFG G' (Hopcroft and Ullman, 1979).

EXAMPLE 2.24 (Context-Free Grammar). Let $G = (\{\mathrm{S}, \mathrm{A}, \mathrm{C}\}, \{a, b\}, P, \mathrm{S})$ be a CFG with $P = \{\mathrm{S} \to \mathrm{AC}, \mathrm{A} \to a\mathrm{A}b, \mathrm{A} \to ab, \mathrm{C} \to ab\}$. The language of G is $\mathcal{L}(G) = \{a^n b^n ab \mid n \in \mathbb{N}\}$. G is not in CNF, but it is ε-free. Figure 10 shows the derivation tree for *aabbab*. □

In a *weighted grammar*, the rules of the grammar are equipped with weights, which extend to weights of derivations of that grammar. A probabilistic grammar is the special case in which the weights define a probability distributions over derivations or strings. Weights (or probabilities) are useful for disambiguation, i. e., for determining which derivation of a string w is the most probable one, or, generally, for ranking parse trees.

Weighted and probabilistic CFG have been defined and discussed in various places in the literature, e. g. in Booth and Thomson (1973), in Wetherell (1980) and in Smith and Johnson (2007).

DEFINITION 2.25 (Weighted Context-Free Grammar). A *weighted Context-Free Grammar (CFG)* is a tuple $G = (N, T, P, S, \omega)$ where (N, T, P, S) is a CFG and $\omega : P \to \mathbb{R}_{\geq 0}$ is a weight function which maps from rules to positive real numbers.[3] □

DEFINITION 2.26 (Weight of a derivation (Weighted CFG)). Let $G = (N, T, P, S, \omega)$ be a weighted CFG, and let $\beta, \beta' \in (N \cup T)^*$.

3 Assigning a rule r a weight of zero is technically possible, but means that r is excluded from P. This holds for all definitions of weighted grammars in this work.

1. Let $A \to \alpha \in P$. The weight of one derivation step $\beta \Rightarrow_G^{A \to \alpha} \beta'$ is defined as

$$\omega(\beta \Rightarrow_G^{A \to \alpha} \beta') = \omega(A \to \alpha)$$

2. Let $A_1 \to \alpha_1, \ldots, A_m \to \alpha_m \in P$, $m \in \mathbb{N}$. The weight of a derivation $\beta \Rightarrow_G^{A_1 \to \alpha_1} \ldots \Rightarrow_G^{A_m \to \alpha_m} \beta'$ is defined as

$$\omega(\beta \Rightarrow_G^{A_1 \to \alpha_1} \ldots \Rightarrow_G^{A_m \to \alpha_m} \beta') = \prod_{i=1}^{m} \omega(A_i \to \alpha_i)$$

□

Definition 2.27 (Probabilistic Context-Free Grammar). A *probabilistic* Context-Free Grammar (CFG) is a weighted CFG $G = (N, T, P, S, \omega)$ with $\omega : P \to [0, 1]$ being such that for each $A \in N$ the following holds:

$$\sum_{A \to \alpha \in P} \omega(A \to \alpha) = 1$$

□

In this case, $\omega(A \to \alpha)$ expresses the conditional probability $p(A \to \alpha | A)$.

Example 2.28 (Weighted CFG). Let $G = (N, T, P, S, \omega)$ be a weighted CFG with (N, T, P, S) as given in ex. 2.24 and ω as follows:

i	r_i	$\omega(r_i)$
1	S $\to$ AC	1.0
2	A $\to$ aAb	0.4
3	A $\to$ ab	0.6
4	C $\to$ ab	1.0

G is not only a weighted, but a probabilistic CFG. As an example, consider the derivation of *aabbab* and its probability:

$$\begin{aligned}\omega(\text{S} \Rightarrow \text{AC} \Rightarrow \text{A}ab \Rightarrow a\text{A}bab \Rightarrow aabbab) &= 1.0 \cdot 1.0 \cdot 0.4 \cdot 0.6 \\ &= 0.24\end{aligned}$$

□

Linear Context-Free Rewriting System (LCFRS)

We now define LCFRS (Vijay-Shanker et al., 1987; Weir, 1988) and related notions. Only the string-rewriting variant of LCFRS will be considered throughout this work. We mostly follow the definitions given in Maier (2013), and use the syntax of Simple Range Concatenation Grammar (SRCG)[4], which is equivalent to LCFRS (Boullier, 1998).

LCFRS has been established as an appropriate formalism for the modeling of the syntax of natural languages, as it has the capability to model discontinuities (see section 2.1.3). It has also been demonstrated that data-driven LCFRS parsing is feasible (see section 2.1.4). LCFRS will furthermore be used within this work as the translation grammar formalism for SMT.

DEFINITION 2.29 (Linear Context-Free Rewriting System). A *Linear Context-Free Rewriting System (LCFRS)* is a tuple $G = (N, T, V, P, S)$ where

1. N is a finite set of non-terminals with a function $dim\colon N \to \mathbb{N}$;
2. T and V are disjoint finite sets of terminals and variables;
3. $S \in N$ is the distinguished start symbol with $dim(S) = 1$; and
4. P is a finite set of *(rewriting) rules* of the form

$$A(\alpha_1, \ldots, \alpha_{dim(A)}) \to A_1(Y_1^{(1)}, \ldots, Y_{dim(A_1)}^{(1)}) \\ \cdots A_m(Y_1^{(m)}, \ldots, Y_{dim(A_m)}^{(m)})$$

where, for $m \geq 0$, $A, A_1, \ldots, A_m \in N$ are non-terminals, $Y_j^{(i)} \in V$ for $1 \leq i \leq m$, $1 \leq j \leq dim(A_i)$ are variables, and $\alpha_i \in (T \cup V)^*$ for $1 \leq i \leq dim(A)$ are arguments.
Furthermore, for all $r \in P$, it holds that every variable $Y \in V$ that occurs in r occurs exactly once in the LHS and exactly once in the RHS of r. □

We now define the rank and the fan-out of LCFRS.

4 A rewriting rule in the SRCG notation can be viewed as a rule in the original formulation of LCFRS encoding the corresponding yield function at the same time.

DEFINITION 2.30 (Rank, fan-out (LCFRS)). Let $G = (N, T, V, P, S)$ be an LCFRS. The *rank* of a rule $r \in P$ is m, the number of non-terminals in the RHS of r. The *fan-out* of a non-terminal $A \in N$ is $dim(A)$. The rank of G is the maximal rank of any of its rules $r \in P$, and the fan-out of G is the maximal fan-out of any if its non-terminals $A \in N$. G is called a v-LCFRS if it has fan-out $\leq v$ and a (u, v)-LCFRS if it has rank $\leq u$ and fan-out $\leq v$. □

We may call a rule $A(\alpha_1, \ldots, \alpha_{dim(A)}) \rightarrow \boldsymbol{\Psi} \in P$ a *terminal rule* if all arguments of the LHS non-terminal consist of terminal symbols, i. e. $\alpha_i \in T^*$ for $1 \leq i \leq dim(A)$, and we may call it a *mixed rule* if the arguments of the LHS non-terminal contain at least one terminal and one variable. We may call the rule a *discontinuous rule* if its LHS non-terminal A has fan-out $v > 1$, the reason being its ability to build discontinuous constituents.

Intuitively, an LCFRS rule r describes how to compute the yield of the LHS non-terminal from the yields of the RHS non-terminals. The rules can be *instantiated* with respect to some input string w. This is a useful concept for, e. g., defining the language of an LCFRS or for parsing. For this, we will first define the ranges of a certain string and a vector of ranges.

DEFINITION 2.31 (Range). Let Σ be an alphabet and $w \in \Sigma^*$ a string with $w = w_1 \ldots w_n$, $n \in \mathbb{N}$. We define:

1. $Pos(w) = \{0, \ldots, n\}$.
2. A *range* in w is a pair $\langle l, r \rangle \in Pos(w) \times Pos(w)$ with $l \leq r$. Its *yield* $\langle l, r \rangle(w)$ is the substring $w_{l+1} \ldots w_r$.
3. For two ranges $\rho_1 = \langle l_1, r_1 \rangle$ and $\rho_2 = \langle l_2, r_2 \rangle$, if $r_1 = l_2$, i. e. if the two ranges are adjacent, then the concatenation of ρ_1 and ρ_2 is $\rho_1 \cdot \rho_2 = \langle l_1, r_2 \rangle$, otherwise the concatenation $\rho_1 \cdot \rho_2$ is undefined.

□

DEFINITION 2.32 (Range vector). Let Σ be an alphabet and $w \in \Sigma^*$.

1. A k-dimensional *range vector* for w is defined as $\boldsymbol{\rho} \in (Pos(w) \times Pos(w))^k$ with $\boldsymbol{\rho} = \langle \langle l_1, r_1 \rangle, \ldots, \langle l_k, r_k \rangle \rangle$, $k \in \mathbb{N}$, where each $\langle l_i, r_i \rangle$, $1 \leq i \leq k$, is a range in w.
2. $\boldsymbol{\rho}(i)$, $1 \leq i \leq k$, denotes the ith range $\langle l_i, r_i \rangle$ of $\boldsymbol{\rho}$. $\boldsymbol{\rho}(i).l$ (respectively $\boldsymbol{\rho}(i).r$) denotes the first (respectively second) component of $\boldsymbol{\rho}(i)$, which is l_i (respectively r_i).

3. The *yield* of a range vector $\boldsymbol{\rho}(w)$ is defined as

$$\langle \boldsymbol{\rho}(1)(w), \ldots, \boldsymbol{\rho}(k)(w) \rangle,$$

   i. e. as the vector of the yields of the individual ranges.

4. The *(yield) length* of range vector $\boldsymbol{\rho}$, notated as $|\boldsymbol{\rho}|$, is defined as

$$\sum_{i=1}^{k} (\boldsymbol{\rho}(i).r - \boldsymbol{\rho}(i).l).$$

□

In an instantiated rule, variables and terminals of the rule are consistently mapped to ranges (Boullier, 2000; Kallmeyer, 2010). We follow the definition in Maier (2013) and first introduce a numbering of the argument elements.

DEFINITION 2.33 (Argument numbering). Let $G = (N, T, V, P, S)$ be an LCFRS. For each $r \in P$,

1. we first assume all occurrences of terminals, all occurrences of ε and the respective first occurrences of variables in r to be consecutively numbered with a unique index with respect to their occurrence from left to right, starting with 1, and

2. ξ_r^{max} stores the maximal index of r under the numbering of r, and

3. ξ_r is a function which yields the variable or the occurrence of a terminal, or the occurrence of ε in r at index i, $1 \leq i \leq \xi_r^{max}$. □

DEFINITION 2.34 (Rule instantiation). Let $G = (N, T, V, P, S)$ be an LCFRS. Let $w \in T^*$ be a string. For a rule $r \in P$ with $\xi_r^{max} = k$, the following is defined:

1. An *instantiation* of r with respect to w is given by a k-dimensional range vector ϕ where $\phi(i)$, for $1 \leq i \leq k$, contains the range to which $\xi_r(i)$ is bound. Thereby,
   a) variables which are adjacent in r must be mapped to adjacent ranges, and
   b) for $0 \leq j < |w|$,

   i. if $\xi_r(i)$ is an occurrence of a terminal t, it must be mapped on a range $\phi(i) = \langle j, j+1 \rangle$ with $w_{j+1} = t$, and

   ii. if $\xi_r(i)$ is an occurrence of ε, it must be mapped on a range $\phi(i) = \langle j, j \rangle$.

2. Applying ϕ to a non-terminal $A(\boldsymbol{\alpha})$ in r, notated as $\phi(A(\boldsymbol{\alpha}))$, is defined as a mapping of all occurrences of terminals and ε and all variables in $\boldsymbol{\alpha}$ to elements of ϕ such that each $\xi_r(i)$, for $1 \leq i \leq k$, is mapped to $\phi(i)$.
   If the result is defined, i. e. if the range images of adjacent variables and terminals can be concatenated, then it is called an *instantiated non-terminal*. We write $A(\boldsymbol{\rho})$ to designate a non-terminal $A(\boldsymbol{\alpha})$ that is instantiated with ϕ.
   If the result of applying ϕ to all non-terminals in r, notated as $\phi(r)$, is defined, then the result of the mapping is called an *instantiated rule*. □

In an LCFRS derivation step, the LHS of an instantiated rule is replaced with its RHS.

DEFINITION 2.35 (Derivation (LCFRS)). Let $G = (N, T, V, P, S)$ be an LCFRS and $w \in T^*$ a string.

1. $\Rightarrow_{G,w}$ is a relation called *derives* between two strings of instantiated non-terminals. It is defined as follows: If $A(\boldsymbol{\rho}) \rightarrow A_1(\boldsymbol{\rho}_1) \ldots A_m(\boldsymbol{\rho}_m)$ is an instantiated rule $r \in P$ with respect to w, then

   $$\Gamma A(\boldsymbol{\rho}) \Gamma' \Rightarrow_{G,w} \Gamma A_1(\boldsymbol{\rho}_1) \ldots A_m(\boldsymbol{\rho}_m) \Gamma'.$$

   Γ, Γ' are strings of instantiated non-terminals. If G and w are clear in the context, we can use $\Rightarrow$ instead of $\Rightarrow_{G,w}$. To make the applied rule explicit, we can use $\Rightarrow^r_{G,w}$.

2. $\overset{*}{\Rightarrow}_{G,w}$ is the reflexive transitive closure of $\Rightarrow_{G,w}$.

3. Let $\Gamma_1, \ldots, \Gamma_m$ be strings of instantiated non-terminals, $m \geq 1$. $\Gamma_1 \Rightarrow_{G,w} \ldots \Rightarrow_{G,w} \Gamma_m$ is a *derivation* of length m. We also write $\Gamma_1 \overset{m}{\Rightarrow}_{G,w} \Gamma_m$. □

Note that the set of instantiated rules with respect to some input w is a CFG whose rules are the instantiated LCFRS rules and whose non-terminals are the instantiated LCFRS non-terminals.

The set of all strings that can be reduced to the empty string is the *language* of an LCFRS.

DEFINITION 2.36 (Language (LCFRS)). Let $G = (N, T, V, P, S)$ be an LCFRS. The *language* of G is $\mathcal{L}(G) = \{w \mid S(\langle 0, |w| \rangle) \stackrel{*}{\Rightarrow}_{G,w} \varepsilon\}$. □

One way of encoding an LCFRS derivation as a tree is presented in Boullier (1998), namely as the derivation tree of the previously mentioned CFG, in which the nodes are instantiated non-terminals. Alternatively, we use the definition presented, e. g., in Kallmeyer (2010) and Maier (2013). In correspondence to CFG derivation trees, the leaves are labeled with terminals and they are ordered, and the inner nodes are labeled with non-terminal symbols.

DEFINITION 2.37 (Derivation tree (LCFRS)). Let $G = (N, T, V, P, S)$ be an LCFRS. Let $w = w_1 \dots w_n$, $w \in T^*$ be a string. A *derivation tree* for G for a derivation $S(\langle 0, |w| \rangle) \stackrel{*}{\Rightarrow}_{G,w} \varepsilon$ is an ordered tree $\mathcal{D} = \langle V_\mathcal{D}, E_\mathcal{D}, r \rangle$ with a node labeling $l : V_\mathcal{D} \to N \cup T \cup \{\varepsilon\}$ and a function $pos : \{v \in V_\mathcal{D} \mid f_{out}(v) = 0, \text{ i.e. } v \text{ is a leaf}\} \to \mathbb{N}$. The following must hold for $\mathcal{D}$:

1. The labeling l is such that
   a) for all $v \in V_\mathcal{D}$, if $f_{out}(v) = 0$, then $l(v) \in T \cup \{\varepsilon\}$, otherwise $l(v) \in N$,
   b) for all w_i, $1 \leq i \leq n$, there is exactly one $v \in V_\mathcal{D}$ with $l(v) = w_i$.
2. For all $v, v_1, \dots v_k \in V_\mathcal{D}$ where $v_1, \dots, v_k$ are the only and all children of v, let $V_{\mathcal{D},N} = \{v_i \mid v_i \in \{v_1, \dots, v_k\} \text{ and } l(v_i) \in N\}$, there exists a rule $r = A(\boldsymbol{\alpha}) \to A_1(\boldsymbol{\alpha}_1) \dots A_m(\boldsymbol{\alpha}_m) \in P$, $m \geq 0$, which can be instantiated with respect to w by a range vector $\boldsymbol{\phi}$, such that
   a) $|\{q \mid 1 \leq q \leq |\boldsymbol{\phi}| \text{ and } \xi_r(q) \notin V\}| = k - m$
   b) for all $1 \leq q \leq |\boldsymbol{\phi}|$ where $\xi_r(q) \notin V$, there is exactly one $v' \in \{v_1, \dots, v_k\}$ such that $l(v') = \xi_r(q)$ and $pos(v') = \boldsymbol{\phi}(q).l$,
   c) $l(v) = A$,
   d) $|V_{\mathcal{D},N}| = m$ and $V_{\mathcal{D},N}$ can be ordered in such a way that the jth element of $V_{\mathcal{D},N}$ has label A_j, for $1 \leq j \leq m$.
3. $\prec$ is as follows: for all $v', v'' \in \{v \in V_\mathcal{D} \mid l(v) \in T \cup \{\varepsilon\}\}$, $v' \prec v''$ iff $pos(v') < pos(v'')$.

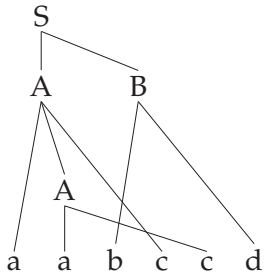


Figure 11: LCFRS derivation tree

Again, $\mathcal{D}$ is called a *partial derivation tree* if $l(r) \neq S$. □

Example 2.38 (Linear Context-Free Rewriting System). Consider the following LCFRS $G = (\{S, A, B\}, \{a, b, c, d\}, \{Y_1, Y_2, Y_3, Y_4\}, P, S)$ where P contains the following rules:

$$
\begin{aligned}
r_1 &: \quad S(Y_1Y_2Y_3Y_4) \rightarrow A(Y_1, Y_3)B(Y_2, Y_4) \\
r_2 &: \quad A(aY_1, cY_2) \rightarrow A(Y_1, Y_2) \\
r_3 &: \quad B(bY_1, dY_2) \rightarrow B(Y_1, Y_2) \\
r_4 &: \quad A(a, c) \rightarrow \varepsilon \\
r_5 &: \quad B(b, d) \rightarrow \varepsilon
\end{aligned}
$$

The language of G is

$$\mathcal{L}(G) = \{a^n b^m c^n d^m \mid n, m > 0\},$$

i. e. it generates cross-serial dependencies which are beyond the expressivity of CFG. The rank of G is 2 and its fan-out is 2. In the following, we give the derivation of $w = {}_0a_1a_2b_3c_4c_5d_6$ under G. The subscripts in w denote the positions. Figure 11 shows the corresponding derivation tree.

$$
\begin{aligned}
S(\langle 0,6\rangle) &\Rightarrow^{r_1} A(\langle 0,2\rangle, \langle 3,5\rangle)\ B(\langle 2,3\rangle, \langle 5,6\rangle) \\
&\Rightarrow^{r_2} A(\langle 1,2\rangle, \langle 4,5\rangle)\ B(\langle 2,3\rangle, \langle 5,6\rangle) \\
&\Rightarrow^{r_4} B(\langle 2,3\rangle, \langle 5,6\rangle) \\
&\Rightarrow^{r_5} \varepsilon
\end{aligned}
$$

We further illustrate rule instantiation at the example of rule r_2 with respect to w. The argument numbering of r_2 is given by $\xi_{r_2}(1) =$

$a, \xi_{r_2}(2) = Y_1, \xi_{r_2}(3) = c, \xi_{r_2}(4) = Y_2$. $\phi = \langle\langle 0,1\rangle, \langle 1,2\rangle, \langle 3,4\rangle, \langle 4,5\rangle\rangle$ represents an instantiation of r_2 with respect to w. We associate each occurrence of a variable or terminal in r_2 to a range by using the mapping from $\xi_{r_2}(i)$ to $\phi(i)$. The result is defined according to def. 2.34, p. 28, i. e. we obtain the following instantiated rule:

$$A(\langle 0,2\rangle, \langle 3,5\rangle) \to A(\langle 1,2\rangle, \langle 4,5\rangle).$$

As an example of a failing rule instantiation, consider $\phi' = \langle\langle 0,1\rangle, \langle 1,2\rangle, \langle 2,3\rangle, \langle 4,5\rangle\rangle$. $\xi_{r_2}(3)$ is the occurrence of the terminal c, but the mapped range $\langle 2,3\rangle$ yields b. Furthermore, c and Y_2 are adjacent, but the concatenation of their mapped ranges $\langle 2,3\rangle \cdot \langle 4,5\rangle$ is undefined. □

LCFRS is equivalent to a range of formalisms which are used for natural language description, amongst others, SRCG and Multiple Context-Free Grammar (MCFG). Furthermore, 1-LCFRS is strongly equivalent to CFG.

The LCFRS terminology differs from the SRCG terminology. The elements of N are called *predicates* and the elements of P are called *clauses* in the SRCG literature. Furthermore, the fan-out of an LCFRS corresponds to the *arity* of a SRCG. The SRCG terms are mentioned for reference here, but we will continue to use the LCFRS terminology in the following.

Parsing can be facilitated if an ordering on the variables is assumed (Villemonte de la Clergerie, 2002).

Definition 2.39 (Monotone LCFRS). An LCFRS $G = (N, T, V, P, S)$ is *monotone* if for every rule $A(\boldsymbol{\alpha}) \to A_1(\boldsymbol{\alpha}_1) \dots A_m(\boldsymbol{\alpha}_m) \in P$ it holds that if a variable Y_1 precedes a variable Y_2 in $\boldsymbol{\alpha}_i$, for all $1 \leq i \leq m$, then Y_1 also precedes Y_2 in $\boldsymbol{\alpha}$. □

For every LCFRS, there exists an equivalent monotone LCFRS (Villemonte de la Clergerie, 2002; Kracht, 2003; Kallmeyer, 2010, p. 145). Generally the construction is exponential in the size of the original grammar. However, usually, natural language LCFRS extracted from treebanks are monotone by nature of the extraction algorithm (Maier and Søgaard, 2008). The property of being monotone is called *ordered* for SRCG.

Furthermore, parsing algorithms often assume that grammars are *ε-free*.

DEFINITION 2.40 (ε-free LCFRS). Let $G = (N, T, V, P, S)$ be an LCFRS. A rule $r \in P$ is called an *ε-rule* if one of its LHS arguments is the empty string ε. G is *ε-free* if either P does not contain any ε-rules or the only ε-rule it contains is the rule $S(\varepsilon) \to \varepsilon$ and S does not appear in any RHS of the other rules in P. □

Just as CFG, LCFRS and equivalent formalisms have also been extended with the notions of weights and probabilities; see, for example, Levy (2005), Kato et al. (2006), Maier (2013) and Denkinger (2015).

DEFINITION 2.41 (Weighted Linear Context-Free Rewriting System). A *weighted Linear Context-Free Rewriting System* is a tuple

$$G = (N, T, V, P, S, \omega)$$

where (N, T, V, P, S) is an LCFRS and $\omega : P \to \mathbb{R}_{\geq 0}$ is a weight function which maps from rules to positive real numbers. □

DEFINITION 2.42 (Weight of a derivation (Weighted LCFRS)). Let $G = (N, T, V, P, S, \omega)$ be a weighted LCFRS. Let $w \in T^*$ be a string and Γ, Γ' strings of instantiated non-terminals of rules in P with respect to w.

1. Let $A(\boldsymbol{\alpha}) \to \boldsymbol{\Psi} \in P$. The weight of a derivation step $\Gamma \Rightarrow_{G,w}^{A(\boldsymbol{\alpha}) \to \boldsymbol{\Psi}} \Gamma'$ is defined as

$$\omega(\Gamma \Rightarrow_{G,w}^{A(\boldsymbol{\alpha}) \to \boldsymbol{\Psi}} \Gamma') = \omega(A(\boldsymbol{\alpha}) \to \boldsymbol{\Psi})$$

2. Let $A_1(\boldsymbol{\alpha}_1) \to \boldsymbol{\Psi}_1, \ldots, A_m(\boldsymbol{\alpha}_m) \to \boldsymbol{\Psi}_m \in P$, $m \in \mathbb{N}$. The weight of a derivation $\Gamma \Rightarrow_{G,w}^{A_1(\boldsymbol{\alpha}_1) \to \boldsymbol{\Psi}_1} \ldots \Rightarrow_{G,w}^{A_m(\boldsymbol{\alpha}_m) \to \boldsymbol{\Psi}_m} \Gamma'$ is defined as

$$\omega(\Gamma \Rightarrow_{G,w}^{A_1(\boldsymbol{\alpha}_1) \to \boldsymbol{\Psi}_1} \ldots \Rightarrow_{G,w}^{A_m(\boldsymbol{\alpha}_m) \to \boldsymbol{\Psi}_m} \Gamma') = \prod_{i=1}^{m} \omega(A_i(\boldsymbol{\alpha}_i) \to \boldsymbol{\Psi}_i)$$

□

DEFINITION 2.43 (Probabilistic Linear Context-Free Rewriting System). A *probabilistic Linear Context-Free Rewriting System* is a weighted LCFRS $G = (N, T, V, P, S, \omega)$ with $\omega : P \to [0, 1]$ being such that for all $A \in N$ of a fixed dimension $dim(A) = k$ the following holds:

$$\sum_{A(\alpha_1, \ldots, \alpha_k) \to \boldsymbol{\Psi} \in P} \omega(A(\alpha_1, \ldots, \alpha_k) \to \boldsymbol{\Psi}) = 1$$

□

2.1.2 *Parsing Fundamentals*

Fundamental concepts and algorithms of parsing are covered in this section. As such, it also serves as the basis for tree-based decoding in the context of SMT.

Given a grammar G and an input string $w = w_1^n = w_1 \dots w_n$ with $n \in \mathbb{N}$ over some alphabet Σ, deciding whether G generates w, i. e. whether $w \in \mathcal{L}(G)$, is called the *recognition problem*. We can speak of the *fixed recognition problem* if G is fixed, otherwise of the *universal recognition problem*. In the first case, we are interested in the complexity of the problem with respect to the input size n only, while in the latter the size of the grammar G is also taken into account. Since natural language grammars tend to be rather large, the universal recognition problem is of interest in NLP.

An algorithm which solves the recognition problem is called a *recognition algorithm* or just a *recognizer*. A *parser* (or *parsing algorithm*) in addition also records the individual derivation steps that are taken and emits the resulting structure (the parse tree) if w is recognized.

Parsing as Deduction

Parsing strategies can be notated as algorithmic descriptions, for example pseudo code. Such formulations usually involve *data structures* and *control structures* which the parsing strategy itself does not depend on. *Parsing as deduction* (Pereira and Warren, 1983; Shieber et al., 1995; Sikkel, 1997) offers a way to separate the parsing algorithm from the control strategy.

A *deduction system* specifies a parsing algorithm in a declarative way, which allows to concentrate on the properties of the algorithm instead of on the implementation. The specification of a parsing strategy as a deduction system requires the following concepts:

1. *Items* designate intermediate parsing results. They characterize the status of (parts of) the input string w with respect to the given grammar G.

2. A finite set of *deduction rules* (also called *inference rules*) which determine how to deduce new items from existing ones.

3. *Goal items* characterize successful parses. If a goal item can be deduced for a given string w, the string is recognized by the grammar.

Deduction rules have the general form

$$\frac{A_1 \dots A_m}{B} \quad c_1 \dots c_k$$

with $m, k \in \mathbb{N}_0$. The *antecedents* $A_1 \dots A_m$ and the *consequent* B are *items*. $c_1 \dots c_k$ are conditions on $A_1 \dots A_m$ and B which usually provide a link to the grammar G and the input w. The interpretation is that if $A_1 \dots A_m$ can be deduced and the conditions $c_1 \dots c_k$ hold, then B can be deduced. For $m = 0$, i. e. deduction rules without antecedents, B is called an *axiom*.

EXAMPLE 2.44 (Deduction System). As an example, we now present one of the most popular parsing strategies, the *CYK algorithm* (Cocke and Schwartz, 1970; Younger, 1967; Kasami, 1965) for CFG as a deduction system. CYK is a bottom-up non-directional parsing strategy.

Given a CFG $G = (N, T, P, S)$ in CNF and an input string $w = w_1 \dots w_n$, items have the form $[A, i, j]$ where $A \in N$ and $0 \leq i < j \leq n$. Their interpretation is that $A \overset{*}{\Rightarrow} w_{i+1} \dots w_j$. The deduction rules are the following:

$$\textsc{scan}: \quad \frac{}{[A, i, i+1]} \quad A \to w_{i+1} \in P$$

$$\textsc{complete}: \quad \frac{[B, i, j] \; [C, j, k]}{[A, i, k]} \quad A \to BC \in P$$

The SCAN deduction rule creates items for the terminals in the input string w. The COMPLETE operation combines two adjacent items to derive a new item. The input string w is recognized if the GOAL item $[S, 0, n]$ is derived

The time complexity for CYK parsing with CFG is $\mathcal{O}(n^3)$. This can be seen from the deduction system as well. The most complex inference rule is the COMPLETE operation, which involves three independent indices i, j and k ranging from 0 to $n - 1$ or from 1 to n. They determine the number of inference rules instantiations c, which is $c \leq |P| \, n^3$. □

Natural language grammars are usually highly ambiguous. During parsing, it is necessary to be able to compute several or all possible analyses for one input string. Furthermore, different analyses can have common sub-analyses for specific substrings. They need to be stored and retrieved efficiently in order to avoid recomputing them over and over again. Implementations of parsers therefore commonly store intermediate parsing results, which can then be reused in different contexts (*computation sharing*). This corresponds to the approach of *dynamic programming* where sub-problems are solved first and then combined in order to solve the overall problem (Bellman, 1957). The formulation of a parsing algorithm as a deduction system lends itself perfectly to this paradigm. Items represent intermediate parsing results. Once they are deduced, they can serve as antecedents in different rule applications.

The standard data structure for storing items is a *chart*. It should be indexed in a way to enable efficient retrieval of items. For instance, the chart in which CYK items $[A, i, j]$ (cf. ex. 2.44, p. 35) are stored is typically implemented as a 3-dimensional table with dimensions for the start index i, the end index j and the non-terminal label A.

A general algorithm for *chart parsing* is provided in algorithm 1. Besides the chart $\mathcal{C}$, it uses an additional data structure $\mathcal{A}$, an agenda, which controls how consequents are produced. If in the end, $\mathcal{C}$ contains a goal item, we can conclude that the input string w is recognized by the grammar, i. e. that $w \in \mathcal{L}(G)$, otherwise $w \notin \mathcal{L}(G)$,

As formulated so far, the chart parsing algorithm is only a *recognizer*. In order to extend it to a real *parser*, each item needs to keep track of its antecedents. This is usually done via references, called *backpointers*. Obviously, because of ambiguity and computation sharing, each item needs a list of backpointers. The items in the chart together with the backpointers compactly represent a *parse forest*. Individual parse trees are obtained by following the backpointers starting with a goal item. For an example of a parse chart with backpointers, the reader is referred to ex. 2.46, p. 40.

For *weighted parsing*, the rules of the grammar G and the derivations are equipped with weights. The weights can be used, e. g., for disambiguation or for ranking parse trees. Given such a weighted grammar and an input string w, finding the most probable derivation for w, or the derivation with the best (often minimal, however throughout this thesis maximal) weight, is one of the predominant problems in NLP.

```
1: G: a grammar of formalism 𝔉
2: w: the input string
3: 𝒞 = ∅
4: 𝒜 = ∅
5: for all items I′ which are consequents of inference rule applications with
   empty antecedent set, instantiated with rewriting rules from G and posi-
   tions in w do
6:     add I′ to 𝒜 and 𝒞
7: end for
8: while 𝒜 ≠ ∅ do
9:     remove an item I from 𝒜
10:    for all items I′ which are consequents of inference rule applications
       with I and possibly other items from 𝒞 in their antecedent, instanti-
       ated with rewriting rules from G and positions in w do
11:        if I′ ∉ 𝒞 then
12:            add I′ to 𝒞 and 𝒜
13:        end if
14:    end for
15: end while
```

Algorithm 1: Generic chart parsing (from Kallmeyer (2010, p. 47))

The corresponding probability or weight is sometimes referred to as the *Viterbi score* according to one of the earliest algorithms for finding it (Viterbi, 1967).

Weighted deduction systems are useful to specify weighted parsers (Goodman, 1999; Nederhof, 2003). They are based on the deduction systems as presented earlier in this section. The deduction rules are extended with weights. They have the form

$$\frac{x_1 : A_1 \ \ldots \ x_m : A_m}{f(x_1, \ldots, x_m) : B} \quad c_1 \ldots c_k$$

with again $m, k \in \mathbb{N}_0$, the *antecedents* $A_1 \ldots A_m$ and the *consequent* B being *items* and $c_1 \ldots c_k$ being side conditions. $x_1 \ldots x_m$ are unique weight variables for each antecedent and f is the *weight function* assigning a weight to the instantiated consequent when the rule is applied.

The definition of the weight function obviously depends on the definition of the weight of a derivation of a specific grammar formalism. Commonly the logarithms of the weights of the weighted or proba-

bilistic grammar G are used as weights in the deduction system, together with a summation weight function f_Σ.

EXAMPLE 2.45 (Weighted Deduction System). We present a weighted deduction system for CFG parsing. In contrast to the CYK algorithm in ex. 2.44, p. 35, this *modified CYK algorithm* does not require the grammar G to be in CNF (Chiang, 2007). However, it is assumed that G is ε-free and has maximally rank 2. Notably, G is allowed to have mixed rules.[5] The items I also have the form $[A, i, j]$ and the interpretation $A \overset{*}{\Rightarrow} w_{i+1} \dots w_j$. The weight function f is $f(x_1, \dots, x_m) = x_1 \cdot \ldots \cdot x_m \cdot \omega(r)$. The deduction rules and the GOAL item are as given in the following:

SCAN:

$$\frac{}{\omega(r) : [A, i, j]} \qquad r = A \to w_{i+1}^{j} \text{ and } r \in P$$

UNARY:

$$\frac{x_1 : [B, i_1, j_1]}{x_1 \omega(r) : [A, i, j]} \qquad r = A \to w_{i+1}^{i_1} B\, w_{j_1+1}^{j} \text{ and } r \in P$$

BINARY:

$$\frac{x_1 : [B, i_1, j_1]\ x_2 : [C, i_2, j_2]}{x_1 x_2 \omega(r) : [A, i, j]} \qquad r = A \to w_{i+1}^{i_1} B\, w_{j_1+1}^{i_2} C\, w_{j_2+1}^{j} \text{ and } r \in P$$

$$\text{GOAL:} \quad x : [S, 0, n]$$

The SCAN deduction rule again builds items from terminal rules. The deduction rules UNARY and BINARY create new items from one, respective two antecedents and a rule r.

The time complexity of this parser is also $\mathcal{O}(n^3)$. This can be seen from the BINARY rule, which is the most complex one in the deduction system. In the case when r does not contain any terminals in the RHS, i. e. when $w_{i+1}^{i_1}$, $w_{j_1+1}^{i_2}$ and $w_{j_2+1}^{j}$ are all empty, none of the indices is already fixed. Furthermore, in this case, $i = i_1$, $j_1 = i_2$ and $j = j_2$, and

5 While grammars for parsing obtained from treebanks usually do not contain mixed rules, they are frequent and essential in tree-based translation models (cf. section 2.2.5). This kind of parser therefore has its application, e. g., in decoding with translation models based on Synchronous Context-Free Grammar (SCFG).

```
 1: G: a grammar of formalism 𝔉
 2: w: the input string
 3: 𝒞 = ∅
 4: 𝒜 = ∅
 5: for all items x : I′ which are consequents of inference rule applications
    with empty antecedent set, instantiated with rewriting rules from G and
    positions in w do
 6:     add x : I′ to 𝒜
 7:     add a backpointer from I′ to ε
 8: end for
 9: while 𝒜 ≠ ∅ do
10:     remove a smallest item y : I from 𝒜 according to ≺
11:     add y : I to 𝒞
12:     for all items x : I′ which are consequents of inference rule applications with I and possibly other items from 𝒞 in their antecedent,
        instantiated with rewriting rules from G and positions in w do
13:         if I′ ∉ 𝒞 ∪ 𝒜 then
14:             add x : I′ to 𝒜
15:             add a backpointer from I′ to its antecedents
16:         else if z : I′ ∈ 𝒜 for some z and x > z then
17:             update weight z of I′ in 𝒜 to x
18:             update the backpointer of I′
19:         end if
20:     end for
21: end while
```

Algorithm 2: Generic Viterbi chart parsing

the rule is reduced to the COMPLETE rule of the deduction system for CYK parsing with CFG in CNF in ex. 2.44, p. 35. A non-empty $w_{i+1}^{i_1}$ means that i and i_1, i. e. the beginning of B, are already fixed, resulting in one free index less and therefore a lower complexity. The same argument holds for $w_{j_1+1}^{i_2}$ and $w_{j_2+1}^{j}$. □

To find the goal item with the best (highest) weight, an extension of algorithm 1 can be used (Viterbi, 1967; Jelinek et al., 1992; Nederhof, 2003), see algorithm 2. It is applicable if the weight function f is monotonic (cf. def. 2.49, p. 43) and a partial order $\prec$ on the items exists such that the antecedents of an instantiated inference rule are

always strictly smaller than the consequent.[6] In this case, items can be treated from small to large to compute their best weight. As an example consider the deduction system in ex. 2.44, p. 35 for CYK parsing of a CFG in CNF: the partial order is determined by the sizes of the spans of the items, i. e. $j - i$ for an item of the form $[A, i, j]$. Nederhof (2003) also notes that, in the case of acyclic derivations and no partial order being a priori available, a topological sorting of all derivable items can be determined instead, to then compute the weights of the items according to that order.

In algorithm 2, the agenda $\mathcal{A}$ returns items respecting their partial order. When an item I is taken off the agenda, it is guaranteed to have reached its maximum weight as all possible analyses and antecedents have been considered before and I has been updated accordingly. Again, in the end, if $\mathcal{C}$ contains a goal item, w is recognized, and the backpointers can be used to trace back the derivation with the highest weight.

Instead of using the generic chart parsing algorithm in algorithm 2, for many parsing algorithms, more specific procedures can be used to enumerate items according to the item order. Reconsider ex. 2.45, p. 38 for instance. For a weighted CYK deduction system for CFG, items are typically produced from small to large using three nested loops ranging from 1 to n, one for each index i, j, and $j_1 = i_2$. From an efficiency perspective, this alleviates the overhead incurred by the priority queue $\mathcal{A}$.

6 Another algorithm for weighted deductive parsing based on Knuth's (1977) algorithm is presented in Nederhof (2003). It is commonly used for probabilistic parsing, but does not fit our presentation here as it solves the problem of finding the item with the *smallest* weight. It does not require an order on the items, but an additional condition on the weight function: $f(x_1, \ldots, x_m) \geq \max(x_1, \ldots, x_m)$ for all $x_1, \ldots, x_m$. In contrast to algorithm 2, Knuth's algorithm does not necessarily compute all derivable items in order to find the one with the best weight. However, a priority queue which sorts the items according to their weight is required in any case and adds some computational overhead. The reader is referred to Nederhof (2003) for more details and a comparison.

EXAMPLE 2.46 (Weighted CFG CYK parsing). Let $G = (\{\mathrm{S}, \mathrm{A}, \mathrm{B}, \mathrm{C}\}, \{a, b, c, d, x\}, P, \mathrm{S}, \omega)$ be a weighted CFG with P and ω as follows:

i	r_i	$\omega(r_i)$
1	$\mathrm{S} \to \mathrm{B}\,d$	1.0
2	$\mathrm{B} \to \mathrm{AC}$	0.8
3	$\mathrm{B} \to abc$	0.2
4	$\mathrm{A} \to ab$	0.4
5	$\mathrm{A} \to x$	0.6
6	$\mathrm{C} \to c$	1.0

All items which are generated when parsing the input $w = abcd$ using the modified CYK deduction system for CFG in ex. 2.45, p. 38 and algorithm 2 are shown in table 1. Note that item $[\mathrm{B}, 0, 3]$ has two derivations. The one with the larger weight (3b) is retained. The GOAL item is item 4. Items 1, 2 and 3b lead to it. The corresponding CFG derivation and its weight is

$$\omega(\mathrm{S} \Rightarrow \mathrm{B}d \Rightarrow \mathrm{AC}d \Rightarrow ab\mathrm{C}d \Rightarrow abcd) = 1.0 \cdot 0.8 \cdot 0.4 \cdot 1.0 = 0.32$$

The corresponding chart is depicted in figure 12. The horizontal dimension denotes the start index i of an item, while the vertical dimension denotes its end index j. Each item itself is represented by its non-terminal label (and its best weight). The arrows are the hyperedges (or the inverted backpointers). Note that with algorithm 2 each item has only one backpointer/incoming hyperedge. The dashed hyperedge corresponds to derivation 3a, which is inferior to 3b. □

Parsing and Hypergraphs

Besides the formalization of parsing as a deductive process, parsing is also closely connected to hypergraphs and algorithms thereon (Klein and Manning, 2001). More specifically, the search space of a (weighted) deduction system can be represented by a hypergraph. This perspective is in particular suitable for *k-best parsing*, i. e. for finding the k-best derivations for an input w, which is usually formulated as a search on a hypergraph (Huang and Chiang, 2005). These works have also been highly influential for SMT, e. g., in the context of generating the k-best

No.	Item	Rule	Antecedents	Weight
1	[C, 2, 3]	SCAN	-	1.0
2	[A, 0, 2]	SCAN	-	0.4
3a	[B, 0, 3]	SCAN	-	0.2
3b		BINARY	1 & 2	$1.0 \cdot 0.4 \cdot 0.8 = 0.32$
4	[S, 0, 4]	UNARY	3b	$0.32 \cdot 1.0 = 0.32$

Table 1: Parse items for ex. 2.46

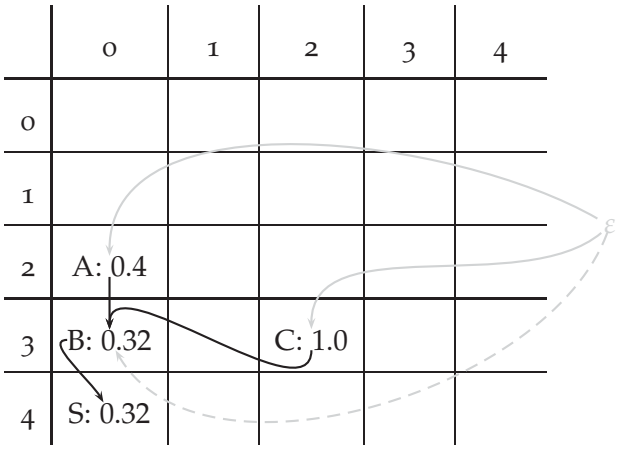


Figure 12: Parse chart/hypergraph for ex. 2.46

translations for feature weight optimization, and for the efficient integration of the language model into the decoding process (Huang and Chiang, 2007). Throughout this thesis, both concepts, parsing as deduction and parsing based on hypergraphs, will be used.

A parse forest (p. 36), i. e. an instantiated (weighted) deduction system, can also be viewed as a (weighted) *directed hypergraph* (Gallo et al., 1993).[7]

DEFINITION 2.47 (Hypergraph). A *directed, ordered hypergraph* is a tuple $\langle V, E \rangle$ where V is a finite set of nodes (also called vertices) and E is a finite set of directed hyperedges. Each directed hyperedge $e \in E$ is a pair $\langle \boldsymbol{T}(e), H(e) \rangle$ where $H(e) \in V$ is the *head* of e and $\boldsymbol{T}(e) \in V^*$ is a vector of *tail* nodes. □

The items in the parse chart are the nodes of the hypergraph, and the hyperedges correspond to the inverted backpointers. The ordering of the nodes in the tail is arbitrary, but usually implicitly defined by some characteristics contained in the item, e. g. the start position in w of the item. The ith node of the tail $\boldsymbol{T}(e)$ is referenced with $T_i(e)$. Some more definitions in relation to hypergraphs are provided here which will be used later on.

DEFINITION 2.48 (Hyperedge weight function). Given a hyperedge $e \in E$ of a hypergraph $\langle V, E \rangle$ and a set of weights $\mathbb{R}$,

$$f_e : \mathbb{R}^{|\boldsymbol{T}(e)|} \to \mathbb{R}$$

is a *hyperedge weight function* which assigns a weight to the hyperedge e. □

When the hypergraph represents an instantiated weighted deduction system, then f_e corresponds to the weight function f of one rule application of the deduction system.

DEFINITION 2.49 (Monotonicity). Let $H = \langle V, E \rangle$ by a hypergraph and f_e a weight function of a hyperedge $e \in E$. f_e is *monotonic* if there is a total order[8] $\preceq$ on $\mathbb{R}$ such that f_e is monotonic in each of its arguments, i. e. if $x_i \preceq x_i'$, then $f_e(x_1, \ldots, x_i, \ldots x_m) \preceq f_e(x_1, \ldots, x_i', \ldots x_m)$ for each $1 \leq i \leq m$. H is monotonic if the weight function f_e of each $e \in E$

7 Only the specific definition of a hypergraph which is useful for parsing is provided here. The definitions in this section mostly follow those given in Huang and Chiang (2005).

8 We deliberately use $\preceq$ for the order instead of $\leq$ as it will be extended beyond $\mathbb{R}$.

is monotonic under $\preceq$. $\max_{\preceq}(x_1, x_2)$ returns x_2 if $x_1 \preceq x_2$, otherwise x_1. □

DEFINITION 2.50 (Derivation (Hypergraph)). Given a hypergraph $\langle V, E \rangle$, a *derivation* D of a node $v \in V$ and its weight $w(D)$ are recursively defined as follows:

- If e is an incoming hyperedge of v and $|\boldsymbol{T}(e)| = 0$, then $D = \langle e, \varepsilon \rangle$ is a derivation of v and its weight is $w(D) = f_e()$.
- If e is an incoming hyperedge of v with $|\boldsymbol{T}(e)| > 0$ and, for $1 \leq i \leq |\boldsymbol{T}(e)|$, D_i is a derivation of $T_i(e)$, then $D = \langle e, D_1 \dots D_{|\boldsymbol{T}(e)|} \rangle$ is a derivation of v and its weight is $w(D) = f_e(w(D_1), \dots, w(D_{|\boldsymbol{T}(e)|}))$. □

The order on weights is extended to an order on derivations: $D \preceq D'$ iff $w(D) \preceq w(D')$. We use the notation $\boldsymbol{D}(v)$ to denote the $|\boldsymbol{D}(v)|$-best derivations of v, where $D_1(v)$ is the best derivation of v, $D_2(v)$ the second best and so forth.

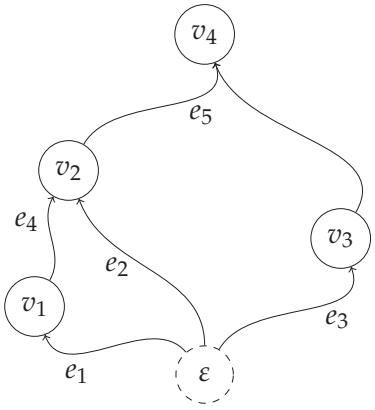
We use the notion of ranked derivations of v to give an alternative, non-recursive representation of derivations. It uses backpointers which point back to a specific kth-best derivation of each tail node of v.

DEFINITION 2.51 (Derivation with backpointers (Hypergraph)). Let $\langle V, E \rangle$ be a hypergraph. A *derivation with backpointers* $\hat{D}$ of $v \in V$ is a tuple $\langle e, \boldsymbol{x} \rangle$ where e is an incoming hyperedge of v and $\boldsymbol{x} \in \{1, \dots, k\}^{|\boldsymbol{T}(e)|}$ is a vector, the backpointer. The one-to-one correspondence $\sim$ between derivations of v and derivations with backpointers of v is the following:

$$\langle e, (x_1, \dots, x_{|\boldsymbol{T}(e)|}) \rangle \sim \langle e, D_{x_1}(T_1(e)) \dots D_{x_{|\boldsymbol{T}(e)|}}(T_{|\boldsymbol{T}(e)|}(e)) \rangle$$

□

Accordingly, $w(\hat{D}) = w(D)$ if $\hat{D} \sim D$, and $\hat{D} \preceq \hat{D}'$ iff $w(\hat{D}) \preceq w(\hat{D}')$. We use the notation $\hat{D}_i(v)$ to denote the ith-best derivation with backpointers of v. We will often refer to *derivations with backpointers* as just *derivations*.



d	$w(d)$
$\hat{D}_1(v_1) = \langle e_1, () \rangle$	1
$\hat{D}_1(v_2) = \langle e_4, (1) \rangle$	$1 + 4 = 5$
$\hat{D}_2(v_2) = \langle e_2, () \rangle$	2
$\hat{D}_1(v_3) = \langle e_3, () \rangle$	3
$\hat{D}_1(v_4) = \langle e_5, (1,1) \rangle$	$5 + 3 + 5 = 13$
$\hat{D}_2(v_4) = \langle e_5, (2,1) \rangle$	$2 + 3 + 5 = 10$

Figure 13: An example hypergraph, the derivations of the nodes and their weights

EXAMPLE 2.52 (Hypergraph). Let $H = \langle \{v_1, v_2, v_3, v_4\}, \{e_1, e_2, e_3, e_4, e_5\} \rangle$ be a hypergraph with

$$
\begin{aligned}
e_1 &= \langle (), v_1 \rangle & f_{e_1}() &= 1 \\
e_2 &= \langle (), v_2 \rangle & f_{e_2}() &= 2 \\
e_3 &= \langle (), v_3 \rangle & f_{e_3}() &= 3 \\
e_4 &= \langle (v_1), v_2 \rangle & f_{e_4}(x_1) &= x_1 + 4 \\
e_5 &= \langle (v_2, v_3), v_5 \rangle & f_{e_5}(x_1, x_2) &= x_1 + x_2 + 5 \quad .
\end{aligned}
$$

A graphical representation of H is given in figure 13. The derivations (with backpointers) of the nodes in H and their weights are given in the table next to it. v_2 has two derivations, one via e_2 and one via e_1 and e_4. Accordingly, v_4 also has two derivations, both via e_5, one which points back to the 1th-best derivation of v_2 and one which points back to the 2nd-best derivation of v_2. □

Given a hypergraph $H = \langle V, E \rangle$, the derivation of the GOAL item with the best weight can be found using a *generic Viterbi algorithm*. See the pseudocode in algorithm 3. It traverses the hypergraph in *topological order*, i. e., in any order in which, for all nodes $v \in V$, all tail nodes of v are visited before v. For each node v, its best derivation $\hat{D}_1(v)$ is found by considering the best derivation $\langle e, \mathbf{1} \rangle$ along each incoming hyperedge e. $\mathbf{1}$ denotes a vector whose elements are all 1, and we assume that is has the appropriate size, i. e. $|\mathbf{1}| = |\boldsymbol{T}(e)|$ in this case.

The creation of the hypergraph H is not part of this generic algorithm. In the context of parsing, it can be created by using a deduction

```
1: ⟨V, E⟩: a directed, ordered acyclic hypergraph for w with a GOAL node
2: function VITERBI(⟨V, E⟩)
3:     for v ∈ V in topological (bottom-up) order do
4:         for e ∈ E where H(e) = v do          ▷ For all incoming hyperedges
5:             D̂1(v) = max⪯(D̂1(v), ⟨e, 1⟩)                          ▷ Update
6:         end for
7:     end for
8:     return D̂1(GOAL)
9: end function
```

Algorithm 3: Generic Viterbi algorithm

system and a slightly extended version of algorithm 1, which construct a hyperedge between each consequent and its antecedent items (in analogy to backpointers). H is either constructed in advance, or dynamically if the topological order of the nodes is known a priori. For a bottom-up CYK-style parser, this roughly means that items covering n words are created/visited before items covering m words, for all $n < m$. Note that then algorithm 3 coincides with algorithm 2.

For *k-best parsing*, Huang and Chiang (2005) present two efficient algorithms, called *algorithm 2* and *3* in the original paper. Following Williams et al. (2016), we will refer to them as the *eager* and *lazy k-best algorithm* respectively. Even though the lazy algorithm is more favorable in terms of time complexity, we still present the eager algorithm in detail, as it serves as the basis for cube pruning, which will be discussed later in this thesis (see sections 2.2.5 and 5.3.2).

EXAMPLE 2.53 (k-best parsing). To illustrate the main idea of k-best parsing, consider a node v, somewhere in a hypergraph, for which we wish to compute the k-best derivations. Let's say it has one incoming hyperedge e with $\boldsymbol{T}(e) = (v_1, v_2)$ where v_1 and v_2 themselves might have many derivations. See figure 14 for the graphical representation. Suppose that we have already found the k-best derivations for each v_1 and v_2. Then the 1-best derivation of v is constructed from the 1-best derivations of v_1 and v_2: $\langle e, (1,1)\rangle$. This is exactly what the Viterbi algorithm would do. For the 2-best derivation of v, we have two candidates:

- $\langle e, (1,2)\rangle$ containing the 1-best derivation of v_1 and the 2-best derivation of v_2, or

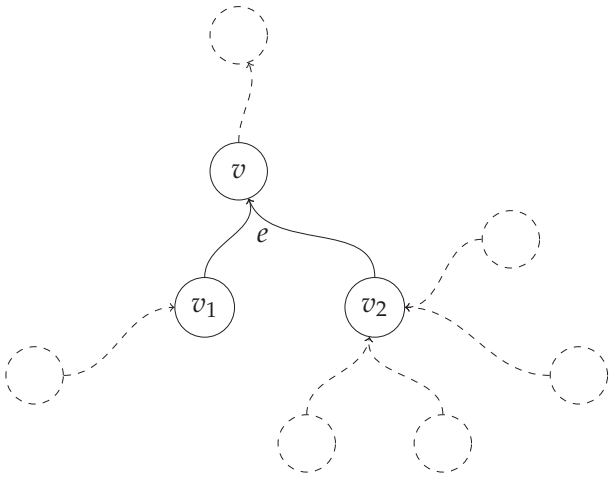


Figure 14: Hypergraph fragment for ex. 2.53

- $\langle e, (2,1) \rangle$ containing the 2-best derivation of v_1 and the 1-best derivation of v_2.

This follows from the fact that the weight function is monotonic. Suppose that $\langle e, (1,2) \rangle$ has the larger weight of the two. It thus constitutes the 2-best derivation of v. For the 3-best derivation of v, $\langle e, (2,1) \rangle$ is still a candidate. In addition, there are two more candidates, the *neighbors* of the 2-best derivation $\langle e, (1,2) \rangle$: $\langle e, (2,2) \rangle$ and $\langle e, (1,3) \rangle$. □

The eager k-best parsing algorithm is presented as algorithm 4. Standard implementations of the list procedure APPEND and the priority queue procedures PUSH and POPMAX (e. g. in Cormen et al. (2001)) are assumed here and in related algorithms. Furthermore, with $\mathbf{1}$, we again denote a vector whose elements all have the value 1, and with $\boldsymbol{b}^i$, a vector whose elements are all 0 except for the ith element which is 1.

In algorithm 4, the procedure lined out in ex. 2.53 is executed until the k-best derivations of v have been found (or no candidates are available anymore, line 13). The candidates are organized in a priority queue *cand*, sorted by their weight according to $\preceq$. In each iteration, the candidate with the maximal weight is popped from *cand*, appended to the list of best derivations of v and new candidates, its neighbors, are added to the priority queue (lines 14 to 16). The computation of the neighboring derivations is given in lines 26 to 33. They are the next best derivations of each of the tail nodes, exploiting the

```
1: ⟨V, E⟩: a directed, ordered acyclic monotonic hypergraph for w with a
   GOAL node
2: k ∈ ℕ
3: function KBEST(⟨V, E⟩, k)
4:     for v ∈ V in topological (bottom-up) order do
5:         KBEST(v, k)
6:     end for
7:     return D̂(GOAL)
8: end function
9:
10: procedure KBEST(v, k)
11:     cand = ∅                                          ▷ Priority queue
12:     GETFIRSTBEST(cand, v, k)                ▷ Initialize the priority queue
13:     while |D̂(v)| < k and |cand| > 0 do
14:         d = POPMAX(cand)
15:         APPEND(D̂(v), d)
16:         PUSHNEIGHBORS(d, cand)
17:     end while
18: end procedure
19:
20: procedure GETFIRSTBEST(cand, v)
21:     for e ∈ E where H(e) = v do              ▷ For all incoming hyperedges
22:         PUSH(cand, ⟨e, 1⟩)
23:     end for
24: end procedure
25:
26: procedure PUSHNEIGHBORS(⟨e, x⟩, cand)
27:     for all i s.t. 1 ≤ i ≤ |T(e)| do
28:         x′ = x + b^i
29:         if x′_i ≤ |D̂(T_i(e))| and ⟨e, x′⟩ has not been seen before then
30:             PUSH(cand, ⟨e, x′⟩)
31:         end if
32:     end for
33: end procedure
```

Algorithm 4: Eager k-best parsing algorithm from Huang and Chiang (2005)

fact that the tail nodes as well as the k-best derivations of each node are ordered vectors. To accommodate for more than one incoming hyperedge, *cand* is initialized with the 1th-best derivation along each hyperedge (lines 20 to 24).

It is the monotonicity of the hypergraph $\langle V, E \rangle$ which guarantees that the derivations which are popped from *cand* are in the correct order and constitute the true k-best derivations.

While the eager k-best algorithm computes the k-best derivations for *each* $v \in V$ in a bottom-up fashion, the lazy k-best algorithm operates in a top-down fashion. It starts by generating the k-best derivations of the GOAL item and only calls itself recursively if necessary. The pseudocode of the lazy algorithm is given as algorithm 5.

EXAMPLE 2.54. As an example, consider the setup from ex. 2.53, p. 46, and $k = 5$. For instance, the 4th-best derivation of v_1 would only be considered and assigned a weight if the 3rd-best derivation of v_1 is part of a candidate which has been taken from *cand* of v. This might however never actually happen, depending on the weights of the derivations of v_2. □

For details on the complexity characteristics of the two algorithms, consult Huang and Chiang (2005).

2.1.3 *Modeling the Syntax of Natural Languages*

This section discusses the modeling of the syntax of natural languages with formal grammars. Its content is based on the presentation of similar material by L. Kallmeyer and W. Maier (e. g. Kallmeyer, 2010, ch. 1, ch. 2; Maier, 2013, sec. 1.1).

Natural Languages and Formal Grammars

The two main factors which generally influence the decision of a computational linguist for a specific grammar formalism are its expressivity and its computational complexity. The expressivity of a grammar formalism usually designates its weak or strong generative capacity. The *weak generative capacity* refers to the class of languages that it can generate, while the *strong generative capacity* refers to the class of syntactic structures which are induced. Ideally, the chosen formalism should be able to generate all and only the grammatical sentences of

```
1: ⟨V, E⟩: a directed, ordered acyclic monotonic hypergraph for w with a
   GOAL node
2: k ∈ ℕ
3: function LAZYKBEST(⟨V, E⟩, k)
4:     LAZYJBEST(GOAL, k)                          ▷ Start with the GOAL item
5:     return D̂(GOAL)
6: end function
7:
8: procedure LAZYJBEST(v, j)
9:     if cand[v] is not defined yet then
10:        GETFIRSTBEST(cand[v], v)        ▷ Initialize the priority queue, alg. 4
11:    end if
12:    while |D̂(v)| < j do
13:        if |D̂(v)| > 0 then
14:            d = D̂_{|D̂(v)|}(v)                          ▷ Get last derivation
15:            LAZYPUSHNEIGHBORS(d, cand[v])               ▷ and update cand
16:        end if
17:        if |cand[v]| = 0 then
18:            break
19:        end if
20:        d =POPMAX(cand[v])                  ▷ Get the next best derivation
21:        APPEND(D̂(v), d)
22:    end while
23: end procedure
24:
25: procedure LAZYPUSHNEIGHBORS(⟨e, x⟩, cand)
26:    for all i s.t. 1 ≤ i ≤ |T(e)| do
27:        x′ = x + b^i
28:        LAZYJBEST(T_i(e), x′_i)                                  ▷ Recursion
29:        if x′_i ≤ |D̂(T_i(e))| and ⟨e, x′⟩ has not been seen before then
30:            PUSH(cand, ⟨e, x′⟩)
31:        end if
32:    end for
33: end procedure
```

Algorithm 5: Lazy k-best parsing algorithm from Huang and Chiang (2005)

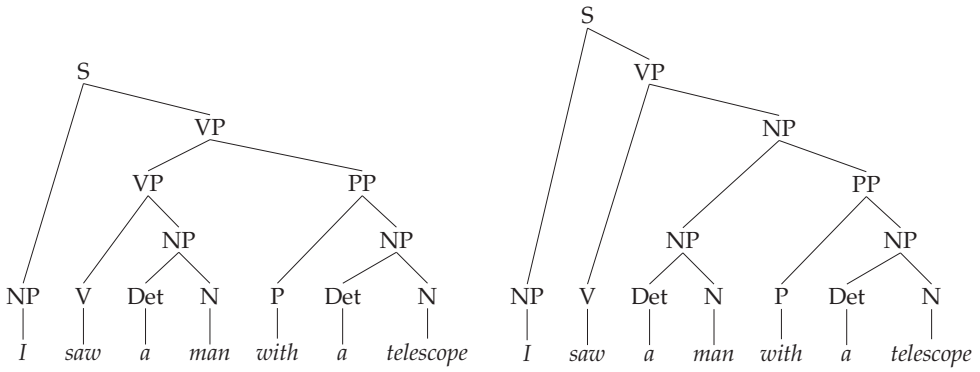


Figure 15: CFG parse trees

a language, producing all and only the syntactic structures which are consistent with the implemented linguistic theory. Usually, computational linguists are a lot less concerned with overgeneration, i. e. the generation of ungrammatical sentences or syntactic structures which do not conform with the linguistic theory, than with undergeneration. This holds in particular in probabilistic, data-driven contexts where weights are used to disambiguate between competing analyses. Commonly, the more expressive a grammar formalism, the more computationally complex is the task of recognizing or parsing a string. In practical applications, the choice for a grammar formalism therefore involves a trade-off between expressivity and tractability.

Context-Free Grammar (CFG) was proposed as a first formalism for syntactic description in the constituency-based framework (Chomsky, 1956). By definition, CFG constituents are continuous, i. e. they consist of a continuous sequence of words of the sentence (cf. def. 2.17, p. 22). Figure 15 shows two CFG analyses for the ambiguous English sentence *I saw a man with a telescope*. Syntactic theories based on CFG, including work which extends the base formalism with transformations (Chomsky, 1956) or features (Gazdar et al., 1985), have been used to model a wide range of linguistic phenomena.

However, in the 80's, it was shown that CFGs are not powerful enough to generate all phenomena of natural languages (Bresnan et al., 1982; Shieber, 1985). The examples which have been discussed to exceed the generative power of CFG are *cross-serial dependencies*. Those are structures in which constituents are intertwined in a way that material from one constituent interrupts the other constituent and vice

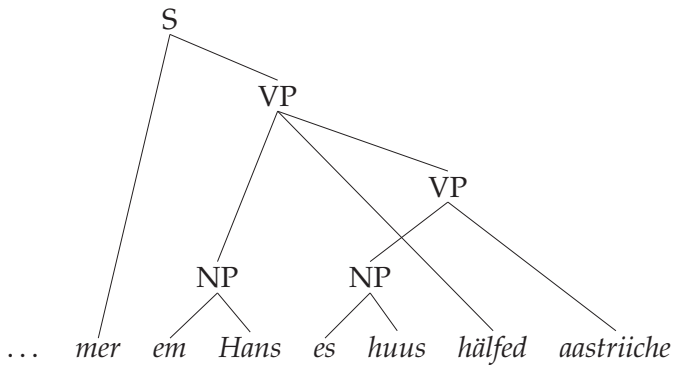


Figure 16: Constituent tree for (4)

versa. One such example from Dutch has already been presented in chapter 1, see (2) and figure 7. Another one from Shieber (1985) is the Swiss-German sentence in (4). The co-indexation denotes that the noun phrase is an argument of the correspondingly indexed verb. To adequately model this dependency, each verb and its co-indexed argument would form one constituent, thus giving rise to interwined constituents. A potential syntactic constituency analysis of (4) is shown in figure 16. Notice how the VPs are intertwined, and that the yield of the lower VP is *discontinuous*. The phenomenon can be iterated, as shown in (5), and is in principle unbounded. Shieber (1985) proves that the strong as well as the weak generative capacity of CFG is too limited to model the Swiss-German cross-serial dependencies.

(4) mer em Hans_i es huus_j hälfed_i aastriiche_j
we Hans_{DAT} $\text{the house}_{\text{ACC}}$ help paint
'[. . . that] we help Hans paint the house'

(5) mer d'chind_i em Hans_j es huus_k lönd_i hälfe_j aastriiche_k
we $\text{the kids}_{\text{ACC}}$ Hans_{DAT} $\text{the house}_{\text{ACC}}$ let help paint
'[. . . that] we let the kids help Hans paint the house'

Besides the rather particular cross-serial dependencies of Dutch and Swiss-German, generally, no *non-local* or *long-distance dependencies* can be described with CFG. Those are structures which consist of several elements which syntactically belong together, but are separated by

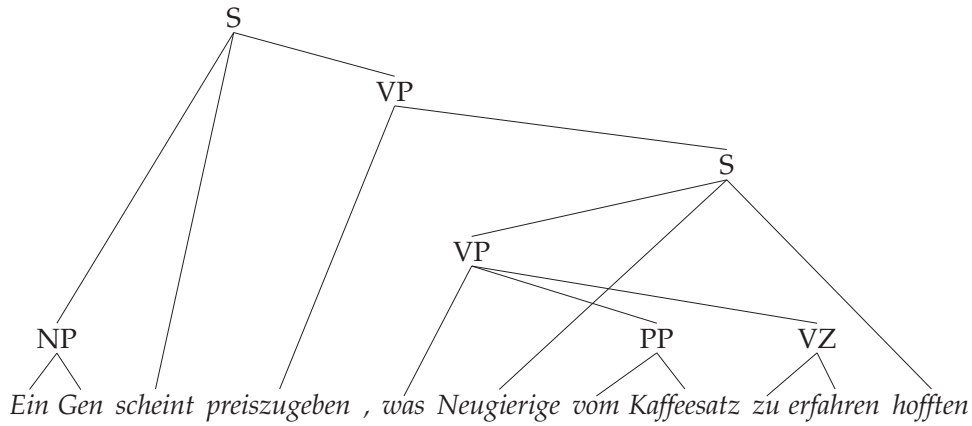


Figure 17: Constituent tree for (6) as provided by the Tiger Treebank

some intervening material. Non-local dependencies are frequent in languages with a rather free word order, such as German, but they also occur in languages with a more rigid word order, such as English.

As a first example consider the relative clause in the German sentence (6). Figure 17 shows the constituency tree as annotated within the Tiger Treebank (Brants et al., 2002).[9] The direct object (*was*) of the embedded verb *erfahren* is realized as a relative pronoun and thus fronted. The lower VP node, which also dominates a PP modifier in addition to the verb, has a gap, which is filled with the subject (*Neugierige*) of the main verb. It is a discontinuous constituent.

(6) Ein Gen scheint preiszugeben , *was* Neugierige *vom*
A gene seems to reveal , what curious people from the
Kaffeesatz *zu erfahren* hofften
coffee grounds to learn hope
'A gene appears to reveal what curious people had hoped to read from the tea leaves'

Sentence (7) demonstrates that non-local dependencies also occur in English. The PP modifier of the subject has been extraposed, leading to

9 A lean version of the constituency structure provided by the Tiger Treebank is reproduced here. Specifically, the part-of-speech nodes and the edge labels as well as some modifiers are omitted for clarity of presentation.

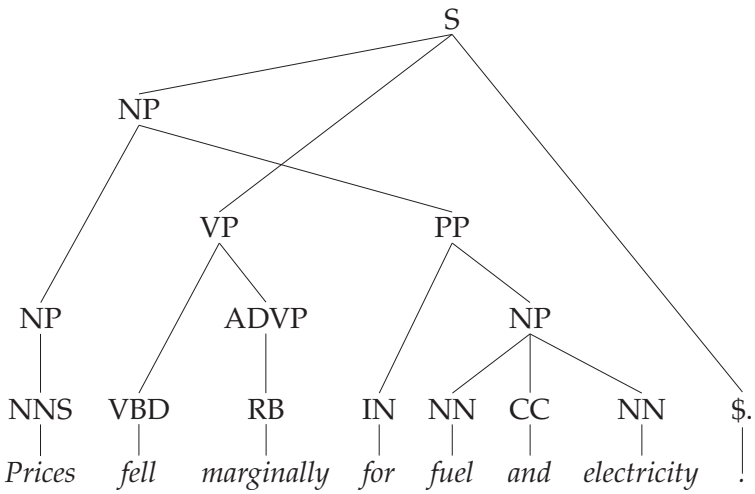


Figure 18: Constituent tree for (7) as annotated by the Penn Treebank

a discontinuous NP. The sentence and its syntactic analysis in figure 18 are taken from the Penn Treebank (Marcus et al., 1993).[10]

(7) *Prices* fell marginally *for fuel and electricity*.

Discontinuous constituents (def. 2.13, p. 20) in syntactic constituency structures such as in figure 7, 16, 17 and 18 are easy to identify by looking for *crossing branches*. For instance, the edge from S to VP in figure 18 crosses into the yield of the (discontinuous) NP. Besides directly representing discontinuous constituents, as we have done so far and as it is done in the German Negra and Tiger treebanks (Skut et al., 1997; Brants et al., 2002), alternative strategies may be chosen. For instance, in the Penn Treebank annotation scheme, trace nodes and co-indexation are used to indicate non-local dependencies which exceed context-free trees (cf. footnote 10). Generally, phenomena whose analyses involve discontinuities, such as cross-serial dependencies, cannot be described using CFGs only.

To model discontinuous constituents with a formal grammar, a formalism beyond the expressivity of CFG is required. In order to char-

10 The annotation scheme of the Penn Treebank does not directly annotate discontinuous constituents. Instead, context-free trees are annotated, and non-local dependencies are described using a co-indexation mechanism (Marcus et al., 1994). However, discontinuous constituents can be recovered from that annotation, as done in Evang and Kallmeyer (2011). The example has been taken from the latter paper.

acterize the formal properties of a grammar formalism that can adequately describe natural languages, Joshi (1985) introduced the notion of *mild context-sensitivity*. The class of mildly context-sensitive languages contains at least the context-free languages, it allows a limited amount of cross-serial dependencies, its languages can be parsed in polynomial time, i. e. the recognition problem is tractable, and the lengths of the words of its languages grow in a linear way. A grammar formalism is called *mildly context-sensitive* if the set of all languages which can be generated with grammars of that formalism is mildly context-sensitive.

Grammar Formalisms beyond CFG

As it has become clear that CFG is not enough for the modeling of natural language syntax, extensions thereof which fall into the category of mildly context-sensitive formalisms have been proposed. One of the weakest extensions is Tree-Adjoining Grammar (TAG), originally proposed in Joshi et al. (1975). The basic units of a TAG are so-called *elementary trees*, templates for syntactic structure which, when implementing natural language grammars, encode a grammatical relationship, e. g., between a verb and its arguments. Starting from such an elementary tree, larger trees are derived by replacing a leaf node with another elementary tree (*substitution*) or replacing an internal node with a special kind of elementary tree, an *auxiliary tree* (*adjunction*). Figure 19 shows a sample derivation. The tree for *Peter* is substituted at the NP subject slot of the elementary tree of *laughs*. The auxiliary tree for the modifier *often* is adjoined at the VP node.

TAGs are more expressive than CFGs due to their extended domain of locality and the factoring of recursion which is contributed by the adjunction operation. The elementary trees may be arbitrarily large, and even elements which are local in one elementary tree, e. g. the verb and its subject, can be arbitrarily far apart in the final derived tree given that other lexical material is adjoined between them. Nevertheless, TAGs are still not expressive enough to model certain phenomena in a linguistically adequate way, e. g. certain extraction and scrambling phenomena (Becker et al., 1991; Kahane et al., 2000). Accordingly, more expressive variants of TAG have been developed, e. g. Multi-Component Tree-Adjoining Grammar (MCTAG).

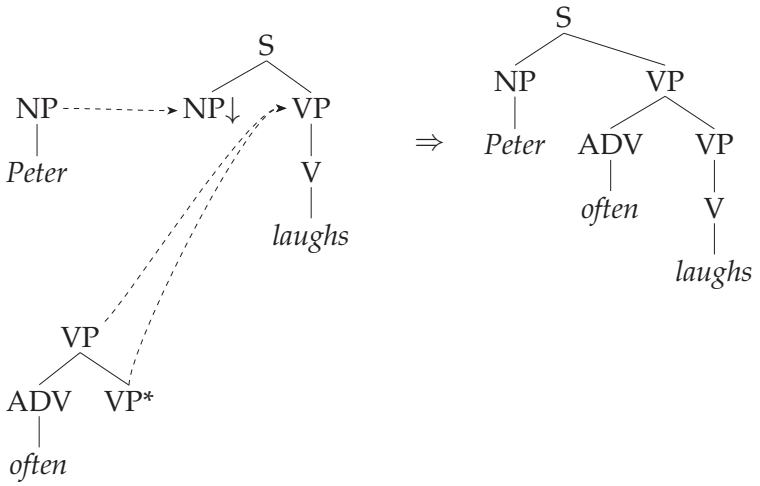


Figure 19: TAG derivation

TAG is a popular formalism amongst (computational) linguists. It has been shown that a wide range of linguistic phenomena can be adequately handled with TAG (e. g. in Abeillé (1988)). TAG and its variants have also been used to implement grammars of varying coverage for several languages, e. g. the XTAG grammar for English (XTAG Research Group, 2001). Various parsing algorithms for TAG have been proposed (e. g. Earley-style parsing by Schabes and Joshi (1988)), and work has been dedicated to the extraction of wide-coverage grammars from treebanks for statistical parsing (e. g. in Xia (2001) and Kaeshammer and Demberg (2012)).

Linear Context-Free Rewriting System (LCFRS) (Vijay-Shanker et al., 1987) (def. 2.29, p. 26) is a mildly context-sensitive grammar formalism which is more powerful than TAG.[11] LCFRS is furthermore a somewhat more natural extension to CFG than TAG as it also uses rewriting rules. In LCFRS, non-terminals span tuples of strings, instead of just strings, as CFG non-terminals do. Accordingly, LCFRS lends itself very well for the modeling of discontinuous constituents. The trees that are generated by grammars of LCFRS and related formalisms exhibit *crossing branches* in the case of discontinuous constituents. The syntactic structures in figures 16, 17 and 18 can be considered as possible LCFRS parse trees of (4), (6) and (7) respectively.

11 Another formalism which is very similar to LCFRS and weakly equivalent is Multiple Context-Free Grammar (MCFG) (Seki et al., 1991).

It has been argued that there exist linguistic phenomena which are beyond the generative capacity of LCFRS. They are however rather of theoretical interest, as LCFRS can nevertheless generate the syntactic structures in question, though at the expense of potentially missing generalizations. For instance, Becker et al. (1992) argue that scrambling in German is in principle unbounded in the number of arguments that are scrambled and the distance over which arguments are scrambled, and show that LCFRS is too restricted for the modeling of this phenomena. However, when assuming scrambling to be bounded, the required structures for scrambling can well we generated.

In recent years, LCFRS has received attention from the data-driven parsing community. Due to the direct correspondence between syntactic structures annotated in constituency treebanks and the LCFRS parse trees, probabilistic LCFRS can be extracted in a natural way (Maier and Søgaard, 2008), and data-driven LCFRS parsing has been shown to be feasible (Maier, 2010; Evang and Kallmeyer, 2011; van Cranenburgh, 2012; Maier et al., 2012; Kallmeyer and Maier, 2013). See section 2.1.4 for more details. LCFRS is also of interest for data-driven grammar-based dependency parsing (Kuhlmann and Satta, 2009; Maier and Kallmeyer, 2010).

Range Concatenation Grammar (RCG) (Boullier, 1998, 2000) is another interesting formalism beyond CFG, which is however not mildly context-sensitive. In contrast to LCFRS, RCG has a copying (and a deleting) mechanism. Intuitively, the copying allows a substring of the input to be part of several constituents which do not dominate each other. It is therefore possible to generate syntactic structures like the one in figure 20. It shows an example of right node raising. The parse tree expresses that the noun phrase *broccoli* is the object of both VPs.

RCG is an interesting formalism in many aspects. It can potentially model scrambling and other phenomena beyond LCFRS (Boullier, 1999). Unlike the languages of CFGs, range concatentation languages are closed under intersection. RCG has therefore been argued to be more modular than other formalisms, since different aspects of language can be modeled independently.[12] RCG has been used as the base formalism for grammar implementation (Sagot, 2005). Even though RCG is not mildly context-sensitive, RCG parsing is still relatively tractable. Published parsing algorithms are polynomial in the

12 Chiang (2004) questions the usefulness of this property and also the account provided for scrambling.

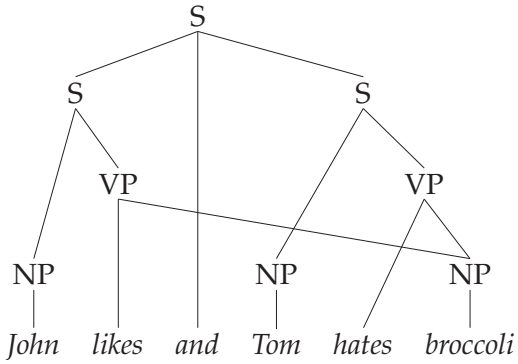


Figure 20: RCG parse tree, reproduced after Kallmeyer (2010, p. 5)

input size, but in practice the cost for the instantiation of the rewriting rules should not be neglected (Boullier, 1998; Bertsch and Nederhof, 2001; Maier, 2013, sec. 3.1). A restricted form of RCG, SRCG, is equivalent to LCFRS, and the RCG notation provides an attractive syntactic variant to original LCFRS notations.

To close this section, figure 21 shows a schematic representation of the domains of locality of the grammar formalisms presented and the resulting tree structures. In a CFG, only one continuous string at a time can be inserted into the yield of a non-terminal (β for B). Due to the adjunction operation, TAG allows for the insertion of two non-adjacent strings simultaneously (β_1 and β_2). In an LCFRS, arbitrarily many strings can be inserted. They then actually interrupt the yield of other non-terminals, leading to crossing branches. In figure 21, the yield of A is interrupted by β and γ. Node A neither dominates C nor D. RCG allows for the same kind of interruption as LCFRS. In addition, yields of non-terminals may overlap. α_3 belongs to constituent A as well as C.

2.1.4 *Parsing Discontinuous Constituents*

In NLP, nowadays usually *data-driven parsing* is performed. This means that the parsing models, e. g. the grammar rules together with their probabilities or weights, are obtained from *treebanks*, as opposed to manually designing grammar rules. Treebanks are collections of sentences which are annotated with syntactic trees.

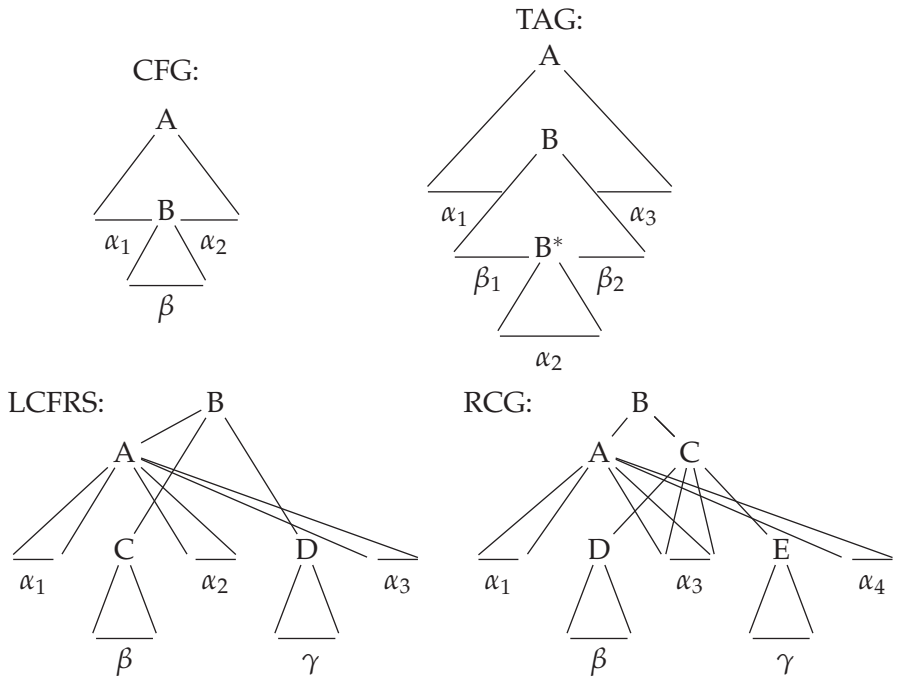


Figure 21: Locality in different grammar formalisms; adapted from Kallmeyer (2010, p. 3)/Maier (2013, p. 10)

Treebanks and CFG Parsing

Even though it was known that CFG is not an adequate formalism for the modeling of natural language syntax (cf. section 2.1.3), a large portion of the published literature on (data-driven) natural language parsing concentrates on CFG parsing.

To extract a (weighted) CFG from a treebank for data-driven parsing, CFG parse trees are required. Some treebanks, such as the English Penn Treebank (Marcus et al., 1993), provide such trees. If an additional mechanism is used to annotate non-local dependencies, e. g. the trace and co-indexation annotation of the Penn Treebank (see p. 54 and Marcus et al. (1994)), this additional information is usually discarded for data-driven CFG parsing. Other constituency treebanks, such as the German Negra (Skut et al., 1997) and Tiger (Brants et al., 2002) treebanks, allow crossing branches for non-local dependencies and group all parts of a discontinuous constituent under one node. For data-driven CFG parsing, the crossing branches need to be resolved in some way such that the trees are CFG parse trees. Usually, an undocumented implementation shipped with Negra is used for that purpose. It first identifies head words and then moves non-head constituents higher up in the tree.[13] In Maier et al. (2012), we first reduce the number of spurious discontinuous constituents by attaching punctuation low in the tree instead of to the root node and applying a series of linguistically motivated transformations, before using the standard method to resolve crossing branches. In Boyd (2007), an additional node is inserted into the tree for each block of a discontinuous constituent when resolving the crossing branches.

A consequence of the described techniques is that information about non-local dependencies is lost with CFG parsing, even though the corresponding annotation might have been present in the treebank. To restore the non-local dependencies annotated in the treebank, one line of work resorts to more sophisticated pre-and post-processing, and augments a weighted CFG parser with additional machine learning techniques (e. g. Johnson, 2002; Dienes and Dubey, 2003; Levy and Manning, 2004). Another possibility is to use a more expressive grammar formalism than CFG which is able to represent the non-local information directly in the parse trees, for instance by means of discontinuous

13 See Maier (2013, sec. 7.2) for a detailed reverse-engineered description of the algorithm.

constituents. Some options for such formalisms have been discussed in section 2.1.3 (p. 55ff.), and LCFRS parsing will be the topic of the upcoming section.

LCFRS Parsing

For the direct parsing of discontinuous constituents, LCFRS and its equivalents (def. 2.29, p. 26) have become popular candidates. The reasons include the following:

- Treebank trees with crossing branches can be directly interpreted as LCFRS parse trees. Grammar extraction is thus a straightforward procedure.
- The proximity of LCFRS to CFG allows the direct extension of (data-driven) CFG parsing techniques to LCFRS.
- LCFRS is polynomially parsable, and therefore its processing is fairly efficient.

We will first sketch grammar extraction, then provide a deduction system for LCFRS parsing and finally provide an overview of applied parsing techniques.

For grammar extraction, the treebank trees with crossing branches are interpreted as LCFRS derivations. LCFRS rules are extracted from each such constituency structure. They are counted for the maximum likelihood estimation of the probabilities, and all rules together constitute the grammar. The rule extraction itself roughly works as follows.[14] For each node v_0 with a single child v_1 such that $l(v_0) \in N$ and $l(v_1) \in T$, a terminal rule of the form $l(v_0)(l(v_1)) \to \varepsilon$ is created. For each node v_0 with children $v_1, \ldots, v_n$ with $n \geq 1$ and $l(v_i) \in N$ for $0 \leq i \leq n$, a rule of the form $l(v_0)(\boldsymbol{\alpha}_0) \to l(v_1)(\boldsymbol{\alpha}_1) \ldots l(v_n)(\boldsymbol{\alpha}_n)$ is created. The number of arguments of each non-terminal (i. e. the length of each $\boldsymbol{\alpha}_i$) is the number of continuous blocks in the yield of the corresponding node. $\boldsymbol{\alpha}_0$ contains variables that explain how the yield of v_0 is obtained from the yields of $v_1, \ldots, v_n$. The algorithm originally has

14 Usually, a treebank tree is a special type of constituency structure: each internal node either has only children labeled with elements from the syntactic categories N, or exactly one child which is labeled with a lexical element from T. The rule extraction procedure only works for this kind of treebank trees. In particular, it does not extract any mixed rules.

been presented in Maier and Søgaard (2008). A detailed description is also given in Kallmeyer and Maier (2013, sec. 5.1).

EXAMPLE 2.55 (LCFRS grammar extraction). Consider figure 18 as a syntactic analysis provided in a treebank. The following LCFRS rules are extracted:

r_0 : $\mathrm{S}(X_1X_2X_3X_4) \rightarrow \mathrm{NP}(X_1, X_3)\ \mathrm{VP}(X_2)\ \$.(X_4)$
r_1 : $\mathrm{NP}(X_1, X_2) \rightarrow \mathrm{NP}(X_1)\ \mathrm{PP}(X_2)$
r_2 : $\mathrm{PP}(X_1X_2) \rightarrow \mathrm{IN}(X_1)\ \mathrm{NP}(X_2)$
r_3 : $\mathrm{VP}(X_1X_2) \rightarrow \mathrm{VBD}(X_1)\ \mathrm{ADVP}(X_2)$
r_4 : $\mathrm{NP}(X_1X_2X_3) \rightarrow \mathrm{NN}(X_1)\ \mathrm{CC}(X_2)\ \mathrm{NN}(X_3)$
r_5 : $\mathrm{NP}(X_1) \rightarrow \mathrm{NNS}(X_1)$
r_6 : $\mathrm{ADVP}(X_1) \rightarrow \mathrm{RB}(X_1)$
r_7 : $\mathrm{NNS}(\mathit{Prices}) \rightarrow \varepsilon$
r_8 : $\mathrm{VBD}(\mathit{fell}) \rightarrow \varepsilon$
r_9 : $\mathrm{RB}(\mathit{marginally}) \rightarrow \varepsilon$
r_{10} : $\mathrm{IN}(\mathit{for}) \rightarrow \varepsilon$
r_{11} : $\mathrm{NN}(\mathit{fuel}) \rightarrow \varepsilon$
r_{12} : $\mathrm{CC}(\mathit{and}) \rightarrow \varepsilon$
r_{13} : $\mathrm{NN}(\mathit{electricity}) \rightarrow \varepsilon$
r_{14} : $\$.(.) \rightarrow \varepsilon$

Note how the discontinuous NP node leads to rule r_1 whose LHS nonterminal has two arguments. Rule r_0 describes how the two parts of the NP node wrap around the yield of the VP. □

For LCFRS parsing, a CYK algorithm can be used. The corresponding symbolic parser goes back to Seki et al. (1991). Probabilistic parsers for LCFRS have been described as a straightforward extension thereof and implemented. W. Maier and colleagues present a chart parser based on probabilistic LCFRS (Maier, 2010; Kallmeyer and Maier, 2013). It is used for treebank parsing, i. e. the applied grammars are extracted from treebanks on the basis of the aforementioned approach of Maier and Søgaard (2008). A. van Cranenburgh and colleagues extend this approach to parsing with discontinuous tree fragments within the Data-Oriented Parsing (DOP) framework (van Cranenburgh et al., 2011; van Cranenburgh and Bod, 2013; van Cranenburgh et al., 2016). The parsing results are competitive to probabilistic CFG parsers (avail-

able at the respective time) while providing richer syntactic descriptions.

We present the CYK parser for LCFRS as a weighted deduction system. Let $G = (N, T, V, P, S, \omega)$ be a weighted LCFRS (def. 2.41, p. 33) and let $w = w_1 \dots w_n$ be the input string. The items have the form $[A, \boldsymbol{\rho}]$ where $A \in N$ is a non-terminal symbol and $\boldsymbol{\rho}$ is a $dim(A)$-dimensional range vector for w (def. 2.32, p. 27), i. e. $\boldsymbol{\rho} \in (Pos(w) \times Pos(w))^{dim(A)}$. The range vector $\boldsymbol{\rho}$ characterizes the ranges in w which are derived by A, i. e. $A(\boldsymbol{\rho}) \stackrel{*}{\Rightarrow}_{G,w} \varepsilon$.

The SCAN deduction rule creates items for terminal rules:

$$\frac{}{\omega(r) : [A, \boldsymbol{\rho}]}$$

with the following side conditions:

1. $r = A(\boldsymbol{\alpha}) \to \varepsilon$ and $r \in P$,
2. there exists a ϕ s.t. $\phi(A(\boldsymbol{\alpha})) = A(\boldsymbol{\rho})$ where ϕ is an instantiation of r with respect to w.

Note that by definition of the instantiation it holds that $\boldsymbol{\rho}(w) = \boldsymbol{\alpha}$.

The COMPLETE operation combines already established items to a new item:

$$\frac{x_1 : [A_1, \boldsymbol{\rho}_1] \ \dots \ x_m : [A_m, \boldsymbol{\rho}_m]}{x_1 \cdot \ldots \cdot x_m \cdot \omega(r) : [A, \boldsymbol{\rho}]}$$

with the following side conditions:

1. $r = A(\boldsymbol{\alpha}) \to A_1(\boldsymbol{\alpha}_1) \dots A_m(\boldsymbol{\alpha_m})$ and $r \in P$,
2. there exists an instantiation ϕ for r s.t. $A(\boldsymbol{\rho}) \to A_1(\boldsymbol{\rho}_1) \dots A_m(\boldsymbol{\rho}_m)$ is the corresponding instantiated rule with respect to w.

The GOAL item is an item which covers the complete input:

$$x : [S, \langle\langle 0, n \rangle\rangle]$$

The parse items and the parse hypergraph for w can be computed, for example, by using the generic chart parsing algorithm (algorithm 1). With the resulting parse hypergraph H, the best or the k-best

No.	Item	Rule	Antecedents	Weight
1	$[\mathrm{A}, \langle\langle 0,1\rangle, \langle 3,4\rangle\rangle]$	SCAN with r_5	-	0.1
2	$[\mathrm{A}, \langle\langle 0,1\rangle, \langle 4,5\rangle\rangle]$	SCAN with r_5	-	0.1
3	$[\mathrm{A}, \langle\langle 1,2\rangle, \langle 3,4\rangle\rangle]$	SCAN with r_5	-	0.1
4	$[\mathrm{A}, \langle\langle 1,2\rangle, \langle 4,5\rangle\rangle]$	SCAN with r_5	-	0.1
5	$[\mathrm{B}, \langle\langle 2,3\rangle, \langle 5,6\rangle\rangle]$	SCAN with r_6	-	0.1
6a	$[\mathrm{A}, \langle\langle 0,2\rangle, \langle 3,5\rangle\rangle]$	COMPLETE with r_2	4	0.07
6b		COMPLETE with r_3	2	0.02
7	$[\mathrm{S}, \langle\langle 0,6\rangle\rangle]$	COMPLETE with r_1	5 & 6a	0.007

Table 2: Parse items for ex. 2.56

derivations can be found in the same fashion as for CFG (algorithms 3 and 4 or 5). Similarly, if a partial order on the items is guaranteed, then the generic Viterbi chart parsing algorithm in algorithm 2 can be applied to find the best derivation for w.

EXAMPLE 2.56 (LCFRS Parsing). Let $G = (\{\mathrm{S}, \mathrm{A}, \mathrm{B}\}, \{\mathrm{a}, \mathrm{b}, \mathrm{c}, \mathrm{d}\}, \{Y_1, Y_2, Y_3, Y_4\}, P, \mathrm{S}, \omega)$ be a weighted LCFRS with P and ω as follows:

i	r_i		$\omega(r_i)$
1	$\mathrm{S}(Y_1Y_2Y_3Y_4)$	$\to \mathrm{A}(Y_1, Y_3)\ \mathrm{B}(Y_2, Y_4)$	1.0
2	$\mathrm{A}(\mathrm{a}Y_1, \mathrm{c}Y_2)$	$\to \mathrm{A}(Y_1, Y_2)$	0.7
3	$\mathrm{A}(Y_1\mathrm{a}, \mathrm{c}Y_2)$	$\to \mathrm{A}(Y_1, Y_2)$	0.2
4	$\mathrm{B}(\mathrm{b}Y_1, \mathrm{d}Y_2)$	$\to \mathrm{B}(Y_1, Y_2)$	0.9
5	$\mathrm{A}(\mathrm{a}, \mathrm{c})$	$\to \varepsilon$	0.1
6	$\mathrm{B}(\mathrm{b}, \mathrm{d})$	$\to \varepsilon$	0.1

As an example, consider table 2 which shows the items that are created when parsing $w = \mathrm{aabccd}$ with G using the generic Viterbi chart parsing algorithm in algorithm 2. □

The complexity of the CYK parser for LCFRS is polynomial in the input size n: $\mathcal{O}(n^{v(u+1)})$ for G being a (u, v)-LCFRS. For a justification, let us consider again the most complex rule of the deduction system, the COMPLETE rule. The most complex case within the COMPLETE rule is a rewriting rule r which has rank $m = u$ and where each non-

terminal has fan-out v. Such a rule provides $u \cdot v$ variables in the RHS which have to be combined in the arguments of the LHS non-terminal. Consider the case where the LHS arguments do not contain any terminals that constrain the ranges of the variables. If two variables Y_1 and Y_2 are adjacent in an argument, they share one index, i. e. $\langle i, j \rangle$ for Y_1 and $\langle j, k \rangle$ for Y_2, which makes three independent indices for two adjacent variables. Combining uv variables into v arguments, yields $uv + v = v(u + 1)$ independent indices.

Since the parsing complexity depends on the fan-out as well as on the rank of the grammar, the grammar is usually binarized before parsing such that the resulting grammar G' has rank 2. Generally, binarization can increase the fan-out of the grammar. In order to minimize parsing complexity, a binarization strategy which also minimizes the fan-out of the binarized grammar can be chosen (Gómez-Rodríguez et al., 2009; Kallmeyer and Maier, 2013).

One disadvantage of the CYK algorithm is that all m antecedents have to be found in order to create a new item with the COMPLETE rule. This involves a lot of index checking at the same time. Directional and Earley parsing strategies have been proposed to alleviate this shortcoming (Ljunglöf, 2004; Burden and Ljunglöf, 2005; Kallmeyer and Maier, 2009).

To provide efficient parsers despite the high parsing complexity of $\mathcal{O}(n^{3v})$ for LCFRS of fan-out v and rank 2 and the large search space (algorithm 1, line 10), several strategies have been followed. In Kallmeyer and Maier (2013), outside estimates which estimate the cost for completing each item are used to speed up parsing. Maier et al. (2012) restrict the fan-out to 2 and present a specialized parser and treebank transformation. In van Cranenburgh (2012), a coarse-to-fine strategy is introduced, which first parses with a probabilistic CFG in order to prune the search space for the actual LCFRS parsing.

Lately, extremely competitive results for parsing discontinuous constituents in terms of parsing accuracy and especially speed have been produced which do not rely on a probabilistic grammar and a chart parser. They either resort to non-projective feature-rich dependency parsing (Hall and Nivre, 2008; Fernández-González and Martins, 2015) or employ a classifier-based transition parser (Versley, 2014; Maier, 2015; Maier and Lichte, 2016; Coavoux and Crabbé, 2017).

2.2 STATISTICAL MACHINE TRANSLATION

This section provides a high-level overview over *statistical machine translation (SMT)* and the corresponding models that are referenced in this thesis. As the focus is on tree-based approaches to SMT, emphasis will be put on models based on Synchronous Context-Free Grammar (SCFG) in section 2.2.5.

A comprehensive and in-depth review of approaches to SMT and methods used therein is beyond the scope of this work. For more complete reports, the reader is referred to Lopez (2008) and Koehn (2010), and to Williams et al. (2016) for tree-based SMT.

2.2.1 *General Framework*

Translation in the context of SMT is formulated as an optimization problem. Given a foreign source sentence f, the task is to find the target sentence e with the highest conditional probability:

$$\hat{e} = \arg\max_{e} P(e|f) \tag{2.1}$$

This is called *decoding*. $f = f_1^J = f_1 \dots f_J$ is a sequence of J words from the source language vocabulary V_F, and $e = e_1^I = e_1 \dots e_I$ is a sequence of I target words from the target language vocabulary V_E.

Traditionally, in the early word-based models (Brown et al., 1993), the objective has been framed as an instance of the noisy-channel approach (Shannon, 1948), where the conditional probability is rewritten according to Bayes' rule:

$$\begin{aligned} \hat{e} = \arg\max_{e} P(e|f) &= \arg\max_{e} \frac{P(e)P(f|e)}{P(f)} \\ &= \arg\max_{e} P(e)P(f|e) \end{aligned} \tag{2.2}$$

The prior $P(f)$ can be dropped because it is constant for a given sentence f. The model is thus factorized into two parts. $P(e)$, the *language model (LM)*, henceforth called $P_{LM}(e)$, is modeling the fluency of the candidate translation e. The language model probability is mostly approximated by an n-gram language model which takes into account

the previous $n-1$ words when determining the probability of a word in a sequence:

$$\begin{aligned} P_{LM}(e) &= \prod_{i=1}^{I} P_{LM}(e_i \mid e_1 \dots e_{i-1}) \\ &\approx \prod_{i=1}^{I} P_{LM}(e_i \mid e_{i-n+1} \dots e_{i-1}) \end{aligned} \tag{2.3}$$

$P(f|e)$ is called the *translation model*. It models the translational correspondence between f and e. The following sections will provide an overview of the various types of translation models. Each type of translation model defines its own notion of *derivation*, explaining how a translation e is generated from f. We define $\mathfrak{f}(d)$ and $\mathfrak{e}(d)$ to denote the source and target side yields of a derivation d respectively.

One specific translation e can usually be derived in various ways under one given model. Accordingly, the calculation of $P(e|f)$ involves a summation over derivations:

$$P(e|f) = \sum_{d} P(e, d|f) \tag{2.4}$$

where $d \in \{d \mid \mathfrak{e}(d) = e, \mathfrak{f}(d) = f\}$ and again

$$P(e, d|f) = \frac{P(e)P(f, d|e)}{P(f)} \,. \tag{2.5}$$

For many translation models, this sum over derivations is computationally intractable. The objective function is therefore approximated using a single derivation:

$$\hat{e} \approx \mathfrak{e}\left(\underset{d \text{ s.t. } \mathfrak{f}(d)=f}{\arg\max}\ P(e, d|f)\right) \tag{2.6}$$

Instead of the direct noisy-channel approach, see equation (2.2) and (2.5), nowadays, a log-linear model of the form

$$\begin{aligned} P(e, d|f) &\propto \prod_{i} \phi_i(d)^{\lambda_i} \\ &\propto \sum_{i} \lambda_i \log \phi_i(d) \end{aligned} \tag{2.7}$$

is generally applied (Och and Ney, 2002), where $\phi_i(d)$ are feature functions defined on derivations d, e and f, and λ_i are feature weights. Features of course include the language model and the translation model, but also many more that have been proven to be useful for translation modeling, such as an inverse translation model feature and a target word count feature.

The weights λ_i are trained discriminatively on held-out data to maximize translation quality as measured by automatic metrics. This step is called *parameter tuning*. The objective of minimizing the error rate can be solved with, e. g., line search (Och, 2003) or downhill simplex (Nelder and Mead, 1965).

The search space for finding the best translation according to equation (2.1) or (2.6) is dependent on the type of translation model used and the respective notion of derivation. Depending on the generative process of a specific type of translation model and the types of features which are used, efficient dynamic programming algorithms have been designed to search for the most likely translation. They will be discussed later in this chapter. In most models, certain independence assumptions are made, which lead to features that are local to each translation rule used in the derivation, and therefore allow to compute solutions to sub-problems. An exception to the local features is the n-gram language model, and a distortion model which is used for phrase-based translation models.

The *evaluation* of machine translation systems is an active research topic on its own. Since human evaluation is costly in terms of time and money, automatic metrics have been developed which compare the machine translation output of test sentences against available human *reference translations*. Their declared goal is to correlate well with human judgment. They are low-cost, consistent and fast to compute, and therefore they are used for iterative system development, as the objective for parameter tuning and for system comparison and benchmarking, despite their flaws.

The most widely used metric still is BLEU (Papineni et al., 2002), which is based on the overlap of n-grams in the translation and the references. More precisely, BLEU is a weighted n-gram precision, typically for $1 \leq n \leq 4$, including a penalty for translations which are shorter than the reference. METEOR (Denkowski and Lavie, 2011) represents an extension of BLEU which puts more emphasis on recall. Furthermore, METEOR also incorporates stemming and synonyms to

allow to also reward words which are not exact matches with the reference. Lately, character-based metrics, e. g. F-measure computed on character level (Popović, 2016), have gained in popularity.

2.2.2 *Word-based Models*

Chronologically, the *word-based models* developed by *IBM* (Brown et al., 1990, 1993) were the first translation models in SMT. While nowadays they are not used as translation models anymore, they introduce many important concepts that are still in use in SMT. Furthermore, word-based models are still applied for certain subtasks of SMT, namely for the task of automatic word alignment (see section 2.2.3) and for some feature functions in translation modeling.

The idea of the word-based models is to view the translation of a sentence as a combination of *lexical translations*. They make the simplifying assumption that each source word f_j corresponds to exactly one target word e_i (or a null token e_0) and that each of those J alignments is independent. This relation is captured by a latent alignment variable a, which is a function $a : \{1, \ldots, J\} \to \{0, 1, \ldots, I\}$, the derivation of the word-based models. For illustration, figure 22 depicts two possible alignments a for the example sentence pair (8).

(8) Ich möchte die Präsidentschaft für ihre Arbeit loben .
I would like the Presidency for its work to praise .
I wish to praise the Presidency for its work.

The translation probability of the sentence pair is then expressed as the sum of the translation probabilities under different alignment functions:

$$P(f|e) = \sum_a P(f, a|e) \tag{2.8}$$

The simplest of the word-based models, IBM Model 1, additionally makes the assumption that all alignments a for a given sentence pair are equally likely. $P(f, a|e)$ then decomposes as follows:

$$P(f, a|e) = P(a|e)P(f|a, e) = \frac{\epsilon}{(I+1)^J} \prod_{j=1}^{J} P(f_j|e_{a(j)}) \tag{2.9}$$

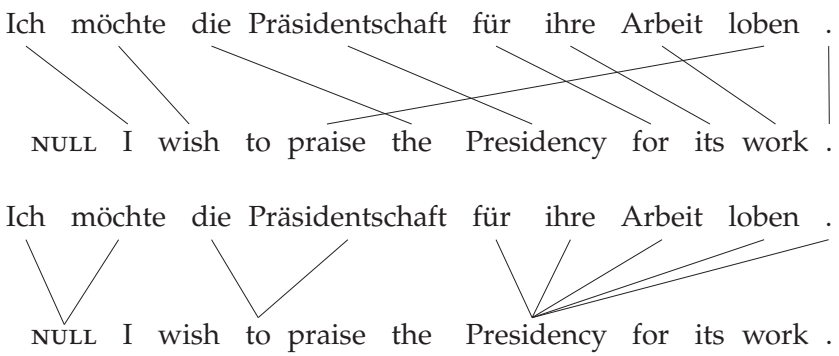


Figure 22: Two possible alignments under the word-based models. The model will (hopefully) learn that the lexical translations in the upper graphic are more likely than those in the lower graphic.

The computation of the translation probability thus boils down to a product of the individual lexical translation probabilities $P(f_j|e_i)$. ϵ is a uniform length probability $P(J|I)$.

IBM Model 2 extends this model by an absolute alignment model which replaces the uniform alignment model $P(a|e)$. The probability of a target word at position i being aligned to a source word at position j then depends on j itself, J and I .

Due to the simple form of these models, $P(f|e)$ can be computed efficiently, as shown in equation (2.10) for IBM Model 1. This means that the exponential number of possible alignments can be compactly represented, making expectation-maximization to train the parameters tractable.

$$P(f|e) = \frac{\epsilon}{(I+1)^J} \prod_{j=1}^{J} \sum_{a(j)=0}^{I} P(f_j|e_{a(j)}) \tag{2.10}$$

The subsequent IBM Models 3–5 increasingly add improvements to Model 2, however at the price of additional complexity for training.

2.2.3 *Word Alignment*

A central concept of SMT in general, that relates to the word-based models, is the *word alignment* $\mathcal{A}$ of a sentence pair $\langle f_1^J, e_1^I \rangle$, capturing the word-to-word correspondences of the two sentences. $\mathcal{A}$ is a set of alignment links $\mathcal{A} \subseteq [1, J] \times [1, I]$. A target word e_i and a source word

f_j are aligned if $(j, i) \in \mathcal{A}$. Words can be unaligned, and words can be aligned to multiple words. Note that this notion of word alignment is symmetrical while the alignment variable a in word-based translation models is asymmetric (cf. section 2.2.2).

Graphically, a word alignment can be viewed as a *bipartite graph*.[15]

DEFINITION 2.57 (Word alignment graph). Let $\mathcal{A}$ be a word alignment of a sentence pair $\langle f_1^J, e_1^I \rangle$. $\mathcal{G} = \langle V, E \rangle$ is a word alignment graph of $\mathcal{A}$ if it holds that $V = \{f_1, \ldots, f_J\} \cup \{e_1, \ldots, e_I\}$ and $E = \{(f_j, e_i) \mid (j, i) \in \mathcal{A}\} \cup \{f_j \prec f_k \mid j < k\} \cup \{e_i \prec e_l \mid i < l\}$. □

We will often refer to a word alignment $\mathcal{A}$ as its alignment graph $\mathcal{G}$ or just call it an *alignment structure*.

A *translation unit* is a maximal connected subgraph of $\mathcal{A}$. We use the notation $\langle D_f, D_e \rangle$ to denote a specific translation unit t where $D_f = \{j \mid f_j \text{ is part of } t\}$ and $D_e = \{i \mid e_i \text{ is part of } t\}$. Where using words instead of indices is unambiguous, we may also use $\langle \boldsymbol{\alpha}_f; \boldsymbol{\alpha}_e \rangle$ to refer to t where $\boldsymbol{\alpha}_f$ and $\boldsymbol{\alpha}_e$ are vectors of words from f and e respectively. They contain the words which constitute t, obeying the precedence relation of the alignment graph.

By this definition, the translation units of a given alignment graph do not overlap. A translation unit is viewed as representing minimal translational equivalence. Note that we do not further constrain translation units; for example, they are not required to consist of contiguous sequences of source and target words. An *alignment configuration* consists of several translation units that are part of one alignment graph.

A word alignment graph is called *complete* if the following condition holds: If f_k is aligned to e_m and e_n, then any f_l that is aligned to e_m must also be aligned to e_n. Goutte et al. (2004) call this property closed under transitivity. If an alignment graph is complete, then its translation units are also complete. In Goutte et al. (2004) they are then called *cepts*. We generally assume complete alignment graphs. In the case of a provided alignment structure not being complete, we just stipulate the missing alignment links.

15 Our definitions of alignment graph and its properties and translation units loosely follow those given by Søgaard (2009) and Søgaard and Kuhn (2009).

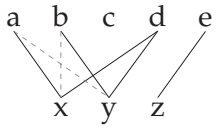


Figure 23: Word alignment example

EXAMPLE 2.58 (Word alignment and translation units). As an example, consider the following word alignment for the sentence pair $\langle$a b c d e , x y z$\rangle$:

$$\mathcal{A} = \{(1,1), (2,2), (4,1), (4,2), (5,3)\}$$

The corresponding alignment graph is given in figure 23. Note that word c is unaligned. This word alignment consists of 3 translation units, which are $\langle\{1,2,4\},\{1,2\}\rangle$, $\langle\{3\},\{\}\rangle$ and $\langle\{5\},\{3\}\rangle$ or, using the other notation, $\langle$a b , d ; x y$\rangle$, $\langle$c ; $\varepsilon\rangle$ and $\langle$e ; z$\rangle$. To make this alignment graph complete, the links $(1,2)$ and $(2,1)$ need to be stipulated. They are shown as dashed lines in figure 23. □

The goal of the *word alignment task* is to identify the word alignment $\mathcal{A}$ of a given sentence pair $\langle f_1^J, e_1^I\rangle$. Large-scale word alignments are created automatically, usually in an unsupervised fashion. Many models and methods have been proposed for this task.

Using the word-based translation models is one option. According to IBM Model 1, the best alignment is computed as:

$$\hat{a} = \arg\max_a \prod_{j=1}^{J} P(f_j|e_{a(j)}) \tag{2.11}$$

In order to obtain a symmetric word alignment from the asymmetric alignments provided by the IBM models, the parallel data is aligned in both directions. The alignments are then combined to a symmetric word alignment by taking the intersection or the union of the alignment links or something in between (Koehn et al., 2003). The IBM models for word alignment and various extensions are implemented as open-source software.[16]

16 *GIZA++* (Och and Ney, 2003) is available at `https://github.com/moses-smt/giza-pp`, accessed on May 19, 2017.

2.2.4 *Phrase-based Models*

Phrase-based translation models are widely used in research and industry today. They are introduced here because they present an intermediate step in the development of hierarchical translation models (section 2.2.5).

In *phrase-based SMT* (Och et al., 1999; Koehn et al., 2003), the translation of a sentence is formulated as the translation of phrases: the source sentence is segmented into phrases, each phrase is translated independently and the target phrases are reordered. Figure 24 illustrates this. A *phrase* in this context is a contiguous sequence of words and does not carry any linguistic meaning. Mathematically:

$$\begin{aligned} P(f, d|e) &= P(d|e)P(f|d, e) \\ &= \prod_{l=1}^{L} r(l)P(\bar{f}_l|\bar{e}_l) \end{aligned} \tag{2.12}$$

A derivation d for phrase-based models captures the segmentation into phrases and the reordering of the phrases. This corresponds to a one-to-one alignment of phrases, and is sometimes called *phrasal alignment*. For a derivation d segmenting the source sentence into L phrases, we notate the target sentence as a sequence of phrases $\bar{e}_1, \ldots, \bar{e}_L$. $\bar{f}_l$ is the source phrase that corresponds to the lth target phrase $\bar{e}_l$, i. e. the source sentence is a permutation of $\bar{f}_1, \ldots, \bar{f}_L$. $P(d|e)$ is the *derivation probability*. The segmentation itself is not explicitly modeled in this formulation of the phrase-based model, i. e. all segmentations are equally likely. $r(l)$ models the *reordering*. In the original formulation, it is a distance-based measure of the amount of source phrase reordering compared to the aligned target phrases, penalizing (long-distance) reorderings. More refined reordering models have been proposed that also take into consideration the words that make up the phrases (Tillmann, 2004; Koehn et al., 2005; Galley and Manning, 2008).

$P(\bar{f}_l|\bar{e}_l)$ is the *phrase translation probability* of translating phrase $\bar{e}_l$ into phrase $\bar{f}_l$. Those probabilities are learned from a parallel training corpus. Usually the corpus is word-aligned first (see section 2.2.3), then all phrase pairs $\langle \bar{f}_l, \bar{e}_l \rangle$ that are consistent with the word alignment are extracted. They serve as the hypothesized distribution from which maximum likelihood estimates are computed. For reasons of practicability, a set of heuristics is applied, e. g. phrase pairs are only extracted up to a certain length (e. g. five words).

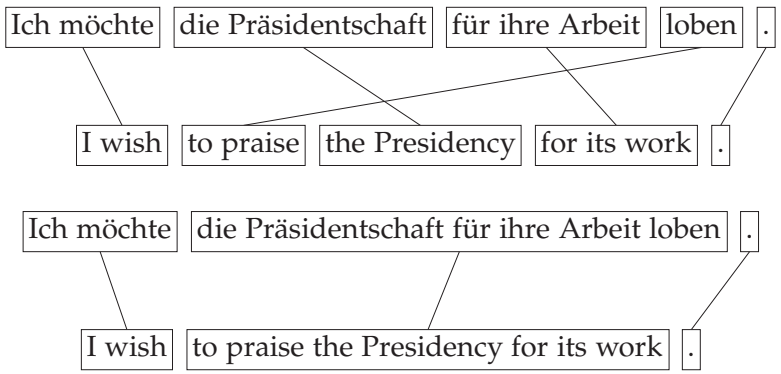


Figure 24: Two possible phrase-based derivations for (8). Which one is chosen depends on the phrase pair inventory of the translation model and the feature values.

As mentioned before, modern SMT systems follow a log-linear approach. The core features which have been established besides the language model feature $P_{LM}(e)$, the phrase translation probabilities $\prod_{l=1}^{L} P(\bar{f}_l|\bar{e}_l)$ and the distortion feature $\prod_{l=1}^{L} r(l)$ are the inverse phrase translation probabilities $\prod_{l=1}^{L} P(\bar{e}_l|\bar{f}_l)$, lexical weighting scores in both directions which are based on lexical translation probabilities (see section 2.2.2), a target sentence length penalty and a phrase count penalty.

During decoding, translation hypotheses are built from left to right on the target side (Koehn, 2004). The search proceeds through a directed acyclic graph of nodes which represent (partial) translation hypotheses. Hypotheses are extended by translating one additional phrase. Hypotheses which are equal in terms of the subsequent translation process are combined. A beam search algorithm using stacks is applied to enumerate hypotheses efficiently. For that, hypotheses are organized in priority queues (called stacks in the SMT literature) of comparable hypotheses, e. g. same number of source words already translated.

Phrase-based translation, including training and decoding, has been implemented in various open-source toolkits, the one with the largest community probably being *Moses*[17] (Koehn et al., 2007).

Since the decoding search problem of phrase-based models is exponential in the source sentence length (Knight, 1999), the amount of

17 `http://www.statmt.org/moses/`, accessed on May 23, 2017.

allowed reordering is usually restricted to a rather small value in practice. This has the shortcoming that many reorderings which would be necessary to correctly translate between languages with different word orders are a priori excluded from the search space. Furthermore, using phrase pairs as translation rules does not allow to learn generalizations which could potentially be useful for translating unseen sentences. Consider the phrase pair *die Präsidentschaft für ihre Arbeit loben / to praise the Presidency for its work* in figure 24. Knowing this phrase pair does not help to translate, e. g., (9). This shortcoming is addressed by the models presented in the next section.

(9) . . . den Jungen für seine Aufmerksamkeit loben .
. . . the boy for his mindfulness to praise .
. . . to praise the boy for his mindfulness.

2.2.5 *SCFG-based Models*

In contrast to the word-based and phrase-based models, *tree-based* (also called *syntax-based*) *models* allow for a hierarchical modeling of translational equivalence. The notion of a phrase is extended to *hierarchical phrases* which can be recursively nested. See figure 25 for an example. The concept of hierarchical phrases is usually formalized by some form of a Synchronous Context-Free Grammar (SCFG), a generalization of Context-Free Grammar (CFG) (def. 2.17, p. 22) that generates pairs of related strings instead of just strings. Translation between string pairs is thus performed via a hierarchical structure, i. e. a parse tree of the source string synchronized with a parse tree of the target string.

Training and decoding for models based on SCFG and related formalisms have been implemented in a wide range of toolkits. They differ in the types of supported models, programming language used, scope and goals, and the size of the community. Three well-known toolkits are *Joshua*[18], *cdec*[19] and *Moses*[20].

18 `http://joshua.incubator.apache.org/`, accessed on May 23, 2017.
19 `http://www.cdec-decoder.org/`, accessed on May 23, 2017.
20 `http://www.statmt.org/moses/` and `http://www.statmt.org/moses/?n=Moses.SyntaxTutorial`, accessed on May 23, 2017.

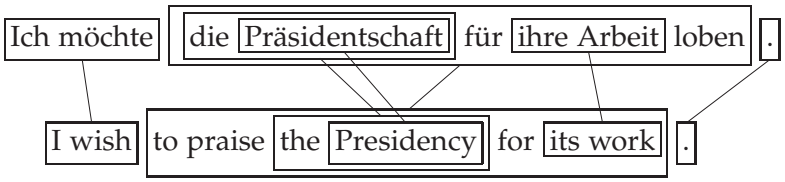


Figure 25: A hierarchical phrase alignment

Formalism

Several definitions for SCFG have been proposed in the literature, e. g. in Chiang (2007) and in Satta and Peserico (2005), in Lewis and Stearns (1968) and Aho and Ullman (1972) as *Syntax-Directed Transduction Grammar*. We will first define an *index annotation* of grammar symbols and then move on to SCFG.

Definition 2.59 (Index annotation (SCFG)). Let N and T be sets of non-terminal and terminal symbols respectively. We define $\mathcal{I}(N) = \{A_{\boxed{k}} \mid A \in N, k \in \mathbb{N}\}$ and $I = \mathcal{I}(N) \cup T$. The set of all indices (integers k) that occur in symbols in $\gamma \in I^*$, is denoted by $ind(\gamma)$. Two strings $\gamma_1, \gamma_2 \in I^*$ are *synchronous* if $ind(\gamma_1) = ind(\gamma_2)$, each index in $ind(\gamma_1)$ occurs exactly once in γ_1 and each index in $ind(\gamma_2)$ occurs exactly once in γ_2. We call γ_1 and γ_2 *independent* iff $ind(\gamma_1) \cap ind(\gamma_2) = \emptyset$. □

Definition 2.60 (Synchronous Context-Free Grammar). A *Synchronous Context-Free Grammar (SCFG)* is a tuple $G = (N_s, N_t, T_s, T_t, P, S_s, S_t)$ where

1. N_s and N_t are finite sets of source and target non-terminal symbols,
2. T_s and T_t are finite sets of source and target terminal symbols,
3. N_s and T_s, respectively N_t and T_t are disjoint,
4. $S_s \in N_s$ and $S_t \in N_t$ are distinguished start symbols,
5. P is a finite set of *synchronous (rewriting) rules* of the form

   $$\langle A \to \alpha_s, B \to \alpha_t \rangle$$

   with $A \in N_s$, $B \in N_t$, $\alpha_s \in I_s^*$, $\alpha_t \in I_t^*$, where $I_s = \mathcal{I}(N_s) \cup T_s$ and $I_t = \mathcal{I}(N_t) \cup T_t$, and α_s and α_t being synchronous strings. □

In certain contexts, it is necessary to refer to the individual components of a synchronous grammar separately.

DEFINITION 2.61 (Source and target projections (SCFG)). Let $G = (N_s, N_t, T_s, T_t, P, S_s, S_t)$ be an SCFG and $r \in P$ with $r = \langle r_s, r_t \rangle$ a synchronous rule. The *source* and *target projection* of r are $proj_s(r) = r_s$ and $proj_t(r) = r_t$ respectively. The source and target projections of G, $proj_s(G)$ and $proj_t(G)$, are the CFGs $G_s = (N_s, T_s, P_s, S_s)$ and $G_t = (N_t, T_t, P_t, S_t)$ with the rule sets $P_s = \{proj_s(r) \mid r \in P\}$ and $P_t = \{proj_t(r) \mid r \in P\}$. □

If the source projection of a rule $\langle A \to \alpha_s, B \to \alpha_t \rangle \in P$ has the same left-hand side (LHS) as its target projection, i. e. $A = B$, then we can also use the following notation: $A \to \langle \alpha_s, \alpha_t \rangle$.

DEFINITION 2.62 (Rank (SCFG)). Let $G = (N_s, N_t, T_s, T_t, P, S_s, S_t)$ be an SCFG. The rank of a rule $\langle A \to \alpha_s, B \to \alpha_t \rangle \in P$ is the number of non-terminals occurring in α_s or α_t. The rank of G is the maximal rank of any of its rules $r \in P$. G is called a u-SCFG if it has rank u. □

The *derives* relation for SCFG simultaneously rewrites two non-terminals with the same index, starting with a pair of co-indexed start symbols. During a derivation step, pairs of fresh indices are introduced. We therefore first establish the notion of a *reindexing*.

DEFINITION 2.63 (Reindexing). A *reindexing* is an injective function $f : \mathbb{N} \to \mathbb{N}$. f is extended to I in the following way: $f(A_{\boxed{k}}) = A_{\boxed{f(k)}}$ for $A_{\boxed{k}} \in \mathcal{I}(N)$ and $f(a) = a$ for $a \in T$. f is furthermore extended to strings in I^* such that $f(\varepsilon) = \varepsilon$ and $f(X\gamma) = f(X)f(\gamma)$, for $X \in I$ and $\gamma \in I^*$. □

DEFINITION 2.64 (Derivation (SCFG)). Let $G = (N_s, N_t, T_s, T_t, P, S_s, S_t)$ be an SCFG. Let $\gamma_s \in I_s^*$, $\gamma_t \in I_t^*$ be synchronous strings.

1. $\Rightarrow_G$ is called the *derives* relation. It is defined as follows:

   $$\langle \gamma_s, \gamma_t \rangle \Rightarrow_G \langle \delta_s, \delta_t \rangle$$

   iff there exists an index k in $ind(\gamma_s)$, a synchronous rule $\langle A_s \to \alpha_s, A_t \to \alpha_t \rangle \in P$ and some reindexing f such that

   a) $f(\alpha_s \alpha_t)$ and $\gamma_s \gamma_t$ are independent, and

   b) $\gamma_i = \gamma_i' A_{i\boxed{k}} \gamma_i''$, $\delta_i = \gamma_i' f(\alpha_i) \gamma_i''$ for $i \in \{s, t\}$.

   If G is clear in the context, we can use $\Rightarrow$ instead of $\Rightarrow_G$. To make the applied rule $r = \langle A_s \to \alpha_s, A_t \to \alpha_t \rangle$ explicit, we can use $\Rightarrow_G^r$.

2. $\overset{*}{\Rightarrow}_G$ is the reflexive transitive closure of $\Rightarrow_G$.

3. Let $m \in \mathbb{N}$ and $\gamma_s^{(i)} \in I_s^*, \gamma_t^{(i)} \in I_t^*$ be synchronous strings, for $1 \leq i \leq m$. $\langle \gamma_s^{(1)}, \gamma_t^{(1)} \rangle \Rightarrow_G \ldots \Rightarrow_G \langle \gamma_s^{(m)}, \gamma_t^{(m)} \rangle$ is a *derivation* d of length m. □

Like derivations of a CFG can be represented as trees, an SCFG derivation can be viewed as a *pair of derivation trees* (one source tree and one target tree). In such a representation, synchronously derived non-terminals are usually linked with lines or they are annotated with the same index k.

In correspondence to the language of a CFG, we define the *translation* of an SCFG.

DEFINITION 2.65 (Translation (SCFG)). Let $G = (N_s, N_t, T_s, T_t, P, S_s, S_t)$ be an SCFG.

1. The *translation* generated by G is a binary relation over $T_s^* \times T_t^*$: $\mathcal{T}(G) = \{\langle f, e \rangle \mid \langle S_s\boxed{1}, S_t\boxed{1} \rangle \overset{*}{\Rightarrow}_G \langle f, e \rangle, f \in T_s^*, e \in T_t^*\}$. We also call $\mathcal{T}(G)$ the *language* of G.

2. Let $f = f_1 \ldots f_n$ be a string. The set of *translations* of f generated by G is the following: $\mathcal{T}(G, f) = \{e \mid \langle f, e \rangle \in \mathcal{T}(G)\}$. □

EXAMPLE 2.66 (Synchronous Context-Free Grammar). Let $G = (\{\mathrm{S}, \mathrm{A}\}, \{\mathrm{S}, \mathrm{B}\}, \{a, b\}, \{a, b, c\}, P, \mathrm{S}, \mathrm{S})$ be an SCFG with P as follows:

$$\begin{aligned}
&\langle \mathrm{S} \to \mathrm{A}\boxed{1} \quad , \mathrm{S} \to \mathrm{B}\boxed{1} \rangle \\
&\langle \mathrm{A} \to a\mathrm{A}\boxed{1}b \, , \mathrm{B} \to abc\mathrm{B}\boxed{1} \rangle \\
&\langle \mathrm{A} \to ab \quad , \mathrm{B} \to abc \rangle
\end{aligned}$$

The translation generated by G is $\mathcal{T}(G) = \{\langle a^n b^n, (abc)^n \rangle \mid n \in \mathbb{N}\}$. Figure 26 shows the synchronous derivation tree for the following derivation of $\langle aabb, abcabc \rangle$:

$$\begin{aligned}
\langle \mathrm{S}\boxed{1}, \mathrm{S}\boxed{1} \rangle &\Rightarrow \langle \mathrm{A}\boxed{2}, \mathrm{B}\boxed{2} \rangle \\
&\Rightarrow \langle a\mathrm{A}\boxed{3}b, abc\mathrm{B}\boxed{3} \rangle \\
&\Rightarrow \langle aabb, abcabc \rangle
\end{aligned}$$

□

An *Inversion Transduction Grammar (ITG)* is a restricted type of SCFG in which the indexed right-hand side (RHS) non-terminals in the

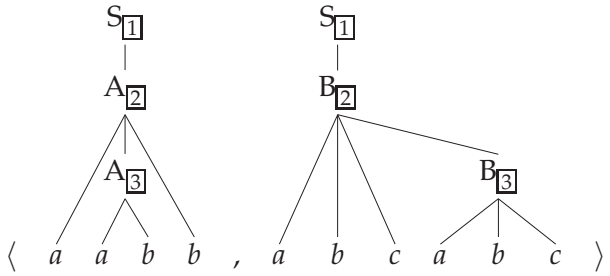


Figure 26: SCFG derivation tree

source projection occur in the same order or in exactly the inverse order as their counterparts in the target projection. ITG was the first formalism used for tree-based SMT (Wu, 1997). Often, a specific notional variant is used for ITG, but we keep the introduced SCFG notation.

Definition 2.67 (Inversion Transduction Grammar). Let $G = (N_s, N_t, T_s, T_t, P, S_s, S_t)$ be an SCFG. G is an ITG iff for any rewriting rule

$$\langle A \to \alpha_s, B \to \alpha_t \rangle \in P$$

with

$$\alpha_s = \alpha_s^0 A^1_{\boxed{l_1}} \alpha_s^1 A^2_{\boxed{l_2}} \dots A^u_{\boxed{l_u}} \alpha_s^u \ ,$$

α_t is exactly such that

$$\alpha_t = \alpha_t^0 B^1_{\boxed{l_1}} \alpha_t^1 B^2_{\boxed{l_2}} \dots B^u_{\boxed{l_u}} \alpha_t^u \quad \text{or} \quad \alpha_t = \alpha_t^0 B^1_{\boxed{l_u}} \alpha_t^1 B^2_{\boxed{l_{u-1}}} \dots B^u_{\boxed{1}} \alpha_t^u$$

with $A^1 \dots A^u \in N_s$ and $B^1 \dots B^u \in N_t$, $\alpha_s^0 \dots \alpha_s^u \in T_s^*$ and $\alpha_t^0 \dots \alpha_t^u \in T_t^*$, $l_1 \dots l_u \in \mathbb{N}$, and u being the rank of the rule. □

Under this defintion of ITG, SCFG and ITG are equivalent for grammars of maximally rank 2.

Most often, ITG makes an appearance in its normal form in the literature, where the rank of the grammar is at most 2 and the RHSs of the rules contain either terminals or non-terminals, but not both.

Definition 2.68 (Normal Form (ITG/SCFG)). Let $G = (N_s, N_t, T_s, T_t, P, S_s, S_t)$ be an SCFG. G is in *normal form* iff all rules $r \in P$ have one of the following forms:

1. $\langle A^0 \to A^1_{\boxed{l_1}} A^1_{\boxed{l_2}}, B^0 \to B^1_{\boxed{l_1}} B^2_{\boxed{l_2}} \rangle$

2. $\langle A^0 \to A^1_{\boxed{l_1}} A^1_{\boxed{l_2}}, B^0 \to B^1_{\boxed{l_2}} B^2_{\boxed{l_1}} \rangle$
3. $\langle A^0 \to t_s, B^0 \to t_t \rangle$

with $A^0, A^1, A^2 \in N_s$, $B^0, B^1, B^2 \in N_t$, $t_s \in T_s^*$, $t_t \in T_t^*$ and $l_1, l_2 \in \mathbb{N}$. □

Other definitions of ITG are more restrictive in that the RHSs of the terminal rewriting rule 3 in def. 2.68 is often limited to at most one source-side and one target-side symbol. The given definition loosely follows Søgaard and Wu (2009) who argue that, even in the original paper, the idea of lexical rules with more than one lexical unit is put forward (Wu, 1997, section 6).

For any ITG G, there exists an equivalent *Normal-form Inversion Transduction Grammar (NF-ITG)* G', i. e. $\mathcal{T}(G) = \mathcal{T}(G')$ (Wu, 1997). As will be explained in chapter 3, G and G' however differ in their alignment capacity.

In contrast, it is not generally the case that any SCFG G can be converted into an equivalent *Normal-form Synchronous Context-Free Grammar (NF-SCFG)* G' because not every SCFG is binarizable (Aho and Ullman, 1972; Zhang et al., 2006). See ex. 2.69. However, when creating grammars, may it be manually or automatically, one can of course stipulate that grammars must be in normal form or that rules must have at most rank 2 from the very beginning.

Example 2.69 (Binarization of ITG/SCFG). Let us consider the following ITG rule:

$$A \to \langle B_{\boxed{1}} C_{\boxed{2}} D_{\boxed{3}} E_{\boxed{4}}, E_{\boxed{4}} D_{\boxed{3}} C_{\boxed{2}} B_{\boxed{1}} \rangle$$

It can be binarized by iteratively grouping two synchronous and adjacent non-terminals under a new non-terminal symbol, e. g.,

$$\begin{aligned} A &\to \langle B_{\boxed{1}} A'_{\boxed{2}}, A'_{\boxed{2}} B_{\boxed{1}} \rangle \\ A' &\to \langle C_{\boxed{1}} A''_{\boxed{2}}, A''_{\boxed{2}} C_{\boxed{1}} \rangle \\ A'' &\to \langle D_{\boxed{1}} E_{\boxed{2}}, E_{\boxed{2}} D_{\boxed{1}} \rangle \end{aligned}$$

However, the following SCFG rule cannot be binarized since no two non-terminals are synchronous and adjacent on both sides:

$$A \to \langle B_{\boxed{1}} C_{\boxed{2}} D_{\boxed{3}} E_{\boxed{4}}, D_{\boxed{3}} B_{\boxed{1}} E_{\boxed{4}} C_{\boxed{2}} \rangle$$ □

The translations which are produced by SCFGs fall into the same class as translations by some other formalisms. First, *Linear Context-Free Rewriting Systems (LCFRSs)* (def. 2.29, p. 26) where each non-terminal has exactly fan-out 2 are equivalent to SCFGs. Furthermore, *Synchronous Tree-Substitution Grammar (STSG)* generates the same class of translations as SCFG. *Tree-Substitution Grammars (TSGs)* are Tree-Adjoining Grammars (TAGs) (see p. 55) without adjunction operation and auxiliary trees. STSG is the generalization of TSG to pairs of trees and the generation of pairs of strings (Eisner, 2003).

DEFINITION 2.70 (Synchronous Tree-Substitution Grammar). A *Synchronous Tree-Substitution Grammar (STSG)* is a tuple $G = (N_s, N_t, T_s, T_t, P, S_s, S_t)$ where N_s, N_t, T_s, T_t, S_s and S_t are defined as for SCFG (def. 2.60, p. 76) and P is a finite set of synchronous (rewriting) rules. Each rule $r \in P$ is a pair of trees $\langle \pi_s, \pi_t \rangle$ where the labels of the nodes in π_x are drawn from N_x except the leaf nodes whose labels are drawn from $N_x \cup T_x$, for $x \in \{s, t\}$. Each non-terminal leaf (a *substitution node*) in π_s is paired exactly with one non-terminal leaf in π_t and vice versa (denoted by boxed indices). □

An STSG derivation starts with a tree pair $\langle \pi_s, \pi_t \rangle$. While in an SCFG derivation, synchronous non-terminals are rewritten with sequences of symbols, in an STSG derivation, paired substitution nodes in $\langle \pi_s, \pi_t \rangle$ are rewritten with a pair of trees $\langle \pi'_s, \pi'_t \rangle \in P$. The label pair of the root nodes of $\langle \pi'_s, \pi'_t \rangle$ has to match the labels of the substitution nodes.

We do not formalize STSGs here any further since conceptually they are included in the context-free grammar formalisms for translation modeling. Even though the trees as basic units for rewriting offer an extended domain of locality compared to SCFG and are therefore interesting for linguistic description of syntax, STSG is weakly equivalent to SCFG. Ignoring the internal structure of the trees $\langle \pi_s, \pi_t \rangle \in P$, each tree pair can be converted to an SCFG rule in which the LHSs are the root labels and the RHSs are the *flattened* trees, i. e. they consist of the frontier nodes.

Translation modeling using SCFG or STSG involves a mapping between two trees. This relationship can also be formalized as a *tree transducer.*[21] Accordingly, some approaches to SMT have been formulated

21 See Knight and Graehl (2005) for tree transducers and their application in natural language processing.

as tree transductions (e. g. Galley et al., 2006; Graehl et al., 2008). While tree transducers in their general form are more powerful than SCFG and STSG, for SMT usually a restricted form is employed (linear and non-deleting) which is equivalent to STSG (Shieber, 2004). Throughout this work, the grammar view will be adopted, but the tree transducer literature will still be referenced where appropriate.

Statistical Model Definition

To weight the translation options in $\mathcal{T}(G, f)$, the features of the log-linear model (cf. equation (2.7)) are defined in the following manner (e. g. Chiang, 2007):

$$\begin{aligned} P(e, d|f) &\propto \prod_i \phi_i(d)^{\lambda_i} \\ &\propto P_{LM}(e)^{\lambda_{LM}} \omega(d) \end{aligned} \tag{2.13}$$

P_{LM} provides the language model feature score. The other features ($i \neq LM$) are defined on the synchronous rules of a *weighted SCFG* which are applied during the derivation d. The weight of a derivation d is the product of the weights of the rules that constitute d.

Definition 2.71 (Weighted Synchronous Context-Free Grammar). A *weighted Synchronous Context-Free Grammar* is a tuple

$$G = (N_s, N_t, T_s, T_t, P, S_s, S_t, \omega)$$

where $(N_s, N_t, T_s, T_t, P, S_s, S_t)$ is an SCFG and $\omega : P \to \mathbb{R}_{\geq 0}$ is a weight function which maps from rules to real numbers. □

Definition 2.72 (Weight of a derivation (Weighted SCFG)). Let $G = (N_s, N_t, T_s, T_t, P, S_s, S_t, \omega)$ be a weighted SCFG. Let $\gamma_s \in I_s^*$ and $\gamma_t \in I_t^*$, and $\delta_s \in I_s^*$ and $\delta_t \in I_t^*$ be synchronous strings.

1. Let $r \in P$. The weight of one derivation step $\langle \gamma_s, \gamma_t \rangle \Rightarrow_G^r \langle \delta_s, \delta_t \rangle$ is defined as

$$\omega(\langle \gamma_s, \gamma_t \rangle \Rightarrow_G^r \langle \delta_s, \delta_t \rangle) = \omega(r)$$

2. Let $r_1, \ldots, r_m \in P$, $m \in \mathbb{N}$. Let d be a derivation $\langle \gamma_s, \gamma_t \rangle \Rightarrow_G^{r_1} \ldots \Rightarrow_G^{r_m} \langle \delta_s, \delta_t \rangle$. The weight of d is defined as

$$\omega(\langle \gamma_s, \gamma_t \rangle \Rightarrow_G^{r_1} \ldots \Rightarrow_G^{r_m} \langle \delta_s, \delta_t \rangle) = \prod_{j=1}^{m} \omega(r_j)$$

□

To fit the log-linear model, ω is usually computed as follows:

$$\omega(r) = \prod_{i \neq LM} \phi_i(r)^{\lambda_i} \tag{2.14}$$

where $r \in P$ is a synchronous rule of G, ϕ_i are feature functions and λ_i are feature weights.

Putting everything together, the search task for translation given a source sentence f is the following:

$$\hat{e} \approx \mathfrak{e}\left(\underset{d \text{ s.t. } \mathfrak{f}(d)=f}{\arg\max}\ P_{LM}(\mathfrak{e}(d))^{\lambda_{LM}} \prod_{j=1}^{m} \prod_{i \neq LM} \phi_i(r_j)^{\lambda_i}\right) \tag{2.15}$$

where d consists of m rule applications.

The base features that have been proven to be useful are the same as for phrase-based translation: the language model probability for the target sentence $P_{LM}(e)$, the direct and the inverse conditional translation probabilities on a rule basis $P(r_s|r_t)$ and $P(r_t|r_s)$, lexical weighting scores in both directions $lex(r_s|r_t)$ and $lex(r_t|r_s)$ which are based on lexical translation probabilities, a rule count penalty and a target sentence length penalty. Note that the language model probability P_{LM} cannot be computed on a rule basis and is therefore not part of the main product in equation (2.15). We will return to this issue in the section about decoding.

Model Classification and Grammar Learning

Many different SCFG- and STSG-based translation models have been proposed. Following the classification in Williams et al. (2016), they can be grouped along the dimension of whether the derived trees are supposed to resemble syntactic constituency trees which allow for a linguistic interpretation or not (see def. 2.5, p. 18 and section 2.1.3 in general). We will first focus on string-to-string models which do not build linguistic syntax trees, and then shortly describe the other classes of tree-based translation models.

Hierarchical phrase-based models (Hiero) (Chiang, 2005, 2007) are the most prominent representative of the class of *string-to-string models*. They do not allude to syntactic parse trees neither during training nor during decoding, and the translation grammar G is only syntactic in a formal sense. The motivation is to extend phrase-based models

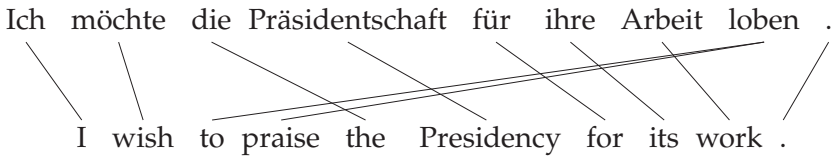


Figure 27: Word alignment for ex. 2.73

(section 2.2.4) by allowing for *holes* (also called *gaps*) in the phrases to make them more general. The holes are represented by non-terminals labeled X.

Hiero rules are extracted from the parallel word-aligned training corpus in a two-step procedure. Firstly, for a word-aligned sentence pair, all conventional phrase pairs (called *initial phrase pairs*) are extracted (see section 2.2.4). From the set of initial phrase pairs, hierarchical phrase pairs are then created by substituting smaller initial phrase pairs which are embedded in larger phrase pairs by a synchronous non-terminal. See ex. 2.73 for illustration. This rule extraction is usually constrained by a list of heuristics in order to obtain grammars of manageable size and to make parsing more efficient (Chiang, 2007):

1. The maximum length of initial phrase pairs is limited, e. g., to 10 words on either side.
2. Unaligned words are not allowed on the boundaries of initial phrase pairs.
3. The length of the RHS of the source projection of SCFG rules is limited, e. g., to five symbols (terminals or non-terminals).
4. The maximum rank of each SCFG rule is 2.
5. Non-terminals in the source projection of SCFG rules are not allowed to be adjacent.

Example 2.73 (Hierarchical phrase-based rule extraction). We provide a non-exhaustive list of the rules which are created for the sentence pair (8) and its the word alignment $\mathcal{A}$ in figure 27. The rules on the top are lexical rules from initial phrase pairs, while the rules on the bottom are rules create from hierarchical phrase pairs. The rules marked with a diamond correspond to the hierarchical phrase pairs shown in figure 25.

$X \to \langle \text{Ich} , \text{I} \rangle$
$X \to \langle \text{Ich möchte} , \text{I wish} \rangle^{\Diamond}$
$X \to \langle \text{loben} , \text{to praise} \rangle$
$X \to \langle \text{die Präs.} , \text{the Pres.} \rangle$
$X \to \langle \text{Präs.} , \text{Pres.} \rangle^{\Diamond}$
$X \to \langle \text{Präs. für} , \text{Pres. for} \rangle$
$X \to \langle \text{die Präs. für} , \text{the Pres. for} \rangle$
$X \to \langle \text{ihre Arbeit} , \text{its work} \rangle^{\Diamond}$
$X \to \langle \text{die Präs. für ihre Arbeit} , \text{the Pres. for its work} \rangle$
$X \to \langle \text{die Präs. für ihre Arbeit loben} , \text{to praise the Pres. for its work} \rangle$
$X \to \langle . , . \rangle^{\Diamond}$
...
$X \to \langle \text{Ich } X_{\boxed{1}} , \text{I } X_{\boxed{1}} \rangle$
$X \to \langle X_{\boxed{1}} \text{ möchte} , X_{\boxed{1}} \text{ wish} \rangle$
$X \to \langle \text{Ich möchte } X_{\boxed{1}} . , \text{I wish } X_{\boxed{1}} . \rangle$
$X \to \langle \text{die } X_{\boxed{1}} , \text{the } X_{\boxed{1}} \rangle^{\Diamond}$
$X \to \langle X_{\boxed{1}} \text{ für } X_{\boxed{2}} , X_{\boxed{1}} \text{ for } X_{\boxed{2}} \rangle$
$X \to \langle X_{\boxed{1}} \text{ für } X_{\boxed{2}} \text{ loben} , \text{to praise } X_{\boxed{1}} \text{ for } X_{\boxed{2}} \rangle^{\Diamond}$
$X \to \langle X_{\boxed{1}} \text{ für } X_{\boxed{2}} \text{ loben} . , \text{to praise } X_{\boxed{1}} \text{ for } X_{\boxed{2}} . \rangle$
$X \to \langle \text{die Präs. } X_{\boxed{1}} \text{ ihre } X_{\boxed{2}} , \text{the Pres. } X_{\boxed{1}} \text{ for } X_{\boxed{2}} \rangle$
... □

In addition to the automatically extracted rules, a few further rules, the so-called *glue rules*, are added to the grammar. They allow to monotonically combine X trees to produce S trees, where S is the start symbol of the grammar, and model the beginning and end of sentences.

$S \to \langle S_{\boxed{1}} X_{\boxed{2}} , S_{\boxed{1}} X_{\boxed{2}} \rangle$
$S \to \langle \langle s \rangle , \langle s \rangle \rangle$
$S \to \langle S_{\boxed{1}} \langle / s \rangle , S_{\boxed{1}} \langle / s \rangle \rangle$

As an example, consider figure 25. Glue rules would be used to combine the synchronous tree for *Ich möchte/I wish* with the tree covering the middle part of the sentence, and then to combine it with the sentence-final period.

Now SCFG rules have been extracted, but the actual derivations d of the sentence pairs in the training corpus are latent. In order to nevertheless estimate values for the conditional translation probabilities $P(r_s|r_t)$ and $P(r_t|r_s)$, the SCFG rules which have been extracted from the training corpus are taken as the distribution from which to obtain maximum likelihood estimates.

We will now point out the other classes of tree-based translation models. In contrast to the string-to-string models, they produce derivations which resemble linguistic parse trees either on the source side, on the target side, or both. This in turn means that the respective side of the training corpus needs to be parsed using a monolingual syntactic parser (see section 2.1).

String-to-tree models aim at producing a linguistic parse tree for the translation. The intuition is that this kind of approach should lead to more syntactically well-formed, fluent translations. Two well-known lines of work can be classified as string-to-tree: *Syntax-Augmented Machine Translation* (Zollmann and Venugopal, 2006) is a variant of Hiero which uses syntactic categories from target side parse trees to learn rules with more refined target non-terminal labels. The *GHKM* approach (Galley et al., 2004, 2006), named after its authors, uses an STSG with target side tree fragments which are extracted from parse trees. Decoding is performed in the same way as for string-to-string models: the source string is parsed using the grammar while building a target side derivation according to the grammar rules. More details will be presented in the next section.

In *tree-to-string* models, a syntactic parse tree is derived on the source side. Instead of performing syntactic parsing and decoding in one step, these models provide the possibility to firstly use a monolingual syntactic parser to parse the source sentence (the one which has been used to provide syntactic analyses of the source sentences in the training data) and to secondly match the synchronous translation rules against this parse tree (*tree decoding*). This reduces the translation search space and makes tree-to-string decoding fast in practice. The first systems in this category have been presented by Huang et al. (2006) and Liu et al. (2006) and are known under the name *Syntax-Directed Translation*. The drawback of potential mistakes in the one parse tree which is generated for the input sentence is addressed in *forest-to-string* models.

Tree-to-tree models can also make use of the efficient tree decoding process. However, they suffer from the fact that source and target linguistic parse trees are usually non-isomorphic, leading to large, non-modular STSG tree fragments and data sparsity, and in turn to a decreased translation quality compared to models which use less syntactic constraints (Chiang, 2010). Instead, syntax is often used as a *soft* constraint (e. g. Zhang et al., 2011); however, strictly speaking, those models do not fall into the tree-to-tree category.

Decoding as Parsing

Decoding with an SCFG-based translation model boils down to CFG parsing (Melamed, 2004; Yamada and Knight, 2002) with beam search to integrate the language model component into decoding (Chiang, 2007). We describe the most common approach within this section. Recall equation (2.15). The goal of tree-based decoding is to find the highest-scoring derivation d, according to a product of weighted feature scores, which yields the input string f on the source side. The target yield of d provides the translation of f.

As it turns out, the main challenge in this search is the feature contributed by the n-gram language model $\phi_{LM}(d) = P_{LM}(e)$ in equation (2.13) and (2.15). Without this term, the task could be solved exactly using dynamic programming (e. g. by using the weighted deductive system in ex. 2.45, p. 38). However, the language model score cannot be decomposed in the same way as the other feature scores since it is not local to the rules of the grammar. It cannot be (exactly) calculated on (S)CFG sub-derivations: when determining the weight of an item ranging from j_1 to j_2, language model scoring needs to know about the translations of words outside of this item in order to calculate the correct score. This means that the optimality of an item cannot be guaranteed without knowing the context in which this item will be used.

For a moment, we will ignore the term contributed by the language model. The decoding task can then be solved by parsing f using the source projection of the synchronous grammar (def. 2.61, p. 77), thereby making sure that valid target side derivations are created, and reading off e from the corresponding target side tree. The machine translation community here draws from efficient dynamic programming algorithms, such as CYK parsing. Note that translation gram-

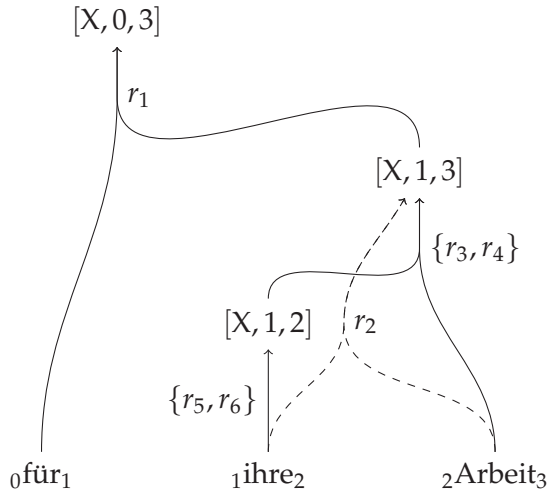


Figure 28: SCFG parse hypergraph for translation

mars are highly lexicalized, in contrast to grammars used for treebank parsing for example, which means that algorithms which require the grammar to be in Chomsky Normal Form (CNF) are probably not a smart choice. One option is the *modified CYK algorithm* presented as a deduction system in ex. 2.45, p. 38. Another popular algorithm in tree-based machine translation systems is *CKY+* (Chappelier and Rajman, 1998). It does not require the grammar to be of rank 2, but implicitly binarizes the grammar on the fly using Earley-style dotted items. Other chart parsing algorithms have been developed for translation grammars specifically (e. g. DeNero et al., 2009; Hopkins and Langmead, 2010). Which algorithm to choose depends on characteristics of the grammar, e. g. its rank and the number of non-terminal symbols.

The result of parsing f with the monolingual source projection of the grammar is a *parse hypergraph* (see section 2.1.2) which represents the full space of derivations of f. Since each source-projected rule of the grammar maps to one or more target projections (sometimes called a *rule bundle*), each hyperedge in the parse hypergraph represents one or more SCFG rule applications.

EXAMPLE 2.74 (SCFG parse hypergraph for translation). Consider the following SCFG translation grammar fragment G:

$$r_1: \quad \mathrm{X} \to \langle \text{für } \mathrm{X}_{\boxed{1}}, \text{for } \mathrm{X}_{\boxed{1}} \rangle$$
$$r_2: \quad \mathrm{X} \to \langle \text{ihre Arbeit}, \text{its work} \rangle$$
$$r_3: \quad \mathrm{X} \to \langle \mathrm{X}_{\boxed{1}} \text{ Arbeit}, \mathrm{X}_{\boxed{1}} \text{ work} \rangle$$

r_4 : $\mathrm{X} \rightarrow \langle \mathrm{X}_{\boxed{1}}$ Arbeit , $\mathrm{X}_{\boxed{1}}$ activities$\rangle$

r_5 : $\mathrm{X} \rightarrow \langle$ihre , its$\rangle$

r_6 : $\mathrm{X} \rightarrow \langle$ihre , her$\rangle$

It contains two rule bundles: $\{r_3, r_4\}$ and $\{r_5, r_6\}$. Figure 28 shows the parse hypergraph representing the parse forest which is generated by parsing *für ihre Arbeit* with G. There are two derivations leading to $[\mathrm{X}, 0, 3]$. One is depicted with solid arrows. The other one differs in the derivation of $[\mathrm{X}, 1, 3]$ which is shown with the dashed hyperedge. □

We now augment the parse hypergraph to a more fine-grained *search hypergraph* from which we can determine the highest scoring translation according to equation (2.15). For this, we take advantage of the independence assumptions of language modeling. See equation (2.3). While a parse item (i. e. a node in the parse hypergraph) represents all subderivations which share the same source span $f_{j_1} \dots f_{j_2}$ and the same non-terminal which covers this span (see CYK items in ex. 2.45, p. 38, and the nodes in figure 28), a *search item* (i. e. a node in the search hypergraph) also contains the information which is relevant for n-gram language model scoring when combining items to superderivations. This information is sometimes called the *language model state*. Put simply, those are the $n-1$ first and last target words of the partial translation generated by the search item. In the search hypergraph, each hyperedge represents the application of one SCFG rule.

EXAMPLE 2.75 (SCFG search hypergraph for translation). We continue with ex. 2.74, p. 88. Figure 29 shows the corresponding search hypergraph which expands the parse hypergraph. It assumes the use of a bigram language model ($n = 2$). The search items therefore record the one rightmost and leftmost word of each translation hypothesis. □

Unfortunately, exploring the complete search hypergraph in order to find the best translation (e. g. using algorithm 3) is usually unfeasible. It amounts to a time complexity of $\mathcal{O}(|P|J^{u+1}|T|^{u \cdot 2(n-1)})$, with $|P|$ the number of rules in the grammar G, J the length of the input sequence f, $|T|$ the number of possible target side words, u the rank of G, typically $u = 2$, and n the order of the language model. This is clearly an explosion of the search space in comparison to standard monolingual syntactic parsing with $\mathcal{O}(|P|J^{u+1})$.

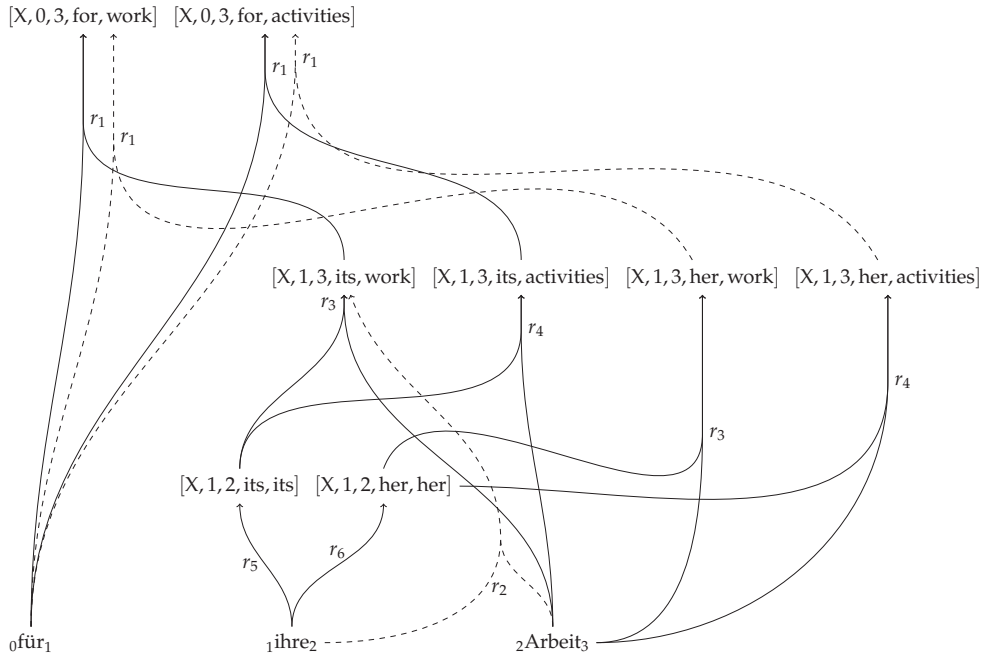


Figure 29: SCFG search hypergraph for translation. Some hyperedges are dashed for better readability.

Several approximate algorithms making this decoding task tractable have appeared in the literature. They use beam search methods to significantly reduce the search space and only compute a limited amount of search items for each parse item. We will present *cube pruning* (Chiang, 2007; Huang and Chiang, 2007) here, which is probably the most popular one. The pseudocode is provided in algorithm 6.

Cube pruning visits the nodes of the parse hypergraph in a bottom-up manner, thereby filling a beam of maximally size k with derivations of search items (lines 3–5). Instead of computing and scoring all search items at one parse node and then selecting the k best ones, cube pruning only considers the most promising items using the eager k-best parsing algorithm of Huang and Chiang (2005). See algorithm 4 on p. 46ff. However, contrary to standard parsing tasks, the weight function is only approximately monotonic due to the language model score that is incurred for the boundary words when combining search items. This means that the derivations which are popped from the candidate queue *cand* (see line 16) are not in the true best-first order. To account for that, they are first stored in an auxiliary data structure (*beam*) be-

```
 1: ⟨V, E⟩: parse hypergraph with a GOAL node
 2: function CUBEPRUNING(⟨V, E⟩)
 3:     for v ∈ V in bottom-up order do
 4:         CUBEPRUNE(v)
 5:     end for
 6:     return D_1(GOAL)
 7: end function
 8:
 9: procedure CUBEPRUNE(v)
10:     cand = ∅                                          ▷ Priority queue
11:     beam = ∅                                          ▷ List
12:     for all e where e is an incoming hyperedge of v do
13:         PUSH(cand,⟨e, 1, 1⟩)
14:     end for
15:     while |beam| < k and |cand| > 0 do
16:         d = POPMAX(cand)
17:         APPEND(beam, d)
18:         PUSHNEIGHBORS(d, cand)
19:     end while
20:     sort beam to D(v)
21: end procedure
22:
23: procedure PUSHNEIGHBORS(⟨e, x, y⟩, cand)
24:     for all i s.t. 1 ≤ i ≤ |T(e)| do
25:         if x_i < |D(T_i(e))| then
26:             x' = x + b^i
27:             PUSH(cand, ⟨e, x', y⟩)
28:         end if
29:     end for
30:     if y < |rule bundle of e| then
31:         PUSH(cand, ⟨e, x, y + 1⟩)
32:     end if
33: end procedure
```

Algorithm 6: Cube pruning (Huang and Chiang, 2007)

fore they are sorted to the final list of derivations of search items of v (line 20), and also the beam size k is usually increased. Search errors can still happen, but are accepted given the obtained speed-up.

Recall the definition of derivations with backpointers in a hypergraph (def. 2.51, p. 44) since a similar concept is used here. To identify a derivation of a search item (which expands a parse item v), we use the notation $\langle e, \boldsymbol{x}, y \rangle$ to denote a derivation along the hyperedge e, with the fine-grained search item antecedents being the x_ith derivation $D_{x_i}(v')$ of the tail node $v' = T_i(e)$ for $1 \leq i \leq |\boldsymbol{x}|$. y is an index into the list of target projections of the rule bundle that is associated with the hyperedge e, sorted by weight. $\boldsymbol{D}(v)$ is used to denote the k highest weighted derivations of search items which correspond to the parse item v, sorted according to their weight (see also def. 2.50, p. 44).

The *cube* in the name of the algorithm refers to the search grid which is represented along one hyperedge e: one dimension for the rule, i. e. the different target sides of the rule bundle, and the other two dimensions for the typically two antecendents due to the application of a rule of rank 2. The core of the algorithm starts by constructing the supposedly best derivation (the one in the corner of the cube), using the rule with the highest weight and the best search item derivation of each of the tail nodes, scoring it and putting it on the priority queue *cand* (line 13). For each derivation that is popped from *cand*, its *neighboring derivations*, using the next best derivations of the antecedents (lines 24–29) and the next best rule (lines 30–32), are processed in the same way (lines 15–19).

Finally, the best search item derivation of the GOAL parse item $D_1(\text{GOAL})$ provides the translation $\hat{e}$ we are looking for. For obtaining the *k-best translations*[22], one of the k-best parsing algorithms of Huang and Chiang (2005) (see algorithms 4 and 5) can be applied to the pruned search hypergraph.

2.3 CONCLUSION AND OUTLOOK

The reader should now be equipped with the required background knowledge to follow along the work at hand. We have introduced the concepts with respect to grammar formalisms and parsing on the

22 This k is different from and usually a lot smaller than the parameter k used within the cube pruning algorithm.

one hand and machine translation on the other hand which build the groundwork of this thesis. We have learned that CFG lacks the capability to model non-local dependencies as they occur in natural language, and that LCFRS allows for data-driven constituency parsing while modeling discontinuous constituents in a direct manner. We have reviewed how statistical machine translation evolved from simple word-based translation models over phrase-based to hierarchical models. The latter provide a link to grammar formalisms and parsing as they employ CFGs in a synchronized manner and rely on parsing for decoding.

The following chapter will show that SCFG is not powerful enough to model all alignment configurations which occur in alignments of parallel natural language texts. We therefore propose to use a grammar formalism beyond CFG for translation modeling, and define, implement and evaluate a hierarchical phrase-based translation system which allows for discontinuous phrases.

3

MOTIVATION

This chapter will provide motivation for going beyond Context-Free Grammar (CFG) for translation modeling. Translational correspondences in the form of manual word alignments are investigated as to how far they can be covered with current popular translation models. Furthermore, a qualitative analysis of the alignment configurations that are beyond the alignment capacity of those models is provided.

The main findings of section 3.2 have already been published in Kaeshammer (2013), and the material in section 3.3 has been published in Kaeshammer and Westburg (2014):

> Kaeshammer, M. (2013). Synchronous linear context-free rewriting systems for machine translation. In *Proceedings of the Seventh Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 68–77, Atlanta, Georgia. Association for Computational Linguistics.
>
> Kaeshammer, M. and Westburg, A. (2014). On complex word alignment configurations. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 1773–1780, Reykjavik, Iceland. European Language Resources Association (ELRA).

3.1 COMPLEX ALIGNMENT CONFIGURATIONS

The space of alignment configurations that can be generated with a certain class of translation models is limited. This is due to constraints inherent in the translation model itself and potentially imposed by the decoding strategy, and, in the case of tree-based statistical machine translation (SMT), due to structural constraints imposed by the particular synchronous grammar formalism. It is therefore also meaningful to talk about the *alignment capacity* of a synchronous grammar formalism (Søgaard and Wu, 2009; Saers et al., 2011) besides its weak and strong generative capacity (see p. 49). This term refers to the ability of

the (synchronous) grammar formalism to generate a set of alignments or translation units.

We call alignment configurations that are beyond the alignment capacity of current translation models, i. e. phrase-based (section 2.2.4) and 2-SCFG-based models (section 2.2.5), *complex alignment configurations*. They have been the matter of some debate in the machine translation community, as they call for more powerful translation models.

3.1.1 *Preliminaries*

When referring to the alignment capacity of a formalism, the following assumption is made in this and related work: words that are recognized or generated simultaneously, by the application of a synchronous rule, are aligned (Wu, 1997; Wellington et al., 2006; Søgaard and Kuhn, 2009; Søgaard, 2010). Accordingly, we call a synchronous derivation tree a *hierarchical alignment*. Each synchronous constituent can be interpreted as directly aligning its immediate lexical children as well as transitively providing alignment information for the other terminals in its yield.

In the interpretation of multi-word translation units, we follow the methodology of interpreting the many-to-many alignments *conjunctively*, as advocated by Søgaard and Kuhn (2009). This means that all alignment links in a translation unit have to be induced in order to say that the respective translation unit or alignment configuration is induced. This differs from the methodotology of Wellington et al. (2006) where multi-word translation units are interpreted in a *disjunctive* manner, i. e. it is enough to generate one link of a translation unit in order to say that the translation unit is induced.

Based on these assumptions and the definition of translation unit in section 2.2.2, the terminals that are recognized or generated by a synchronous rule correspond to one or more translation units. Alignment structures induced by synchronous grammars are *complete* (Søgaard and Kuhn, 2009).

It is assumed that translation units represent minimal translational equivalence and that they are crucial for translation. An adequate formalism for the modeling of translation should thus be able to induce each translation unit separately.

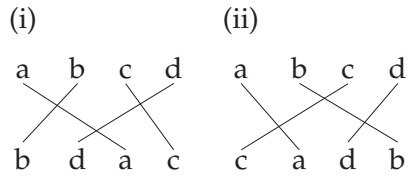


Figure 30: Schematic IO alignments

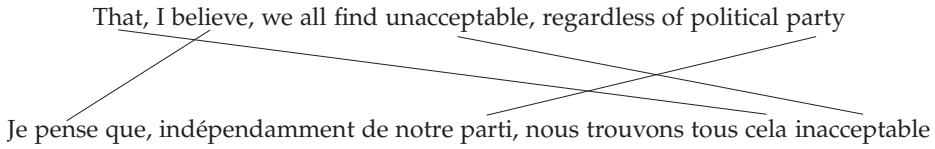


Figure 31: English-French example of an IO alignment from Wellington et al. (2006). Only the links which constitute the complex alignment configuration are depicted.

3.1.2 *Inside-out Alignments*

Wu (1997) identifies two configurations, the so-called *inside-out (IO) alignments*, that the full class of Inversion Transduction Grammars (ITGs) (def. 2.67, p. 79) cannot induce. Figure 30 schematically depicts those alignment configurations. The reason is that there is no way to put four constituents together with purely straight or inverted operations such that the shown alignments are generated. See figure 31 for a natural language example.

Synchronous Context-Free Grammars (SCFGs) (def. 2.60, p. 76) of rank 2 neither induce IO alignments. Every context-free derivation of rank 2 on one side, either source or target side, leads to at least one discontinuous constituent on the respective other side and is therefore beyond context-free expressivity. The full class of SCFGs in contrast is able to generate IO alignments, due to higher ranks. However, for reasons of parsing complexity, in practice usually grammars with a maximal rank of 2 are used for translation.

EXAMPLE 3.1 (SCFG rules for IO alignment).

$$\langle X \to \mathrm{a} \quad , X \to \mathrm{a} \rangle$$
$$\langle X \to \mathrm{b} \quad , X \to \mathrm{b} \rangle$$
$$\langle X \to \mathrm{c} \quad , X \to \mathrm{c} \rangle$$
$$\langle X \to X_{\boxed{1}} X_{\boxed{2}} X_{\boxed{3}}\ \mathrm{d}\ , X \to X_{\boxed{2}}\ \mathrm{d}\ X_{\boxed{1}} X_{\boxed{3}} \rangle$$

The example shows sample SCFG rules of rank 3 that generate the IO alignment configuration in figure 30(i). Note that this grammar is beyond the expressivity of ITG since, in the last rule, the right-hand side (RHS) non-terminals on the target side do neither occur in the same order as on the source side nor in exactly the inverse order. □

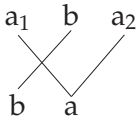
Phrase-based translation systems are generally able to generate IO alignments due to the reordering component that is usually employed in corresponding decoders. In practice, a distortion limit might however limit the generation of IO alignments that involve large phrases.

3.1.3 *Discontinuous Translation Units*

Discontinuous translation units (DTUs) are translation units which have at least one gap, i. e. the sequence of words that belong to the translation unit is interrupted by at least one word which is not part of this translation unit. The alignment in ex. 3.2 features one DTU.

Unrestricted ITG and SCFG can induce some DTUs, while the corresponding normal forms cannot (Søgaard and Wu, 2009). For a Normal-form Synchronous Context-Free Grammar (NF-SCFG), a terminal rule which generates the DTU would be required. Since by definition the DTU is not continuous on the source or the target side (or both), such a rule is not possible. With unrestricted SCFG, in contrast, a mixed rule can be used which combines the DTU with an already established constituent filling the gap.

EXAMPLE 3.2 (SCFG rules for DTU).



$$\langle X \to \mathrm{b} \quad , X \to \mathrm{b} \rangle$$
$$\langle X \to \mathrm{a}_1 X_{\boxed{1}}\ \mathrm{a}_2\ , X \to X_{\boxed{1}}\ \mathrm{a} \rangle$$

The example shows a schematically depicted DTU and corresponding SCFG rules that derive it. □

Two configurations of DTUs cannot be induced by the full class of ITG nor SCFG. Those are *cross-serial discontinuous translation units (CDTUs)* (Søgaard and Kuhn, 2009) and *bonbon alignments* (Simard et al., 2005). A CDTU consists of two DTUs which are discontinuous on the same side and grouped in such a way that material from each DTU is in the gap of the other DTU. A bonbon alignment is similar, but the DTUs are discontinuous on different sides. See figure 32 for schematic representations of the configurations, and figure 33 and 34 for natural language examples.

Other interesting DTUs are *multigap DTUs*, i. e. DTUs with arbitrarily many gaps. As already pointed out before, Normal-form Inversion Transduction Grammar (NF-ITG) and NF-SCFG do not induce them. Unrestricted ITGs induce a particular subclass of multigap DTUs, namely those where the aligned material in the gaps occurs on the target side in the same order or in the inverse order compared to the source side (Søgaard and Wu, 2009). Unrestricted SCFG in contrast also induces multigap DTUs where the material in the gaps does not adhere to this constraint.

A 2-SCFG, or equivalently a 2-ITG, also induces only a particular subclass of multigap DTUs, namely those where the words in the gaps form at most two continuous sequences of aligned source and target words. This means that, e. g., DTUs with three or more gaps on one side cannot be generated.

During our investigation (see section 3.2), we found that there is another class of configurations that is beyond the alignment capacity of 2-SCFG and phrase-based systems, which has not been reported in the literature before. They consist of a DTU with one gap and three other translation units. The four translation units are configured in a similar way as the IO alignment, i. e. no three of the four translation units form a continuous sequence in the source and target strings. We name the configurations in this class *inside-out discontinuous translation units (IO-DTUs)*.

More specifically, the configurations can be described by the following patterns, where x is the DTU and a, b and c are the other three translation units: (i) one of the set $\{xabc, abxc\}$ on one side, and one of the set $\{bx_1acx_2, x_1cax_2b\}$ on the other side, or (ii) one of the set $\{axbc, abcx\}$ on one side and one of the set $\{bx_1cax_2, x_1acx_2b\}$ on the other side, assuming that same letters are aligned and therefore form

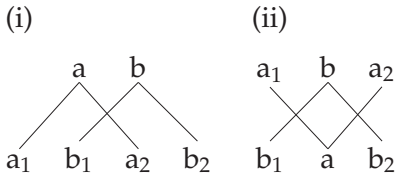


Figure 32: Schematic CDTU (i) and bonbon alignment (ii). They can also occur upside down.

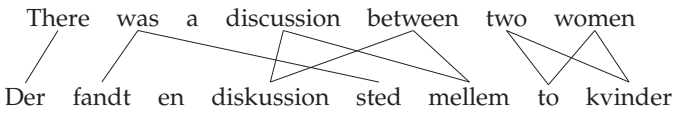


Figure 33: English-Danish example from Søgaard (2008a) which includes a CDTU. Søgaard (2008a) argues that *fandt . . . sted* (lit. *found place*) is fully idiomatic and therefore one translation unit, and furthermore that the noun-preposition pairs are idiosyncratic in that the preposition to chose is stored with the lexical entry of the noun and that therefore they are also best treated as translation units.

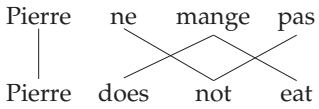


Figure 34: French-English example of a bonbon alignment from Simard et al. (2005).

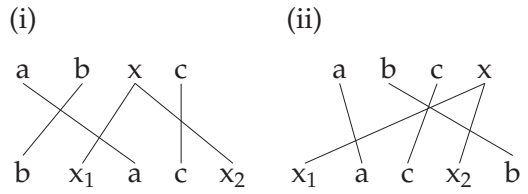


Figure 35: Two schematic IO-DTUs. They can also occur upside down.

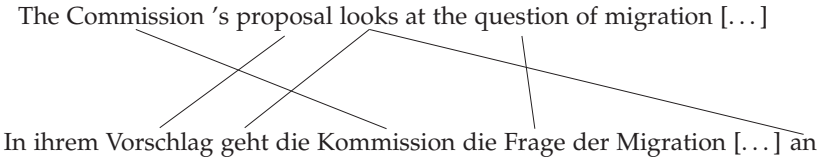


Figure 36: English-German example of a IO-DTU. It corresponds to the schematic configuration (i) in figure 35. The configuration has been found in the manually aligned Europarl data of Padó and Lapata (2006).

a translation unit. Figure 35 shows two instances of IO-DTUs, and figure 36 depicts a natural language example.

Some other combinations of the translation units *a*, *b*, *c* and *x* are also beyond the alignment capacity of 2-SCFG, but they coincide with the already known IO alignments.

Standard phrase-based translation models cannot induce any of the alignment configurations involving a DTU since phrases are by definition continuous.

3.2 EMPIRICAL ALIGNMENT CAPACITY OF SYNCHRONOUS CONTEXT-FREE GRAMMAR

The alignment capacity of a formalism can be studied from a theoretical point of view, e. g. by identifying and characterizing alignment structures that a formalism cannot generate in contrast to the alignment structures that are induced. In addition, the alignment capacity of a formalism can also be studied empirically by evaluating it on word-aligned sentence pairs. Such sentence pairs can be generated automatically in large amounts using machine learning or by human annotators in smaller quantities. The empirical alignment capacity of

a formalism allows to quantify the structures which cannot be generated by the formalism under question.

In this section, we investigate the empirical alignment capacity of different variants of SCFG with respect to manually aligned data. We focus on SCFGs of rank 2 (equivalent to 2-ITG) since this is the prevalent formalism in tree-based machine translation.

3.2.1 *Alignment Validation*

Our study is based on *alignment validation* (Søgaard, 2010). This term refers to checking whether a given alignment structure $\mathcal{A}$ is valid with respect to a formalism, meaning that an all-accepting grammar of the formalism under question can generate the alignment. An *all-accepting grammar* is a grammar which contains all possible rules that can be expressed in the respective formalism.

Example 3.3 (All-accepting grammar).

$$\langle M \to w^+ \quad , M \to w^+ \rangle$$
$$\langle A \to M_{\boxed{1}} \quad , A \to M_{\boxed{1}} \rangle$$
$$\langle A \to A_{\boxed{1}} A_{\boxed{2}} , A \to A_{\boxed{1}} A_{\boxed{2}} \rangle$$
$$\langle A \to A_{\boxed{1}} A_{\boxed{2}} , A \to A_{\boxed{2}} A_{\boxed{1}} \rangle$$

This example shows the rules for an all-accepting SCFG in normal form, which is equivalent to an ITG in normal form. Both M and A are generic non-terminals; M serves as a pre-terminal. w is a word in the source or target vocabulary. □

For alignment validation, we use the idea of a *bottom-up hierarchical aligner* (Wellington et al., 2006). It works very much like a synchronous parser. However, the constraints for inference are not the rewriting rules of a synchronous grammar, but the word alignments and potentially other things, such as the normal form constraints specified in the all-accepting grammar in ex. 3.3. Initial constituents are built from the word alignments, then constituents are combined with each other, obeying the specified constraints. The goal is to find a constituent that completely covers the input pair $\langle s, t \rangle = \langle s_1 \dots s_n, t_1 \dots t_{n'} \rangle$.

We specify the hierarchical aligner in terms of a deduction system (see section 2.1.2). Our items have the form $[X, \boldsymbol{\rho}_s, \boldsymbol{\rho}_t]$ where $X \in \{A, M\}$ is a non-terminal symbol of the simulated grammar. All-

accepting grammars usually have only one non-terminal symbol, but we need a distinction between pre-terminal constituents M and general constituents A for simulating SCFG in normal form as well as the full class. $\boldsymbol{\rho}_s$ and $\boldsymbol{\rho}_t$ are bit vectors that characterize the spans of the synchronous constituent on the source and target side respectively. Their interpretation is that s_i is in the yield of X if $\boldsymbol{\rho}_s(i) = 1$; otherwise s_i is not in the yield of X. Accordingly for the target side, t_i is in the yield of X if $\boldsymbol{\rho}_t(i) = 1$; otherwise t_i is not in the yield of X.

We furthermore specify some useful operations for bit vectors. The $\cup$ operator combines bit vectors of the same length to a new bit vector by an element-wise *or* operation. The intersection $\cap$ of two bit vectors of the same length is the element-wise *and* operation. 0^l is a bit vector $\boldsymbol{\rho}$ of length l such that $\boldsymbol{\rho}(i) = 0$ for all $0 < i \leq l$. The function $b(\boldsymbol{\rho})$ returns the number of blocks of $\boldsymbol{\rho}$, i. e. the number of continuous sequences of 1s in $\boldsymbol{\rho}$.

The input sentence pair $\langle s, t \rangle$ is segmented into m disjoint translation units $\langle D_s^{(j)}, D_t^{(j)} \rangle$ $(1 \leq j \leq m)$ based on the given word alignment $\mathcal{A}$. $D_s^{(j)}$ and $D_t^{(j)}$ are sets of word indices into s and t respectively.

We now specify the deduction rules of the hierarchical aligner which simulate an all-accepting NF-SCFG. Each rewriting rule in ex. 3.3 gives rise to one deduction rule. The SCAN operation builds pre-terminal items from the translation units of the input sentence pair:

$$\frac{}{[M, \boldsymbol{\rho}_s, \boldsymbol{\rho}_t]} \quad \text{a translation unit } \langle D_s^{(j)}, D_t^{(j)} \rangle$$

where $\boldsymbol{\rho}_s(i) = 1$ if $i \in D_s^{(j)}$, otherwise $\boldsymbol{\rho}_s(i) = 0$, and $\boldsymbol{\rho}_t(i) = 1$ if $i \in D_t^{(j)}$, otherwise $\boldsymbol{\rho}_t(i) = 0$.

The UNARY rule creates A items from pre-terminal M items:

$$\frac{[M, \boldsymbol{\rho}_s, \boldsymbol{\rho}_t]}{[A, \boldsymbol{\rho}_s, \boldsymbol{\rho}_t]} \quad b(\boldsymbol{\rho}_s) = 1, b(\boldsymbol{\rho}_t) = 1$$

The side condition specifies that only A items with continuous spans can be built. This is because we are simulating an SCFG.

Two A items are combined to a larger A item with the BINARY operation:

$$\frac{[A, \boldsymbol{\rho}_s^1, \boldsymbol{\rho}_t^1], [A, \boldsymbol{\rho}_s^2, \boldsymbol{\rho}_t^2]}{[A, \boldsymbol{\rho}_s^3, \boldsymbol{\rho}_t^3]} \quad \begin{array}{c} \boldsymbol{\rho}_s^1 \cap \boldsymbol{\rho}_s^2 = 0^n, \boldsymbol{\rho}_t^1 \cap \boldsymbol{\rho}_t^2 = 0^{n'}, \\ \boldsymbol{\rho}_s^3 = \boldsymbol{\rho}_s^1 \cup \boldsymbol{\rho}_s^2, \boldsymbol{\rho}_t^3 = \boldsymbol{\rho}_t^1 \cup \boldsymbol{\rho}_t^2, \\ b(\boldsymbol{\rho}_s^3) = 1, b(\boldsymbol{\rho}_t^3) = 1 \end{array}$$

The side conditions specify that the spans of the two antecedent items, characterized by the bit vectors, may not overlap and, again, that only new items with continuous spans can be built.

The GOAL item is an item of the form $[A, \boldsymbol{\rho}_s, \boldsymbol{\rho}_t]$ with $\boldsymbol{\rho}_s(i) = 1$ for all $0 < i \leq n$ and $\boldsymbol{\rho}_t(i) = 1$ for all $0 < i \leq n'$, i. e. an A item which spans the complete input sentence pair $\langle s, t \rangle$. If the hierarchical aligner finds such a GOAL item, the alignment structure $\mathcal{A}$ of $\langle s, t \rangle$ is valid, i. e. can be induced with an SCFG in normal form.

We are also interested in the empirical alignment capacity of SCFG without normal-form restriction since this is the grammar formalism behind the prevalent hierarchical phrase-based translation model; see section 2.2.5. We therefore present an extended deduction system which features two additional rules. They lead to the simulation of an SCFG of rank 2. Terminals and non-terminals can be combined on the RHS of a rewriting rule.

The UNARYMIXED rule combines an M item with an A item:

$$\frac{[M, \boldsymbol{\rho}_s^M, \boldsymbol{\rho}_t^M], [A, \boldsymbol{\rho}_s^A, \boldsymbol{\rho}_t^A]}{[A, \boldsymbol{\rho}_s, \boldsymbol{\rho}_t]} \quad \begin{array}{c} \boldsymbol{\rho}_s^M \cap \boldsymbol{\rho}_s^A = 0^n, \boldsymbol{\rho}_t^M \cap \boldsymbol{\rho}_t^A = 0^{n'}, \\ \boldsymbol{\rho}_s = \boldsymbol{\rho}_s^M \cup \boldsymbol{\rho}_s^A, \boldsymbol{\rho}_t = \boldsymbol{\rho}_t^M \cup \boldsymbol{\rho}_t^A, \\ b(\boldsymbol{\rho}_s) = 1, b(\boldsymbol{\rho}_t) = 1 \end{array}$$

This simulates an SCFG rule of rank 1 which combines terminals with an already created A constituent.

The BINARYMIXED rule combines an M item with two A items simulating an SCFG rule of rank 2:

$$\frac{[M, \boldsymbol{\rho}_s^M, \boldsymbol{\rho}_t^M], [A, \boldsymbol{\rho}_s^1, \boldsymbol{\rho}_t^1], [A, \boldsymbol{\rho}_s^2, \boldsymbol{\rho}_t^2]\rangle}{[A, \boldsymbol{\rho}_s^3, \boldsymbol{\rho}_t^3]} \quad \begin{array}{c} \boldsymbol{\rho}_s^M \cap \boldsymbol{\rho}_s^1 = 0^n, \boldsymbol{\rho}_s^1 \cap \boldsymbol{\rho}_s^2 = 0^n, \\ \boldsymbol{\rho}_s^2 \cap \boldsymbol{\rho}_s^M = 0^n, \boldsymbol{\rho}_t^M \cap \boldsymbol{\rho}_t^1 = 0^{n'}, \\ \boldsymbol{\rho}_t^1 \cap \boldsymbol{\rho}_t^2 = 0^{n'}, \boldsymbol{\rho}_t^2 \cap \boldsymbol{\rho}_t^M = 0^{n'}, \\ \boldsymbol{\rho}_s^3 = \boldsymbol{\rho}_s^M \cup \boldsymbol{\rho}_s^1 \cup \boldsymbol{\rho}_s^2, \\ \boldsymbol{\rho}_t^3 = \boldsymbol{\rho}_t^M \cup \boldsymbol{\rho}_t^1 \cup \boldsymbol{\rho}_t^2, \\ b(\boldsymbol{\rho}_s^3) = 1, b(\boldsymbol{\rho}_t^3) = 1 \end{array}$$

Note that the creation of M items in the SCAN rule is not constrained by the number of blocks of the bit vectors. According to the

deduction system, the span of M items is only dictated by the translation units which are based on the alignment $\mathcal{A}$. M items can therefore have discontinuous spans, i. e. the covered words are not necessarily adjacent. This is crucial for alignment validation with the full class of SCFGs since they can cover certain discontinuous translation units by means of rules with mixed right-hand sides. The lack of the continuity constraint in the SCAN rule does however not lead to derivations that exceed the power of SCFG. The reason is that M items are not GOAL items and that all other rules obey the continuity constraint.

For the computation of the items, we use standard chart parsing techniques, maintaining a chart and an agenda; see algorithm 1.

3.2.2 *Experiments*

We apply the bottom-up hierarchical aligner to each manually aligned sentence pair in our data sets. If a goal item is found, its alignment structure can be induced with the formalism in question. We measure the number of sentence pairs for which a hierarchical alignment was reached over the total number of sentence pairs. Søgaard (2010) refers to this as *alignment reachability*, which is the inverse of *parse failure rate* (Wellington et al., 2006).

Data

While large-scale word alignments are created automatically, mostly in an unsupervised fashion, e. g. (Och and Ney, 2000), a number of gold alignment data sets for several language pairs exist. They are manually created, high quality reference alignments that have emerged from various projects and shared tasks on word alignment. They vary in size, annotation methodology, alignment guidelines and original purpose.

For our study, we use such manually aligned parallel corpora, assuming that they represent gold translational equivalence.[1] The following data sets have already been used in previous similar experiments, e. g. in Wellington et al. (2006), Søgaard and Wu (2009) and Søgaard (2010): English-Romanian and English-Hindi sentence pairs from Martin et al. (2005), English-French data from Mihalcea and Pedersen (2003), the Europarl data sets described in Graça et al. (2008a)

1 Whenever there are sure (S) and possible (P) alignments annotated, we use both.

for the six combinations of English, French, Portuguese and Spanish, the English-German Europarl data that was created for Padó and Lapata (2006), and the data sets with Danish as the source language that are part of the Parole corpus of the Copenhagen Dependency Treebank (CDT) (Buch-Kromann et al., 2009).

We furthermore perform our study on data sets that, to the best of our knowledge, have not been evaluated in a similar setting before. Those are English-Swedish gold alignments documented in Holmqvist and Ahrenberg (2011), the English-Inuktitut data used in Martin et al. (2005), more English-German Europarl data aligned by T. Schoenemann[2], the English-Spanish data set in Lambert et al. (2005) and English-Dutch alignments that are part of the Dutch parallel Corpus (Macken, 2010). Characteristics about the data sets are presented in table 3.

Results

Table 4 shows the results of our experiments, namely alignment reachability scores for NF-SCFG and 2-SCFG (or equivalently NF-ITG and 2-ITG) on a large variety of data sets. They confirm results of previous similar studies, namely that NF-SCFG is not capable of generating the majority of alignment configurations.

For grammars without normal-form constraint, alignment reachability is generally higher. We tested grammars of rank 2, and we found that for each data set over 90% of the sentence pairs can be induced with a 2-SCFG, i. e. without the necessity of discontinuous constituents. One exception are the alignments in the Schoenemann data set. 2-SCFG is the grammar formalism behind successfully applied translation models, e. g. in *Hiero* (Chiang, 2007).

Nevertheless, our experiments clearly show that the gold alignments contain a proportion of structures that cannot be generated by 2-SCFGs. They range from 1% in the Portuguese-Spanish data to 23.89% in the English-German Schoenemann data. Only the English-Inuktitut alignments can be completely derived with a 2-SCFG.

2 Obtained from `http://user.phil-fak.uni-duesseldorf.de/~tosch/downloads.html`, accessed on February 12, 2013.

		#SPs	min	med	max
Martin et al.	*en-ro*	447	2 | 2	20 | 19	96 | 94
	en-hi	115	1 | 1	10 | 12	45 | 58
	en-iu	100	10 | 3	26 | 10	79 | 26
Padó & Lapata	*en-de*	987	5 | 5	23 | 23	40 | 40
Mihalcea & Pedersen	*en-fr*	447	2 | 2	16 | 17	30 | 30
Graça et al.	*en-fr*	100	4 | 4	11 | 13	14 | 21
	en-pt	100	4 | 3	11 | 12	14 | 21
	en-es	100	4 | 4	11 | 11	14 | 24
	pt-fr	100	3 | 4	12 | 13	21 | 21
	pt-es	100	3 | 4	12 | 11	21 | 24
	es-fr	100	4 | 4	11 | 13	24 | 21
CDT	*da-en*	5464	1 | 1	16 | 17	89 | 98
	da-de	449	1 | 1	17 | 18	75 | 74
	da-es	807	1 | 1	16 | 18	78 | 97
	da-it	1514	1 | 1	16 | 19	78 | 268
Holmqvist & Ahrenb.	*en-sv*	1164	1 | 1	21 | 19	40 | 40
Schoenemann	*en-de*	300	1 | 1	21 | 22	77 | 79
Lambert et al.	*en-es*	500	4 | 4	26 | 27	90 | 99
Macken	*en-nl*	699	1 | 1	20 | 19	107 | 105

Table 3: Characteristics of the manually aligned data sets: number of sentence pairs, minimal, median and maximal sentence length on the source and the target side

		NF-SCFG	**2-SCFG**	Søgaard (2010) NF-ITG	ITG
Martin et al.	*en-ro (30)*	45.07	**95.07**	-	-
	en-hi (40)	82.73	**96.36**	-	-
	en-iu (40)	40.66	**100.00**	-	-
Padó & Lapata	*en-de (15)*	73.74	**94.41**	38.97	45.13
Mihalcea & Pedersen	*en-fr*	67.56	**95.30**	*76.98	*81.75
Graça et al.	*en-fr*	73.00	**95.00**	65.00	68.00
	en-pt	76.00	**98.00**	65.00	67.00
	en-es	82.00	**96.00**	73.00	74.00
	pt-fr	73.00	**92.00**	63.00	63.00
	pt-es	90.00	**99.00**	80.00	81.00
	es-fr	74.00	**91.00**	68.00	68.00
CDT	*da-en (25)*	72.90	**97.80**	-	-
	da-de (25)	64.87	**94.94**	*47.62	*49.35
	da-es (25)	66.61	**97.50**	*30.68	*35.54
	da-it (25)	69.01	**97.95**	*60.00	*60.00
Holmqvist & Ahrenb.	*en-sv (30)*	82.83	**95.60**	-	-
Schoenemann	*en-de (40)*	29.15	**76.11**	-	-
Lambert et al.	*en-es (40)*	47.15	**94.85**	-	-
Macken	*en-nl (30)*	57.14	**94.86**	-	-

Table 4: Alignment reachability scores of our experiments and those of Søgaard (2010) for reference. The numbers in parentheses are the sentence length cut-offs that were used in our experiments. The results marked with * are not directly comparable to ours because different versions of the data sets were used.

Discussion

Our alignment validation study shows that, even though translation models based on 2-SCFG are successfully applied and can currently be considered state-of-the-art, they fail to cover a considerable portion of alignment configurations. This observation is based on a variety of language pairs from a wide range of data sets.

It should be mentioned that it is not clear yet how alignment reachability relates to machine translation quality and evaluation. In the end, experimental evidence will have to show how alignment capacity relates to machine translation quality. The results in Galley and Manning (2010) for Chinese-English translation indicate that more powerful translation models also lead to better translations.

We can nevertheless infer from the presented results that what is considered as translationally equivalent by the various annotators of the data sets and their guidelines is beyond the search space of SCFG. This finding motivates investigations towards more powerful translation models which can induce the complex alignment configurations.

3.2.3 *Related Work*

Our empirical investigation is similar to previous studies concerning the alignment capacity of grammar formalisms, but differs in some crucial points. Wu (1997), Zens and Ney (2003), Wellington et al. (2006) and Søgaard (2010) all use some sort of hierarchical alignment algorithm based on a given word alignment with the goal of investigating the alignment complexity. Most notably, the number of corpora and the variety of language pairs investigated in our study exceeds all previous investigations.

In Zens and Ney (2003), a weaker normal-form for ITG is assumed, and only two automatically aligned data sets are tested.

The methodology of Wellington et al. (2006) differs from ours in the interpretation of alignment links. While we treat them conjunctively, which has been argued for in Søgaard and Kuhn (2009), Wellington et al. (2006) treat them *disjunctively*, which means that in the case of n-to-m alignments with $n, m \geq 1$, it is enough to induce one of the involved alignment links. With this methodology one issue of translational equivalence modeling, namely discontinuous translation units, is ignored. The failure rates they present are therefore much lower

than ours. Wellington et al. (2006) in addition show that allowing discontinuities in translation models becomes even more essential for inducing gold alignments when the synchronous derivations are further constrained by monolingual syntactic parse trees on the source and/or the target side.

The results by Søgaard (2010) are based on the same methodology as ours, and even on some of the same data sets. However, his alignment reachability scores, repeated for convenience in table 4, are much lower than ours. We found that they are faulty, and that they therefore present a highly distorted picture about the need of more expressive translation models. The problem is caused, amongst others, by the fact that the implementation[3] used for his experiments handles unaligned words incorrectly. They are added deterministically to the first constituent that encounters them, which leads to false negatives as further explained in figure 37. After fixing this issue, the same results as for NF-SCFG are obtained. Another problem of the implementation concerns discontinuous translation units. Søgaard's alignment validation returns *false* if the words in the gap are aligned, although many of such configurations are in fact induced by unrestricted ITG (Søgaard and Wu, 2009, section 3.2.1). In summary, our presented results correct and extend the results provided by Søgaard (2010).

Both Søgaard and Wu (2009) and Søgaard and Kuhn (2009) are also interested in the empirical alignment complexity of manually aligned parallel corpora, but their approach differs from ours. They induce lower bounds on the translation unit error rate (TUER) of various synchronous context-free grammar formalisms: they identify alignment structures that cannot be induced with a particular formalism and then simply count how often those configurations occur in the data. For example, Søgaard and Kuhn (2009) report that, in the English-German data set from Padó and Lapata (2006), 1.75% of the translation units are involved in IO alignments, 0.45% in CDTUs and 0.05% in bonbon alignments. Søgaard and Kuhn (2009) report amongst others that between 1.6% (for Danish-English data) and 12.1% (for Danish-Spanish data) of all translation units are discontinuous and therefore not derivable by ITG in normal form.

Our finding of a new class of configurations beyond ITG, the so-called IO-DTUs (cf. section 3.1.3), furthermore means that lower

3 `http://cst.dk/anders/itg-search.html`, accessed in February 2013.

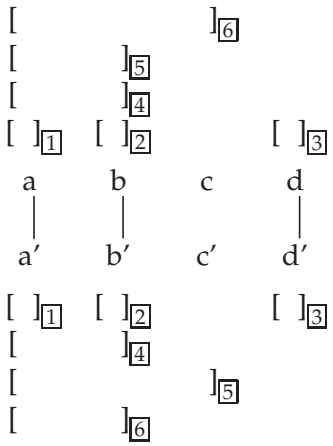


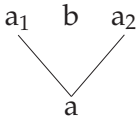
Figure 37: Synchronous ITG parse chart provided by the implementation from Søgaard (2010): Word c is already part of constituent 6, while c′ is part of constituent 5. When trying to combine, e. g., 4 and 3, c and c′ are not considered as unaligned because they are already part of a constituent. Consequently, neither 4 nor 5 nor 6 can be combined with 3 without creating a discontinuous constituent. The algorithm cannot find a larger continuous constituent, the alignment validation therefore returns *false*. However, this simple alignment structure clearly lies within the power of NF-ITG and ITG.

bounds on TUER for ITG are higher than reported in Søgaard and Wu (2009).

What we have termed alignment validation, is referred to as *parsing word alignments* by Maillette de Buy Wenniger and Sima'an (2013) and formalized therein. They also implement their approach for NF-ITG and provide empirical evidence for some data sets which we also used in our study, e. g. the Europarl data sets described in Graça et al. (2008a). The numbers they provide for alignment reachability (Table 1, column *GC discontiguous TEs*) are higher than what we reported for NF-SCFG. The reason is a different treatment of unaligned words. While we do not have a special treatment for unaligned words, i. e. they are monolingual translation units according to our definition, Maillette de Buy Wenniger and Sima'an (2013) argue that they would be attached to a neighboring translation unit, and they basically delete them before alignment validation. Ex. 3.4 demonstrates the difference. When deleting unaligned words from the Graça et al. (2008a) data set

and then performing our alignment validation algorithm, we obtain the same results as Maillette de Buy Wenniger and Sima'an (2013).

EXAMPLE 3.4 (Unaligned words and DTUs). This example shows a DTU with an unaligned word in the gap. Modeling unaligned words is allowed in all definitions of ITG and NF-ITG that are around. Nevertheless, strictly speaking, NF-ITG is not able to induce this alignment structure as it includes a DTU. ITG rules which induce the configuration are given below.



$$\langle X \rightarrow \mathrm{b} \quad , X \rightarrow \varepsilon \rangle$$
$$\langle X \rightarrow \mathrm{a}_1 X_{\boxed{1}} \, \mathrm{a}_2 \, , X \rightarrow X_{\boxed{1}} \, \mathrm{a} \rangle$$

By deleting the unaligned word b, the DTU is not discontinuous anymore and can be induced with the simple NF-ITG rule $\langle X \rightarrow \mathrm{a}_1\mathrm{a}_2, X \rightarrow \mathrm{a} \rangle$. □

Other studies that are loosely related to our work in this section are concerned with translation model search spaces (Zens and Ney, 2003; Dreyer et al., 2007) and translational equivalence modeling (Zhang et al., 2008a).

3.3 MANUAL ALIGNMENT INVESTIGATION

Knowing of the limitations of the alignment space of current translation models, their empirical adequacy has been put into question. As has been detailed in section 3.2, the empirical alignment capacity of various formalisms with respect to mostly manually aligned data has been investigated in different setups in several studies. We have confirmed for a wide range of language pairs and data sets that a proportion of the aligned sentence pairs cannot be induced with an SCFG of rank 2.

As we will see in later chapters, more expressive translation models that capture the complex alignment configurations come with further complications, such as the cost of higher decoding complexity, loss of translation speed or no tight probability estimators. A verification of the substantiality of the complex alignment configurations would serve as a justification for the use of more powerful translation models and further research in this area. To this end, it is necessary to exam-

ine instances of complex alignment configurations instead of merely relying on their frequency counts.

In this section, we thus investigate the nature of the complex alignment configurations that occur in hand-aligned data. We approach this task by manual categorization of the complex alignment configurations. Our categories address the issue of whether the involved alignments adhere to the corresponding alignment guidelines or whether they are annotation errors. We furthermore identify alignments that are correct with respect to the guidelines of the data set, but which could be questioned since other guidelines would align them in a different way, leading to fewer complex alignment configurations. We furthermore point out which linguistic phenomena cause the complex alignment configurations in our data, and we address the question of how necessary they are for translation. To the best of our knowledge, the available word alignment resources have not been studied with respect to these aspects so far.

3.3.1 *Alignment Data Sets and Guidelines*

In this investigation, we concentrate on data sets that we have also explored in the previous study on empirical alignment capacity: the manual alignments of 987 English-German (*en-de*) sentence pairs from Europarl, whose original purpose was the projection of semantic roles (Padó and Lapata, 2006), and the Europarl data sets described in Graça et al. (2008a) for the combinations of English (*en*), French (*fr*) and Spanish (*es*), each containing 100 sentence pairs.

Annotating word alignment is a non-trivial, complex and ambiguous task for a human annotator. Accordingly, different sets of alignment guidelines have been developed. The *en-de* data is aligned following the style guide of the *Blinker Project*, specified in Melamed (1998), which was originally concerned with English-French alignment. We will refer to it as the *Blinker* style guide in the following. No documentation is provided on how the guidelines were transferred to language specific, i. e. English-German, phenomena. Some other alignment projects have also used the Blinker guidelines as a starting point, e. g. Yadav and Gupta (2010). The *en-fr-es* data is aligned according to guidelines provided in Graça et al. (2008b). They are a refined version of the style guide for English-Spanish alignment by Lambert et al.

(2005). We will call them the *Graça* and the *Lambert* guidelines respectively.

The basic assumptions in the guidelines are similar. Their goal are full-text alignments, as opposed to sample word alignments. They strive to align units of the same meaning on both sides that are as small as possible, but that include as many words as necessary. While in the Blinker style guide only one type of alignment link is used, the *en-fr-es* data is aligned with S(ure) and P(ossible) links if the correspondence is valid in every respectively some context. However, since in our previous experiments and related work no distinction is made between the types of links, we will not differentiate between them either.

The style guides differ, of course, in many, sometimes language-specific, details. We only review those which we came across when studying the complex alignment configurations in the data.

ANAPHORA The Blinker guidelines specify that, if a pronoun occurs in one sentence with its antecedent and does not have a translation in the other language, both the antecedent and the pronoun are aligned to the translation of the antecedent. According to the Lambert guidelines, however, such anaphoric links between a pronoun in one language and a co-referent noun or proper noun in the other language are not licensed because they cannot be considered translations of each other.

REPETITIONS According to the Blinker guidelines, for repetitions that occur only in one language, but not in the other, all instances of the repetition are linked to the one translation. This stands in contrast to the Lambert guidelines which state that only the first instance of the repetition is aligned while the subsequent ones are without correspondence.

PUNCTUATION The style guides generally agree on how to align punctuation marks. However, the Blinker guidelines explicitly advise to align similar punctuation symbols which occur in different quantities on the two sides such that as few crossing links as possible arise. This can also mean that punctuation symbols remain unaligned. Even though the Lambert and Graça style guides do not contradict this

	en-de (30)	*en-fr*	*en-es*	*es-fr*
Sentence pairs	694	100	100	100
Sentence pairs with at least one compl. config.	116	5	4	9
IO	92	2	4	1
IO-DTU	38	1	0	2
CDTU	34	1	0	6
Bonbon	5	2	0	0
Multigap	8	0	0	0

Table 5: Data characteristics and frequency of the complex alignment configurations (number in parentheses: sentence length cut-off)

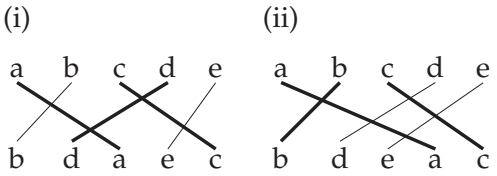


Figure 38: Example for the counting of complex alignment configurations for the manual classification. In (i), two IO configurations are counted: *a*, *b*, *c*, and *d*, and *a*, *c*, *d*, and *e*. In (ii), only one IO is counted as the links *d* and *e* are considered in one go. Bold links indicate overlapping links.

guideline, the *en-fr-es* data set by Graça et al. (2008a) contains alignments that do not adhere to it.

3.3.2 *Categories and Classification*

We automatically identified the complex alignment configurations in our data, leaving aside unaligned words. We then inspected each instance and classified it into one of the categories explained in the following. For visualization of the word alignments, we used the *Tree-Alignment Visualizer* (Maillette de Buy Wenniger et al., 2010).

Table 5 shows the number of analysed complex alignment configurations. Two overlapping configurations are counted as distinct here if the differing alignment link does not occupy the same position in

the configuration. See figure 38 for illustration.[4] The evaluators were three German native speakers with very good command of English, two of them also with good to excellent knowledge of French and Spanish, one of them being the author herself. Each complex alignment configuration was classified by two evaluators with command of the corresponding languages. Unclear cases were discussed until the evaluators agreed.[5] The categories for classification were the following:

ANNOTATION ERROR A link is an obvious annotation error if we can neither find a guideline that justifies the link nor think of any context in which the indicated translational equivalence would hold. By removing it, the alignment configuration is not complex anymore. See figure 39 for an example.[6]

ARTIFACT OF STYLE GUIDE A link is correct with respect to the style guide of the data (the Blinker guidelines for the *en-de* alignments for example). However, its existence is arguable since a different established style guide (the Lambert guidelines for example) would have aligned the phenomenon differently, thereby simplifying the configuration. The details about such differences in the style guides have been worked out in section 3.3.1. Figure 40 shows an example of anaphor alignment.

CORRECT ALIGNMENT All translation units that are part of the complex configuration are correctly aligned.

4 This is different than the methodology of Søgaard and Wu (2009) where configurations have to differ on *all* translation units to be counted as distinct configurations when they determine lower bounds on TUER. This is because in their scenario any of the links could be removed in order to simplify the overlapping configurations.

5 Note that none of the evaluators, including the author, were part of the original annotation body of any of the data sets. We have drawn our knowledge from the published annotation style guides and patterns observed in the annotated data. Our decisions have been made to the best of our knowledge on the basis of this information.

6 For the sake of clarity, all alignment figures in this section show only the links of the alignment which make up the complex alignment configuration. It should be assumed that the other words are aligned appropriately.

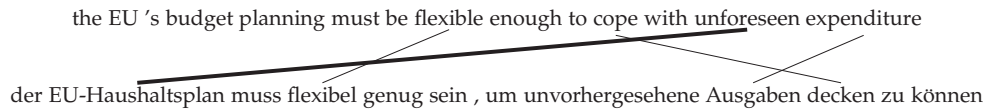


Figure 39: IO alignment due to an annotation error: *unforeseen* is aligned to *EU-Haushaltsplan* (*en-de*, sent. 183)

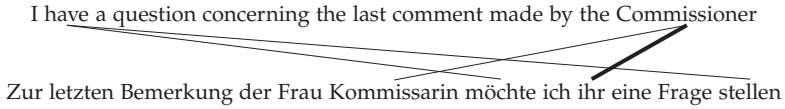


Figure 40: CDTU due to an artifact of the style guide: *Commissioner - ihr* is correctly aligned according to the Blinker guidelines (*en-de*, sent. 887)

Table 6 shows the classification results. If several overlapping configurations are caused by the same dubious link, we only count them once.

For all of the investigated data sets, no or very few complex alignment configurations are due to real annotation errors. In the *en-de* data, quite a large portion of complex configurations (32.8%) are caused by artifacts of the Blinker guidelines. Thereof 58.1% are due to anaphora, and 41.9% are due to repetitions. In the *en-fr-es* data, all complex configurations in the artifact category are due to questionable punctuation alignment.

The remaining complex alignment configurations, 55.7% in the *en-de* data and >83% in each of the *en-fr-es* data sets, see table 6, are those which are correctly aligned and therefore interesting for translation modeling. We will therefore further examine them in the following section.

3.3.3 *Phenomena*

First, we shed more light on which linguistic phenomena elicit the complex alignment configurations. This is of course dependent on the language pair. The issue of how important those configurations are for translation will also be considered.

			en-de (30)	*en-fr*	*en-es*	*es-fr*
(I)	ANNOTATION ERROR		0.115	0.000	0.000	0.000
(II)	ARTIFACT		0.328	0.167	0.000	0.111
(III)	CORRECT ALIGNMENT		0.557	0.833	1.000	0.889
	from (III)	IO	0.822	0.200	1.000	0.125
		IO-DTU	0.096	0.200	0.000	0.125
		CDTU	0.068	0.200	0.000	0.750
		Bonbon	0.014	0.400	0.000	0.000

Table 6: Classification results: ratio of the classes of complex alignment configurations

English-German

In the *en-de* data, most complex alignment configurations are caused by the different word orders of the English and German sentences. While English sentences follow subject-verb-object (SVO) order, the order of German constituents in a sentence is less rigid, traditionally described within the topological field model (Höhle, 1983).[7] In a nutshell, the positions of the verbs in a German sentence are fixed: in main clauses, the finite verb occupies the second position of the sentence, the *left bracket* (LB); in subordinate clauses, it occupies the final position, the *right bracket* (RB). Non-finite verbs are also located in the right bracket, but left of the finite verb if there is one. Argument and modifier constituents of the sentence are located between this verbal frame, in the *middle field*. However, in verb-second clauses, the initial position (the *initial field*) is filled by one constituent. The position right of the right bracket is called the *final field*. It can be occupied by certain types of large constituents to improve the comprehensibility of the sentence. The preference for a specific position and ordering of the arguments and modifiers is influenced by many syntactic and non-syntactic factors, e.g. pronominalization, information structure and pragmatic constraints. Certain word orders lead to IO alignments, as will be exemplified in the following.

Figure 41 shows an example of a German main clause, in which both sentence brackets are filled, showing how the auxiliary-participle com-

7 For an introduction to the topological field model, the reader is referred to Telljohann et al. (2012), section 3.1.

bination which is adjacent in English is placed in very distant parts of the German sentence. The subject *1813 Menschen* and the prepositional phrase *in 31 Ländern* are located in the middle field of the German sentence. Together with the verbs, they are ordered in such a way that an IO alignment is created.

Figure 42 shows a similar example, but where the German sentence is verb-final since it is a subordinate clause. The complementizer *dass* is usually analysed as filling the left bracket. Here, the subject (*die israelischen Streitkräfte*), a prepositional phrase (*aus einer Stadt*) and a reflexive pronoun (*sich*), elicited by the fact that the English verb *withdraw* translates into the German reflexive verb *sich zurückziehen*, are located in the German middle field. Together with the finite verb, the IO alignment is created. Even though not shown in the examples, the German initial field can also be involved in creating IO alignments.

A related phenomenon is the translation of an English verb into a German separable particle verb. While the core of the German finite verb remains in the left sentence bracket in main clauses, the particle is found at the end of the clause in the right bracket. Together with the English verb, this configuration forms a DTU which, in combination with the other arguments and modifiers of the clause, can lead to an IO alignment or IO-DTU. An example has already been shown in figure 36. The particle *an* of the German verb *angehen* is located in the right bracket.

While 56.5% of the aforementioned IO alignments and IO-DTUs occur with fairly literally translated translation units that differ in their ordering (as in the sentence pairs in figures 41, 42 and 43), the others are part of rather free translations or freely translated translation units, sometimes crossing clause boundaries. Figure 44 shows an example. Free translations also give rise to a few CDTUs and bonbon alignments.

A question that often arises when considering the complex alignment configurations is whether they are essential for translation. For the examples of IO/IO-DTU alignments due to different word orders of literally translated translation units on the clause level, we therefore additionally investigated the sentence pairs according to the following criterion: *Given one sentence of the sentence pair (e. g. English) as input to a translation system, and given that the translation model yields the provided target (e.g. German) translations of each translation unit, is the provided target (e. g. German) sentence the only valid translation?*

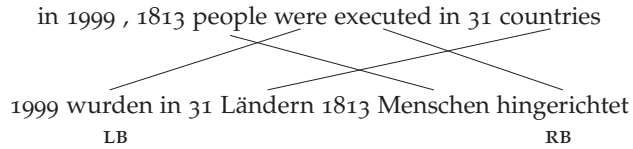


Figure 41: IO alignment due to different word orders; verb-second clause in German (*en-de*, sent. 119)

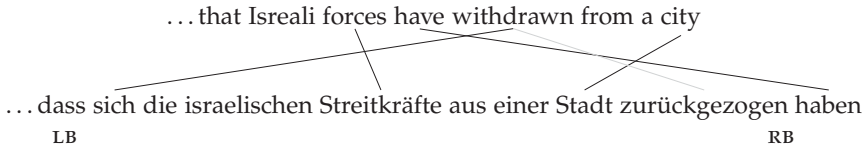


Figure 42: IO alignment due to different word orders; verb-final clause in German (*en-de*, sent. 306)

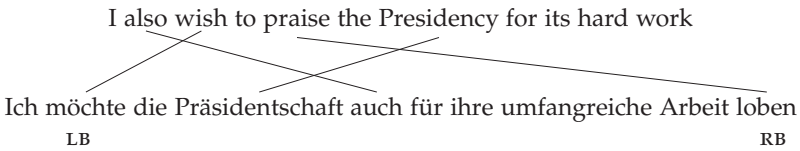


Figure 43: IO alignment due to different word orders, includes a focus adverb (*en-de*, sent. 389)

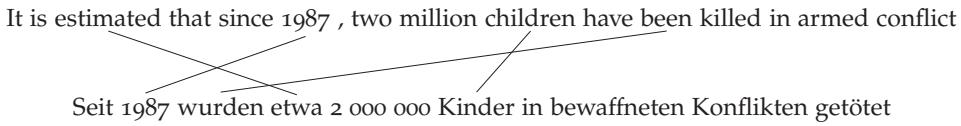


Figure 44: IO alignment due to different word orders in a rather free translation (*en-de*, sent. 876)

Let us consider figure 41 as an example. When translating from English to German, a German sentence with the provided translation units but with a simpler, i. e. non-complex, alignment could be produced, e. g. *1999 wurden 1813 Menschen in 31 Ländern hingerichtet*. The same holds for the other direction, e. g. *in 31 countries, 1813 people were executed in 1999*. In contrast to that, given the German sentence in figure 42 as input, and given that the model provides the shown English translation units, the English sentence in figure 42 is the only valid translation, due to the strict English word order. Only the prepositional phrase *from a city* could potentially be placed at the beginning of the clause. However, it is very marked in this position and it therefore obtains a strong focus which it does definitely not have in the original sentence.

Figure 43 shows an IO alignment which involves a focus adverb (*also/auch*). The complicacy of this is that it is often impossible to unambiguously determine the focus denoted by the adverb in a single sentence (Sudhoff, 2010). Since our data sets neither include context beyond the aligned sentence pair nor intonation information, we are not able to judge whether a reordering of the involved constituents would lead to an equally good translation (without complex alignment). In the given sentence pair, for the direction *en→de*, it is for instance grammatical to place *auch* at the very beginning or end of the middle field, which would simplify the configuration. However, this probably changes the focus, so we decide against making a statement about alternative translations via a different word ordering. A further complication is that in some sentence pairs, the focus denoted by a focus adverb in the English sentence is obviously different than in the German sentence. Negation particles and their scope are similar to the focus adverbs.

Under the given criterion, for 48.6% of the IO/IO-DTU alignments caused by word order, an equally good translation without complex configuration can be found when translating from English to German. For the remaining ones, a different translation would involve possible scope/focus changes. In the other direction (*de→en*), 31.4% of the cases can be simplified, and 28.6% involve a possible change in scope/focus. Remarkably, in 40% of the cases, the word order that involves the IO/IO-DTU configuration is the only possible one, under the criterion defined above.

Only 11 of the complex alignment configurations in category (III) are not due to the above explained word order phenomena. They arise from a variety of different phenomena, including local nominal discontinuities on both sides which lead to a bonbon alignment, infinitive clauses where *um ... zu* in German and *to* in English form a DTU in combination with a verbal DTU such that *zu* is in its gap creating a CDTU, and coordinations where the order within the conjuncts is different in English and German creating an IO or IO-DTU together with the verb.

English-French

The *en-fr* data of course includes the often cited bonbon alignment due to the French two-part negation that frames the finite verb, and a verbal DTU with the gap on the English side that is usually elicited by the use of an auxiliary because of the negation. Figure 45 (top half) shows an example from the data set.

If the French negation occurs in combination with a complex French verbal unit, e. g. in a compound tense, a CDTU is formed, since *ne* precedes the verbal translation unit and *pas* interrupts it.

CDTUs and bonbon alignments caused by the French negation can generally not be resolved by reordering in analogy to the IO/IO-DTU alignments, because of the strict rules of placement of negation in relation to verbs in the involved languages. For generating these configurations, translation models beyond phrase-based and SCFG-based models are thus necessary. Further considerations about the translation of these structures are presented in section 3.3.4.

The IO/IO-DTU configurations in the *en-fr* data are again caused by different word orders. They involve an adverb, which is placed at the beginning of the sentence in English, but between the finite and the non-finite verb in French compound tenses. They are either free translations, or, if not, a different word order without the complex configuration is possible.

English-Spanish

All complex alignment configurations in the *en-es* data are IO alignments. Most of them reside in rather free translations. One that is created by a relatively literal translation is shown in figure 46. It is caused by the different adjectival placement in English and Spanish

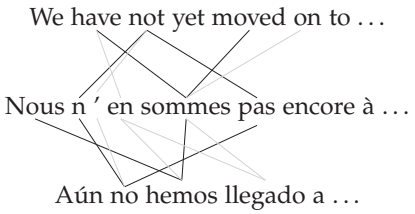


Figure 45: Bonbon alignment and CDTU due to French two-part negation (*en-fr-es*, sent. 13)

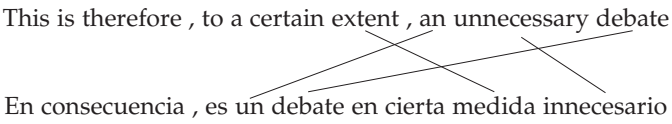


Figure 46: IO alignment due to modifier placement (*en-es*, sent. 59)

(pre-nominal vs. post-nominal) and the fact that the adverbial phrase *to a certain extent* is located on the clause level in English, but within the noun phrase in Spanish. In this case, a more direct translation that does not require an IO alignment is possible in both translation directions.

Spanish-French

In the *es-fr* data, all CDTUs involve a French two-part negation. We observe two different phenomena. First, if the French verb is complex, e. g. in a compound tense, the second part of the negation (*pas*) is placed after the finite verb, thus creating a CDTU. The same phenomenon is also found in the *en-fr* data, as previously described. Second, Spanish usually does not realize pronouns in subject position. The French pronoun is then aligned to the verbal translation unit. In combination with a negation, the first part of the negation (*ne*) interrupts this verbal translation unit, creating a CDTU. An example is shown in figure 45 in the lower half.

Just as in the *en-fr* data, the IO/IO-DTU configurations are also due to different word orders involving adverbial phrases. In the French sentences, the adverbials are placed between the finite and the infinite verb, while in Spanish they are found at the beginning of the sentence or on the right of the verb. One of the configurations occurs in a rather

free translation, the other in a sentence pair that could be reordered for a simpler alignment, but possibly involving a change of focus.

3.3.4 *Discussion*

The results of our manual investigation show that, while only very few of the complex alignment configurations are true annotation errors, many can be argued away with reference to other annotation guidelines. The remaining ones are those of interest for translation modeling.

Within the limited context of our analysis, we can certainly not generally answer how crucial it is for an SMT system to be expressive enough to induce complex alignment configurations. When thinking about this question, it is important to keep the following points in mind: Translation is concerned with producing a good/correct output string, not with producing a correct alignment. This means that generating a specific alignment is not necessarily important for generating a specific translation. It might even happen that a system produces a translation that corresponds to a complex alignment configuration on the surface, without actually having generated this complex configuration.

Furthermore, it is absolutely clear that, by paraphrasing, a good translation that is different from the one with the complex configuration present in our data can usually be found. It is not possible to answer the question of whether the translation in our data is the best from all possible translation options. Deciding whether one translation is better than another one is highly subjective and depends on many factors outside the context of one sentence.

What we argue for is that there exist good translation options that involve complex alignment configurations (category III). As we have found, many of them are fluent and relatively literal translations. A machine translation system should thus not exclude them a priori from the translation search space. This entails the usage of more expressive translation models than those based on 2-SCFG. In particular, when translating from a language with a rather free word order to a language with a rather rigid word order, it seems important to be able to induce IO and IO-DTU alignments: for 40% of the *en-de* complex configurations caused by different word orders of literal translation units, the English word order present in the data is the only possible

one when translating the German sentence. If focus and scope considerations are also taken into account, the number will certainly be higher.

For the *en→de* direction, we always found a reordered translation alternative without complex configuration (or a focus/scope ambiguity). This is not surprising due to the flexible word order of German. However, as already pointed out before, in a given context, pragmatic constraints can lead to a strong preference for one of the word orders. This can certainly be the one induced by the complex alignment configuration. The same holds for the instances of word order phenomena in the *en-fr-es* data sets: even though a translation with reordered translation units is possible, the other one might be the more canonical one in a certain situation. Thus, even for those language pairs and translation directions, there is reason to investigate more powerful translation models.

It should furthermore be noted that being able to induce the complex configurations is especially of importance if the involved construction is *productive*. This is certainly the case for the word order phenomena. If a construction is not productive, the translation model can just memorize the non-productive parts as a whole, without being able to induce each translation unit individually. As an example, consider the English-French bonbon alignments, caused by the French two-part negation and the requirement of an auxiliary verb by the English negation (figure 45). While the two DTUs are certainly perfectly aligned in terms of the style guides and in terms of lexical translational correspondence, one could argue that an SMT model could memorize all non- or less productive parts, i. e., the negation translation unit together with the English auxiliary. Combining this with the productive main verbal translation unit does not involve a bonbon anymore. Such considerations, however, come at the price of a less modular translation model.

For correctly aligned, but relatively free translations, we did not make statements about how essential generating the complex configuration is. The reason is twofold: First, a more literal translation could be produced. Second, in a machine translation system, ideally, free translations are not generated by composing individual translation units, but as larger structures/blocks, since it is only together that they make sense. This means that, in those cases, being able to generate the complex alignment configuration is less important. However,

especially if the free translation option involves productive parts, a perfect translation model should of course also be able to generate them.

3.4 CONCLUSION

This chapter started with a description of complex alignment configurations, i. e. configurations that cannot be induced with SCFGs of rank 2, the grammar formalism behind the most popular tree-based translation model. Those are inside-out alignments and certain configurations of discontinuous translation units.

We then presented a large study concerning the empirical alignment capacity of 2-SCFG which corrects previous similar studies and provides new results for language pairs that have not been tested before. We confirmed that normal-form SCFG is not adequate for representing the translational equivalence found in a variety of manually aligned corpora. For unrestricted 2-SCFG, our empirical investigation shows that more manual alignments can be captured than previously reported. However, there remains a portion of aligned sentence pairs that cannot be generated; for example more than 5% of the English-Dutch, Danish-German and English-German sentence pairs, and 9% of the Spanish-French sentence pairs.

The status of the alignment configurations beyond 2-SCFG has been debated in the SMT community. Several different empirical studies have investigated them quantitatively. We complemented this body of work with a small qualitative investigation of the complex alignment configurations in manually aligned data sets. While some of those configurations are caused by annotation errors or artifacts of a specific annotation style guide, we found that more than half of the complex configurations in the *en-de* data and between 83% and 100% in the *en-fr-es* data sets are indeed correctly aligned. Mostly, especially in the *en-de* data, it is the word order on the clausal level which leads to IO alignments and IO-DTUs. In the *en-fr* and *es-fr* data, the French two-part negation is often involved in CDTUs and bonbon alignments.

Even though the translations generated by the complex alignment configurations certainly do not represent the only translation options, they are correct, often fairly literal translations. Especially if they involve productive constructions, one should not exclude them a priori

from the translation search space. This motivates translation modeling beyond phrase-based and context-free grammars.

4

TRANSLATION MODELING BEYOND CONTEXT-FREE GRAMMAR

Chapter 3 has elaborated on the alignment configurations that are beyond the alignment capacity of Synchronous Context-Free Grammar (SCFG) of rank 2. In this chapter, we will introduce Synchronous Linear Context-Free Rewriting System (SLCFRS), a more powerful formalism which is able to model those alignment configurations.

The main ideas of this chapter have previously been published in a conference publication:

> Kaeshammer, M. (2013). Synchronous linear context-free rewriting systems for machine translation. In *Proceedings of the Seventh Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 68–77, Atlanta, Georgia. Association for Computational Linguistics.

4.1 INTRODUCTION

The *complex alignment configurations* are inside-out (IO) alignments, bonbon alignments and cross-serial discontinuous translation units (CDTUs) among others (p. 96). In order to induce them in a grammar-based framework, *discontinuous constituents* are necessary. Discontinuous constituents are constituents whose yield is interrupted by material which is not part of the constituent itself. To be able to talk about the alignment capacity of grammar formalisms, this chapter makes the same assumptions as the previous chapter, see section 3.1.1. In particular, it is assumed that the terminals generated by the application of one rewriting rule form one translation unit (i. e. they are aligned), and we require each translation unit to be generated by one (synchronous) rule for an adequate modeling of translational equivalence.

Figure 47 illustrates the necessity for discontinuous constituents. It shows sample derivations for three of the complex alignment configurations. Co-indexed non-terminals are generated synchronously. In (i),

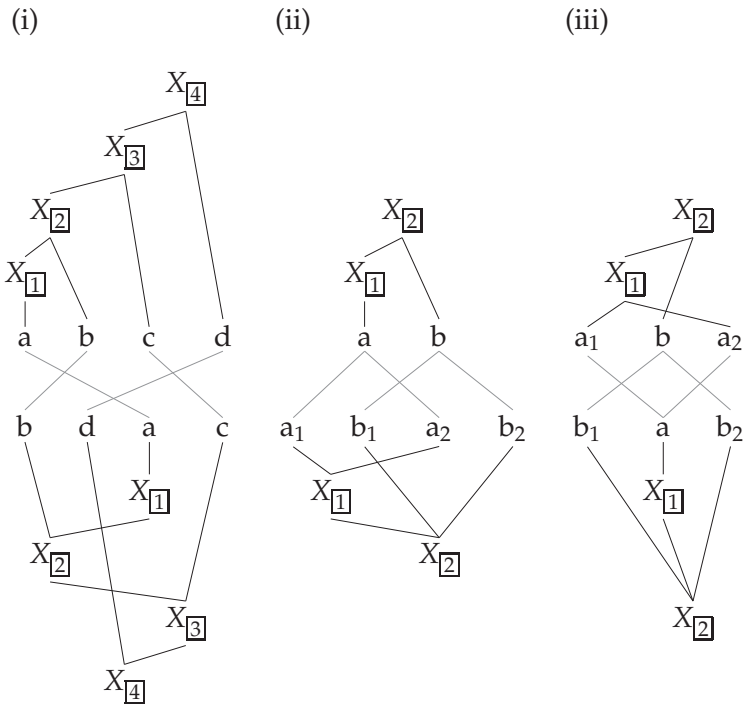


Figure 47: Synchronous derivations for an IO alignment (i), a CDTU (ii), and the bonbon alignment (iii).

$X_{\boxed{2}}$ and $X_{\boxed{3}}$ are discontinuous on the target side. $X_{\boxed{1}}$ is discontinuous on the target side in (ii) and on the source side in (iii). Note that many other derivations of maximally rank 2 which comply with our assumptions and which induce the same alignment structure are possible. However, all of them involve at least one discontinuous constituent.

We propose to augment tree-based approaches to statistical machine translation (SMT) (cf. section 2.2.5) such that they can account for discontinuous constituents in the source and/or the target derivation. In section 2.1.3, we have seen for monolingual syntactic descriptions that this implies going beyond the power of Context-Free Grammar. The same obviously holds for the synchronous case: synchronous derivations with possibly discontinuous synchronous constituents lie beyond the expressivity of SCFG. Linear Context-Free Rewriting System (LCFRS) has been established in the monolingual parsing community as an appropriate formalism for the modeling of discontinuous constituents (Maier and Lichte, 2011; Kuhlmann and Satta, 2009) (cf. sections 2.1.3 and 2.1.4). In particular, it has been shown that prob-

abilistic, data-driven parsing with highly ambiguous treebank grammars is feasible (e. g. Kallmeyer and Maier, 2013; van Cranenburgh, 2012). We will transfer those findings to statistical machine translation.

We are aware of the potential argument that this step is not necessary because SMT does not necessarily require the alignment modeling of each individual translation unit. Instead all translation units in one complex alignment configurations could be covered by one rewriting rule or phrase pair together, thereby defeating the requirement for more expressive alignment and translation models. This line of thought has already been picked up in section 3.3.4. We think that, since there is at least one interpretation in which current translation models are not powerful enough, it is worth pursuing the idea of using more expressive translation models.

4.2 SYNCHRONOUS LINEAR CONTEXT-FREE REWRITING SYSTEMS

Synchronous Linear Context-Free Rewriting System (SLCFRS) retains many characteristics of Synchronous Context-Free Grammar (SCFG), with the crucial difference being that pairs of tuples of strings are rewritten instead of pairs of strings.

4.2.1 *Formalism*

We define SLCFRS in parallel to SCFG (def. 2.60, p. 76). Rewriting rules consist of paired LCFRS rules (def. 2.29, p. 26) in which the right-hand side (RHS) non-terminals are synchronized by index annotation. Recall def. 2.32, p. 27 for range vectors.

DEFINITION 4.1 (Index annotation (LCFRS)). Let N be a set of non-terminal symbols and V be a set of variables. We define $\mathcal{I}(N) = \{A_{\boxed{k}} \mid A \in N, k \in \mathbb{N}\}$, and $I = \{A_{\boxed{k}}(\boldsymbol{\alpha}) \mid A_{\boxed{k}} \in \mathcal{I}(N)$, and $\alpha_i \in V$ for $1 \leq i \leq |\boldsymbol{\alpha}|, \alpha_i \neq \alpha_j$ for all $i \neq j$, or $\boldsymbol{\alpha}$ is a range vector$\}$. The set of all indices (integers k) that occur in $\gamma \in I^*$, is denoted by $ind(\gamma)$. Two strings γ_1, γ_2 are *synchronous* iff $ind(\gamma_1) = ind(\gamma_2)$, each index in $ind(\gamma_1)$ occurs exactly once in γ_1 and each index in $ind(\gamma_2)$ occurs exactly once in γ_2. γ_1 and γ_2 are *independent* iff $ind(\gamma_1) \cap ind(\gamma_2) = \emptyset$. □

DEFINITION 4.2 (Synchronous Linear Context-Free Rewriting System). A *Synchronous Linear Context-Free Rewriting System (SLCFRS)* is a tuple $G = (N_s, N_t, T_s, T_t, V_s, V_t, P, S_s, S_t)$ where

1. N_s and N_t are finite sets of source and target non-terminal symbols with a function $dim : N_i \to \mathbb{N}$ for $i \in \{s, t\}$, $N = N_s \cup N_t$,
2. T_s and T_t are finite sets of source and target terminals,
3. V_s and V_t are finite sets of source and target variables,
4. T_s and V_s, respectively T_t and V_t are disjoint,
5. $S_s \in N_s$ and $S_t \in N_t$ are distinguished start symbols,
6. P is a finite set of *synchronous (rewriting) rules* of the form

$$\langle A(\alpha_1, \ldots, \alpha_{dim(A)}) \to \gamma_s \,,\, B(\beta_1, \ldots, \beta_{dim(B)}) \to \gamma_t \rangle$$

with $A \in N_s$, $\alpha_i \in (T_s \cup V_s)^*$ for $1 \leq i \leq dim(A)$, $B \in N_t$, $\beta_i \in (T_t \cup V_t)^*$ for $1 \leq i \leq dim(B)$, and

$$\gamma_s = A_{1\boxed{l_1}}(Y_1^{(1)}, \ldots, Y_{dim(A_1)}^{(1)}) \cdots A_{m\boxed{l_m}}(Y_1^{(m)}, \ldots, Y_{dim(A_m)}^{(m)})$$

$$\gamma_t = B_{1\boxed{l_1}}(Z_1^{(1)}, \ldots Z_{dim(B_1)}^{(1)}) \cdots B_{m\boxed{l_m}}(Z_1^{(m)}, \ldots, Z_{dim(B_m)}^{(m)})$$

where, for $m \geq 0$,

a) $l_1, \ldots, l_m \in \mathbb{N}$,
b) $A_{1\boxed{l_1}}, \ldots, A_{m\boxed{l_m}} \in \mathcal{I}(N_s)$,
c) $B_{1\boxed{l_1}}, \ldots, B_{m\boxed{l_m}} \in \mathcal{I}(N_t)$,
d) $Y_j^{(i)} \in V_s$ for $1 \leq i \leq m$, $1 \leq j \leq dim(A_i)$, and
e) $Z_j^{(i)} \in V_t$ for $1 \leq i \leq m$, $1 \leq j \leq dim(B_i)$.

Furthermore, for all $\langle r_s, r_t \rangle \in P$, it holds that every variable $Y \in V_s$ that occurs in r_s occurs exactly once in the left-hand side (LHS) and exactly once in the RHS of r_s, and that every variable $Z \in V_t$ that occurs in r_t occurs exactly once in the LHS and exactly once in the RHS of r_t. □

Note that γ_s and γ_t in def. 4.2 are synchronous. SLCFRS are equivalent to Simple Range Concatenation Transducers (SRCTs) (Bertsch and Nederhof, 2001). See the related work in section 4.4.2.

In certain contexts, it is useful to consider the source side and the target side of a synchronous grammar separately.

DEFINITION 4.3 (Source and target projections (SLCFRS)). Let $G = (N_s, N_t, T_s, T_t, V_s, V_t, P, S_s, S_t)$ be an SLCFRS and $r = \langle r_s, r_t \rangle \in P$ a synchronous rule. The *source* and *target projection* of r are $proj_s(r) = r_s$ and $proj_t(r) = r_t$ respectively. The source and target projections of G, $proj_s(G)$ and $proj_t(G)$, are the LCFRSs $G_s = (N_s, T_s, V_s, P_s, S_s)$ and $G_t = (N_t, T_t, V_t, P_t, S_t)$ with the rule sets $P_s = \{proj_s(r) \mid r \in P\}$ and $P_t = \{proj_t(r) \mid r \in P\}$. □

Rank and *fan-out* are important properties of LCFRS as well as its synchronous counterpart SLCFRS.

DEFINITION 4.4 (Rank, fan-out (SLCFRS)). Let $G = (N_s, N_t, T_s, T_t, V_s, V_t, P, S_s, S_t)$ be an SLCFRS. As for LCFRS, the *rank* of a rule $r \in P$ as defined in def. 4.2 is m. The rank of G is the maximal rank of any of its rules $r \in P$. The *fan-out* v of G is the sum of the fan-outs of the source projection $G_s = proj_s(G)$ and the target projection $G_t = proj_t(G)$, $v = v_{G_s} + v_{G_t}$. We sometimes write $v_{v_{G_s}|v_{G_t}}$ to express how the fan-out of G is distributed over the source and the target side. As in the monolingual case, G is called a v-SLCFRS if it has fan-out v and a (u, v)-SLCFRS if it has rank u and fan-out v. □

In the context of parsing, it will be useful to assume *monotone* and *ε-free* grammars.

DEFINITION 4.5 (Monotone SLCFRS). An SLCFRS $G = (N_s, N_t, T_s, T_t, V_s, V_t, P, S_s, S_t)$ is *monotone* if its source projection $proj_s(G)$ as well as the target projection $proj_t(G)$ are monotone (def. 2.39, p. 32). □

DEFINITION 4.6 (ε-free SLCFRS). An SLCFRS $G = (N_s, N_t, T_s, T_t, V_s, V_t, P, S_s, S_t)$ is *ε-free* if its source projection $proj_s(G)$ as well as the target projection $proj_t(G)$ are ε-free (def. 2.40, p. 32). □

EXAMPLE 4.7 (Synchronous Linear Context-Free Rewriting System). Consider the following SLCFRS $G = (\{S, A, B\}, \{S, C, D\}, \{a, b, c, d\}, \{a, b, c, d\}, \{Y_1, Y_2, Y_3, Y_4\}, \{Z_1, Z_2, Z_3\}, P, S, S)$ where P contains the following rules:

$$\begin{aligned}
&\langle \mathrm{S}(Y_1Y_2Y_3Y_4) \to \mathrm{A}_{\boxed{1}}(Y_1, Y_3)\,\mathrm{B}_{\boxed{2}}(Y_2, Y_4)\ , && \\
&\qquad\qquad\qquad\qquad\qquad\qquad && \mathrm{S}(Z_1Z_2Z_3) \to \mathrm{C}_{\boxed{1}}(Z_1, Z_3)\,\mathrm{D}_{\boxed{2}}(Z_2)\rangle \\
&\langle \mathrm{A}(\mathrm{a}Y_1, \mathrm{c}Y_2) \to \mathrm{A}_{\boxed{1}}(Y_1, Y_2)\ , && \mathrm{C}(\mathrm{a}Z_1, Z_2\mathrm{c}) \to \mathrm{C}_{\boxed{1}}(Z_1, Z_2)\rangle \\
&\langle \mathrm{B}(\mathrm{b}Y_1, \mathrm{d}Y_2) \to \mathrm{B}_{\boxed{1}}(Y_1, Y_2)\ , && \mathrm{D}(\mathrm{b}Z_1\mathrm{d}) \to \mathrm{D}_{\boxed{1}}(Z_1)\rangle \\
&\langle \mathrm{A}(\mathrm{a}, \mathrm{c}) \to \varepsilon\ , && \mathrm{C}(\mathrm{a}, \mathrm{c}) \to \varepsilon\rangle \\
&\langle \mathrm{B}(\mathrm{b}, \mathrm{d}) \to \varepsilon\ , && \mathrm{D}(\mathrm{bd}) \to \varepsilon\rangle
\end{aligned}$$

The rank of G is 2 and its fan-out is $4_{2|2}$. It is monotone and ε-free. □

We will now introduce the notions of *derivation* and *translation* of SLCFRS. Intuitively, during a derivation step, the yields of two co-indexed non-terminals have to be explained from one synchronous rule. Given an input pair $\langle w_s, w_t \rangle$, the start of the derivation is

$$\langle S_{s\boxed{1}}(\langle 0, |w_s| \rangle), S_{t\boxed{1}}(\langle 0, |w_t| \rangle)\rangle,$$

i. e. the source start non-terminal S_s yielding w_s and the target start non-terminal S_t yielding w_t, and S_s and S_t are co-indexed. We will first establish the helpful concepts of *rule instantiation* and *reindexing* before moving on to the definitions of derivation and translation.

Definition 4.8 (Rule instantiation (SLCFRS)). Let $G = (N_s, N_t, T_s, T_t, V_s, V_t, P, S_s, S_t)$ be an SLCFRS and $\langle w_s, w_t \rangle \in T_s^* \times T_t^*$ a pair of strings. A rule $\langle r_s, r_t \rangle \in P$ is an *instantiated rule*

$$\langle A(\boldsymbol{\rho}) \to A_{1\boxed{l_1}}(\boldsymbol{\rho_1}) \dots A_{m\boxed{l_m}}(\boldsymbol{\rho_m}), B(\boldsymbol{\sigma}) \to B_{1\boxed{l_1}}(\boldsymbol{\sigma_1}) \dots B_{m\boxed{l_m}}(\boldsymbol{\sigma_m})\rangle$$

with respect to $\langle w_s, w_t \rangle$ if there exists an instantiation $\langle \boldsymbol{\phi}, \boldsymbol{\psi} \rangle$ for which the following holds when applying it to $\langle r_s, r_t \rangle$:

1. applying $\boldsymbol{\phi}$ to the source side rule r_s leads to an instantiated LCFRS rule $A(\boldsymbol{\rho}) \to A_1(\boldsymbol{\rho_1}) \dots A_m(\boldsymbol{\rho_m})$ with respect to w_s, and
2. applying $\boldsymbol{\psi}$ to the target side rule r_t leads to an instantiated LCFRS rule $B(\boldsymbol{\sigma}) \to B_1(\boldsymbol{\sigma_1}) \dots B_m(\boldsymbol{\sigma_m})$ with respect to w_t.

See def. 2.34, p. 28 for the definition of rule instantiation for LCFRS. □

DEFINITION 4.9 (Reindexing (SLCFRS)). A *reindexing* is an injective function $f : \mathbb{N} \to \mathbb{N}$. f is extended to I such that $f(A_{\boxed{k}}(\boldsymbol{\alpha})) = A_{\boxed{f(k)}}(\boldsymbol{\alpha})$, and to strings in I^* such that $f(\varepsilon) = \varepsilon$ and $f(X\Gamma) = f(X)f(\Gamma)$, for $X \in I$ and $\Gamma \in I^*$. □

DEFINITION 4.10 (Derivation (SLCFRS)). Let $G = (N_s, N_t, T_s, T_t, V_s, V_t, P, S_s, S_t)$ be an SLCFRS and $\langle w_s, w_t \rangle \in T_s^* \times T_t^*$ a pair of strings. Let $\Gamma_s, \Gamma_t \in I^*$ be synchronous.

1. $\Rightarrow_{G,\langle w_s,w_t\rangle}$ is a relation called *derives* between two pairs of synchronous strings of instantiated non-terminals. It is defined as follows:

   $$\langle \Gamma_s, \Gamma_t \rangle \Rightarrow_{G,\langle w_s,w_t\rangle} \langle \Delta_s, \Delta_t \rangle$$

   iff $\langle A(\boldsymbol{\rho}) \to \gamma_s, B(\boldsymbol{\sigma}) \to \gamma_t \rangle$ is an instantiated rule $r \in P$ with respect to $\langle w_s, w_t \rangle$, there exists an index k in $ind(\Gamma_s)$ and some reindexing f such that

   a) $f(\gamma_s\gamma_t)$ and $\Gamma_s\Gamma_t$ are independent, and

   b) $\Gamma_s = \Gamma_s' A_{\boxed{k}}(\boldsymbol{\rho})\Gamma_s''$, $\Delta_s = \Gamma_s' f(\gamma_s)\Gamma_s''$ and $\Gamma_t = \Gamma_t' B_{\boxed{k}}(\boldsymbol{\sigma})\Gamma_t''$, $\Delta_t = \Gamma_t' f(\gamma_t)\Gamma_t''$

   with $\Gamma_s, \Gamma_t, \Gamma_s', \Gamma_s'', \Gamma_t', \Gamma_t'', \Delta_s, \Delta_t \in I^*$.
   If G, w_s and w_t are given in the context, we can use $\Rightarrow$ instead of $\Rightarrow_{G,\langle w_s,w_t\rangle}$. To make the applied rule $r = \langle A(\boldsymbol{\alpha}) \to \gamma_s, B(\boldsymbol{\beta}) \to \gamma_t \rangle$ explicit, we can use $\Rightarrow_G^r$.

2. $\overset{*}{\Rightarrow}_{G,\langle w_s,w_t\rangle}$ is the reflexive transitive closure of $\Rightarrow_{G,\langle w_s,w_t\rangle}$.

3. For $1 \leq i \leq m$, let $r_i \in P$ be rules and let $\Gamma_s^{(i)}, \Gamma_t^{(i)} \in I^*$ be synchronous strings of instantiated non-terminals.

   $$\langle \Gamma_s^{(1)}, \Gamma_t^{(1)} \rangle \Rightarrow_{G,\langle w_s,w_t\rangle}^{r_1} \dots \Rightarrow_{G,\langle w_s,w_t\rangle}^{r_{m-1}} \langle \Gamma_s^{(m)}, \Gamma_t^{(m)} \rangle$$

   is a *derivation* of length m. □

In the following, we will sometimes write $r \in d$ to denote the rules $r_1, \dots, r_{m-1}$ which constitute a derivation d of length m. It should however be clear that d is not just a set of rules. In fact, it is not a set at all since rules can occur more than once in a single derivation.

An SLCFRS derivation can be represented as a pair of two LCFRS derivation trees (def. 2.37, p. 30). The synchronization of two non-terminals that have been derived in parallel is usually shown by index annotation (indices k after applying the reindexing function). Alternatively the synchronized non-terminals can be linked with lines, just as it is sometimes done for SCFG derivation trees (p. 78).

We define the *translation* of an SLCFRS in analogy to the language of an LCFRS.

DEFINITION 4.11 (Translation (SLCFRS)). Let $G = (N_s, N_t, T_s, T_t, V_s, V_t, P, S_s, S_t)$ be an SLCFRS.

1. The *translation* generated by G is a binary relation over $T_s^* \times T_t^*$: $\mathcal{T}(G) = \{\langle w_s, w_t\rangle \mid \langle S_{s\boxed{1}}(\langle 0, |w_s|\rangle), S_{t\boxed{1}}(\langle 0, |w_t|\rangle)\rangle \overset{*}{\Rightarrow}_{G,\langle w_s,w_t\rangle} \langle\varepsilon,\varepsilon\rangle, w_s \in T_s^*, w_t \in T_t^*\}$. We also call $\mathcal{T}(G)$ the *language* of G.

2. Let $w = w_1 \dots w_n$ be a string. The set of *translations* of w generated by G is the following: $\mathcal{T}(G, w) = \{w_t \mid \langle w, w_t\rangle \in \mathcal{T}(G)\}$. □

EXAMPLE 4.12 (Derivation (SLCFRS)). Let G be the SLCFRS in ex. 4.7, p. 133. The translation or language of G is

$$\mathcal{T}(G) = \{\langle a^n b^m c^n d^m, a^n b^m d^m c^n\rangle \mid n, m > 0\},$$

i. e. G translates cross-serial dependencies into nested ones. As an example, we give the derivation of $\langle$aabccd, aabdcc$\rangle$ under G. Figure 48 shows the corresponding derivation tree pair.

$$\begin{aligned}
&\langle S_{\boxed{1}}(\langle 0,6\rangle)\ ,\ S_{\boxed{1}}(\langle 0,6\rangle)\rangle && \Rightarrow\\
&\langle A_{\boxed{2}}(\langle 0,2\rangle,\langle 3,5\rangle)B_{\boxed{3}}(\langle 2,3\rangle,\langle 5,6\rangle)\ ,\ C_{\boxed{2}}(\langle 0,2\rangle,\langle 4,6\rangle)D_{\boxed{3}}(\langle 2,4\rangle)\rangle && \Rightarrow\\
&\langle A_{\boxed{4}}(\langle 1,2\rangle,\langle 4,5\rangle)\ ,\ C_{\boxed{4}}(\langle 1,2\rangle,\langle 4,5\rangle)\rangle && \Rightarrow\\
&\langle\varepsilon\ ,\ \varepsilon\rangle
\end{aligned}$$

□

An alternative to the definition of SLCFRS for translation modeling beyond SCFG would have been to use an LCFRS in which the fan-out of each non-terminal is ≥ 2 with a dedicated source/target argument boundary. The synchronization would then be formulated between the arguments of the non-terminals. This is an extension to viewing

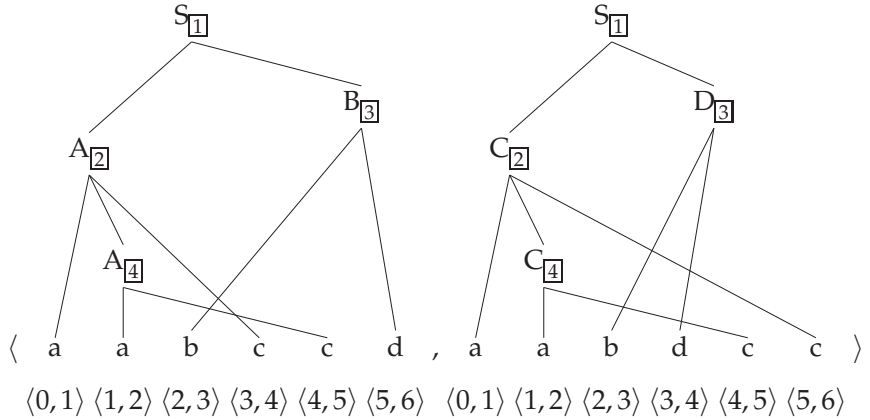


Figure 48: SLCFRS derivation tree pair, including ranges, for ex. 4.12

SCFG as LCFRS of fan-out 2 (see p. 81). However, the direct LCFRS formalization seems to be less perspicuous than SLCFRS, when moving from SCFG to mild context-sensitivity. Synchronous modeling on the non-terminal level is a very straight forward approach. Another disadvantage is that this sort of modeling with LCFRS requires $N_s = N_t$.

The relationship between Context-Free Grammar (CFG) and LCFRS naturally carries over to the synchronous case: A $2_{1|1}$-SLCFRS is strongly equivalent to an SCFG, i. e. an SCFG is an SLCFRS with a source and target fan-out of 1 respectively. While Inversion Transduction Grammar (ITG) (def. 2.67, p. 79) constrains the order of the non-terminals on the RHS of the target projection of a rule to be in the same or exactly in the reverse order compared to the non-terminals in the RHS of the source projection, SLCFRS does not impose such ordering constraints on its variables. However, it is obvious that a $(2, 2_{1|1})$-SLCFRS is equivalent to an ITG of rank 2.

In correspondence to SCFG and Normal-form Synchronous Context-Free Grammar (NF-SCFG) (p. 80), the notion of a *normal form* is also introduced for SLCFRS.

Definition 4.13 (Normal form (SLCFRS)). Let $G = (N_s, N_t, T_s, T_t, V_s, V_t, P, S_s, S_t)$ be an SLCFRS. G is in *normal form* if the following holds:

1. for all $\langle A(\alpha_1, \ldots, \alpha_{dim(A)}) \to \gamma_s, B(\beta_1, \ldots, \beta_{dim(B)}) \to \gamma_t \rangle \in P$, it holds that either $\alpha_i \in V_s^*$ and $\beta_j \in V_t^*$ or $\alpha_i \in T_s^*$ and $\beta_j \in T_t^*$ for $1 \leq i \leq dim(A)$ and $1 \leq j \leq dim(B)$; i. e. the arguments of

the LHS non-terminals consist either of variables or of terminals, but not of a mixture of both, and

2. the rank of G is at most 2.

We then call G a Normal-form Synchronous Linear Context-Free Rewriting System (NF-SLCFRS). □

NF-SCFG and $2_{1|1}$-SLCFRS in normal form are equivalent. Generally, to transform an SLCFRS G into an equivalent NF-SLCFRS G', two major steps are necessary to fulfill the two conditions of def. 4.13. First, in mixed rules, i. e. in rules where the source and/or the target LHS non-terminal contains variables as well as terminal symbols, a new synchronous RHS non-terminal is introduced which captures all lexical material. Second, the rules of G are binarized. The procedure for that is a combination of the binarization of LCFRS (see e. g. Kallmeyer (2010), p. 147) and the binarization of SCFG (see e. g. Zhang et al. (2006). Roughly speaking, in each rule with rank > 2 a new non-terminal and corresponding rule is introduced which covers all but one synchronous RHS non-terminal. If G has fan-out v, then the fan-out of G' will be $v' = v + c$, $c \in \mathbb{N}_0$, where the value of c depends on the chosen binarization strategy and the grammar G itself. Since the SLCFRSs in the context of this thesis are always guaranteed to be in normal form or of rank 2 by means of their design, the detailed algorithm for the transformation to normal form is not provided here.

EXAMPLE 4.14 (NF-SLCFRS). The SLCFRS G in ex. 4.7 is not in normal form, while the following SLCFRS G' with $\mathcal{T}(G) = \mathcal{T}(G')$ is. We only provide the rules in P. The remainder of the definition of G' is trivial.

$$\langle \mathrm{S}(Y_1Y_2Y_3Y_4) \to \mathrm{A}_{\boxed{1}}(Y_1, Y_3)\,\mathrm{B}_{\boxed{2}}(Y_2, Y_4)\ , \quad \mathrm{S}(Z_1Z_2Z_3) \to \mathrm{C}_{\boxed{1}}(Z_1, Z_3)\,\mathrm{D}_{\boxed{2}}(Z_2)\rangle$$

$$\langle \mathrm{A}(Y_1Y_2, Y_3Y_4) \to \mathrm{A}_{\boxed{1}}(Y_2, Y_4)\,\mathrm{A}'_{\boxed{2}}(Y_1, Y_3)\ , \quad \mathrm{C}(Z_1Z_2, Z_3Z_4) \to \mathrm{C}_{\boxed{1}}(Z_2, Z_3)\,\mathrm{C}'_{\boxed{2}}(Z_1, Z_4)\rangle$$

$$\langle \mathrm{B}(Y_1Y_2, Y_3Y_4) \to \mathrm{B}_{\boxed{1}}(Y_2, Y_4)\,\mathrm{B}'_{\boxed{2}}(Y_1, Y_3)\ , \quad \mathrm{D}(Z_1Z_2Z_3) \to \mathrm{D}_{\boxed{1}}(Z_2)\,\mathrm{D}'_{\boxed{2}}(Z_1, Z_3)\rangle$$

$$\langle \mathrm{A}(\mathrm{a}, \mathrm{c}) \to \varepsilon\ , \quad \mathrm{C}(\mathrm{a}, \mathrm{c}) \to \varepsilon\rangle$$

$$\langle \mathrm{A}'(\mathrm{a}, \mathrm{c}) \to \varepsilon\ , \quad \mathrm{C}'(\mathrm{a}, \mathrm{c}) \to \varepsilon\rangle$$

$$\langle \mathrm{B}(\mathrm{b}, \mathrm{d}) \to \varepsilon\ , \quad \mathrm{D}(\mathrm{bd}) \to \varepsilon\rangle$$

$$\langle \mathrm{B}'(\mathrm{b}, \mathrm{d}) \to \varepsilon\ , \quad \mathrm{D}'(\mathrm{b}, \mathrm{d}) \to \varepsilon\rangle$$

The non-terminals A′ and B′ on the source side and C′ and D′ on the target side have been newly introduced in comparison to G in order to remove mixed rules. This grammar already has rank 2, thus no further binarization is required. □

4.2.2 *Alignment Capacity*

In this section, it is demonstrated that SLCFRS is able to induce the complex alignment configurations. In the following, grammar fragments are presented which generate the hierarchical alignments (i), (ii) and (iii) in figure 47.

Example 4.15 (SLCFRS rules for the IO alignment (i)).

$$\langle X(\mathrm{a}) \to \varepsilon \quad , X(\mathrm{a}) \to \varepsilon \rangle$$
$$\langle X(Y_1\mathrm{b}) \to X_{\boxed{1}}(Y_1) \, , X(\mathrm{b}, Z_1) \to X_{\boxed{1}}(Z_1) \rangle$$
$$\langle X(Y_1\mathrm{c}) \to X_{\boxed{1}}(Y_1) \, , X(Z_1, Z_2\mathrm{c}) \to X_{\boxed{1}}(Z_1, Z_2) \rangle$$
$$\langle X(Y_1\mathrm{d}) \to X_{\boxed{1}}(Y_1) \, , X(Z_1\mathrm{d}Z_2) \to X_{\boxed{1}}(Z_1, Z_2) \rangle$$

Each rule generates one translation unit of the IO alignment. The second and the third rule are used to derive constituents which are discontinuous on the target side. □

Example 4.16 (SLCFRS rules for the CDTU (ii)).

$$\langle X(\mathrm{a}) \to \varepsilon \quad , X(\mathrm{a}_1, \mathrm{a}_2) \to \varepsilon \rangle$$
$$\langle X(Y_1\mathrm{b}) \to X_{\boxed{1}}(Y_1) \, , X(Z_1\mathrm{b}_1 Z_2\mathrm{b}_2) \to X_{\boxed{1}}(Z_1, Z_2) \rangle$$

The first rule derives the translation unit $\langle \mathrm{a}; \mathrm{a}_1, \mathrm{a}_2 \rangle$ which is discontinuous on the target side. The second rule generates the translation unit $\langle \mathrm{b}; \mathrm{b}_1, \mathrm{b}_2 \rangle$. □

Example 4.17 (SLCFRS rules for the bonbon alignment (iii)).

$$\langle X(\mathrm{a}_1, \mathrm{a}_2) \to \varepsilon \quad , X(\mathrm{a}) \to \varepsilon \rangle$$
$$\langle X(Y_1\mathrm{b}Y_2) \to X_{\boxed{1}}(Y_1, Y_2) \, , X(\mathrm{b}_1 Z_1 \mathrm{b}_2) \to X_{\boxed{1}}(Z_1) \rangle$$

The first rule derives the translation unit $\langle \mathrm{a}_1, \mathrm{a}_2; \mathrm{a} \rangle$ which is discontinuous on the source side. The second rule contributes the translation unit $\langle \mathrm{b}; \mathrm{b}_1, \mathrm{b}_2 \rangle$. □

As has been indicated before, generally, the same word alignment configuration can be induced by a set of different hierarchical alignments. An alternative set of rules for the CDTU configuration is presented in ex. 4.18 and for the bonbon alignment in ex. 4.19.

EXAMPLE 4.18 (Alternative SLCFRS rules for the CDTU (cf. ex. 4.16)).

$$\langle X(b) \to \varepsilon \qquad , X(b_1, b_2) \to \varepsilon \rangle$$
$$\langle X(aY_1) \to X_{\boxed{1}}(Y_1) \, , X(a_1 Z_1 a_2 Z_2) \to X_{\boxed{1}}(Z_1, Z_2) \rangle$$

The first rule derives the translation unit $\langle b; b_1, b_2 \rangle$, while the second rule derives the translation unit $\langle a; a_1, a_2 \rangle$. □

EXAMPLE 4.19 (Alternative SLCFRS rules for the bonbon alignment (cf. ex. 4.17)).

$$\langle X(b) \to \varepsilon \qquad , X(b_1, b_2) \to \varepsilon \rangle$$
$$\langle X(a_1 Y_1 a_2) \to X_{\boxed{1}}(Y_1) \, , X(Z_1 a Z_2) \to X_{\boxed{1}}(Z_1, Z_2) \rangle$$

The first rule derives the translation unit $\langle b; b_1, b_2 \rangle$, while the second rule generates the translation unit $\langle a_1, a_2; a \rangle$. □

For the IO alignment, many more derivations than the one presented in figure 47 (i) are possible: there exist 4! possibilities to combine the four translation units in a binary way. One such alternative is presented in ex. 4.20. Further combinations are possible if we do not restrict the space of grammars to those of rank 1. A grammar of rank 2 which also generates the same IO alignment is provided in ex. 4.21. The corresponding derivations are depicted in figure 49.

EXAMPLE 4.20 (Alternative SLCFRS rules (rank 1) for the IO alignment (cf. ex. 4.15)).

$$\langle X(c) \to \varepsilon \qquad , X(c) \to \varepsilon \rangle$$
$$\langle X(Y_1 d) \to X_{\boxed{1}}(Y_1) \qquad , X(d, Z_1) \to X_{\boxed{1}}(Z_1) \rangle$$
$$\langle X(a, Y_1) \to X_{\boxed{1}}(Y_1) \qquad , X(Z_1 a Z_2) \to X_{\boxed{1}}(Z_1, Z_2) \rangle$$
$$\langle X(Y_1 b Y_2) \to X_{\boxed{1}}(Y_1, Y_2) \, , X(b Z_1) \to X_{\boxed{1}}(Z_1) \rangle$$

□

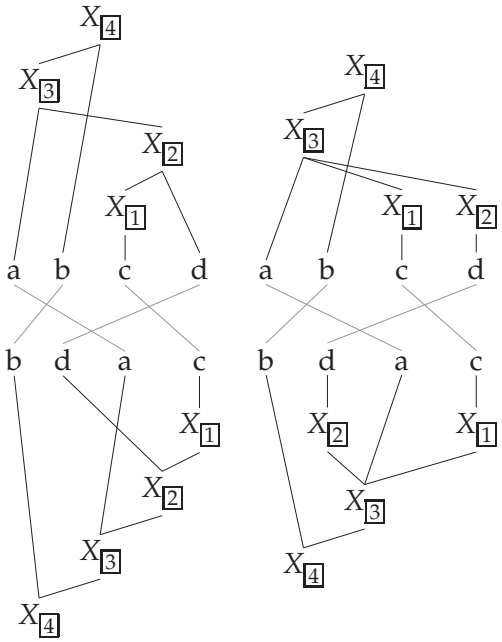


Figure 49: Two alternatives to the derivation in figure 47(i) for the IO alignment. They correspond to the rules provided in ex. 4.20 and ex. 4.21 respectively.

EXAMPLE 4.21 (Alternative SLCFRS rules (rank 2) for the IO alignment (cf. ex. 4.15)).

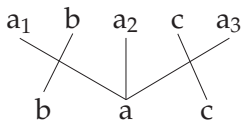
$$\langle X(\mathrm{c}) \to \varepsilon \qquad\qquad , X(\mathrm{c}) \to \varepsilon \rangle$$
$$\langle X(\mathrm{d}) \to \varepsilon \qquad\qquad , X(\mathrm{d}) \to \varepsilon \rangle$$
$$\langle X(\mathrm{a}, Y_1 Y_2) \to X_{\boxed{1}}(Y_1) X_{\boxed{2}}(Y_2) \, , X(Z_1 \mathrm{a} Z_2) \to X_{\boxed{1}}(Z_1) X_{\boxed{2}}(Z_2) \rangle$$
$$\langle X(Y_1 \mathrm{b} Y_2) \to X_{\boxed{1}}(Y_1, Y_2) \qquad , X(\mathrm{b} Z_1) \to X_{\boxed{1}}(Z_1) \rangle$$

□

It has been noted that ITG and Normal-form Inversion Transduction Grammar (NF-ITG) do not generate the same class of alignments (Søgaard and Wu, 2009). The same holds for SLCFRS: a v-SLCFRS in normal form does not generate the same class of alignments as an unrestricted $(2, v)$-SLCFRS. For illustration, consider a discontinuous translation unit a with two gaps on the source side, e. g. $\langle \mathrm{a}_1, \mathrm{a}_2, \mathrm{a}_3; \mathrm{a} \rangle$, and an SLCFRS G with fan-out $v = 3_{2|1}$. If G is in normal form, it cannot induce a. This is because a non-terminal of source fan-out 3 would be required to induce a. Generally, for generating x gaps with a

rule in normal form, a LHS non-terminal of fan-out $x + 1$ is required. However, if G is not constrained to the normal form, it is possible that G induces a by a rule which generates all terminals of a and combines them with some other constituents that fill at least one of the two gaps. This case is shown in ex. 4.22.

EXAMPLE 4.22 (Alignment capacity of NF-SLCFRS). Consider the alignment structure below. It consists of three translation units, one of them having two gaps on the source side. The shown rules provide one possibility to derive this alignment structure. This grammar fragment has a fan-out of $v = 3_{2|1}$ and is not in normal form.



$$\langle X(\mathrm{b}) \to \varepsilon \quad , X(\mathrm{b}) \to \varepsilon \rangle$$
$$\langle X(\mathrm{a}_1 Y_1 \mathrm{a}_2, \mathrm{a}_3) \to X_{\boxed{1}}(Y_1) \, , X(Z_1 \mathrm{a}) \to X_{\boxed{1}}(Z_1) \rangle$$
$$\langle X(Y_1 \mathrm{c} Y_2) \to X_{\boxed{1}}(Y_1, Y_2) \, , X(Z_1 \mathrm{c}) \to X_{\boxed{1}}(Z_1) \rangle$$

To derive the alignment structure with a NF-SLCFRS the rule

$$\langle X(\mathrm{a}_1, \mathrm{a}_2, \mathrm{a}_3) \to \varepsilon, X(\mathrm{a}) \to \varepsilon \rangle$$

would be required. It has fan-out $v = 4_{3|1}$, i. e. this alignment structure cannot be induced by an $3_{2|1}$-NF-SLCFRS. □

4.2.3 *Parsing*

We now consider bitext parsing with SLCFRS. The presented parser is an extended version of the CYK parser for LCFRS lined out in section 2.1.4. The addition is that one has to keep track of the source and the target side.

Deduction rules

A pair $\langle w_s, w_t \rangle \in T_s^* \times T_t^*$ is the input. The task is to decide whether a given SLCFRS $G = (N_s, N_t, T_s, T_t, V_s, V_t, P, S_s, S_t)$ generates $\langle w_s, w_t \rangle$, i. e. whether $\langle w_s, w_t \rangle \in \mathcal{T}(G)$. The items have the form $[A, \boldsymbol{\rho}, B, \boldsymbol{\sigma}]$ where $A \in N_s$ is a source non-terminal, $B \in N_t$ is a target non-terminal, $\boldsymbol{\rho}$ is a $dim(A)$-dimensional range vector in w_s and $\boldsymbol{\sigma}$ is a $dim(B)$-dimensional range vector in w_t.

The SCAN deduction rule creates items for terminal rules, i. e. rules where the source and the target RHS are ε.

$$\frac{}{[A, \boldsymbol{\rho}, B, \boldsymbol{\sigma}]}$$

The side conditions are the following:

1. $\langle r_s, r_t \rangle = \langle A(\boldsymbol{\alpha}) \to \varepsilon, B(\boldsymbol{\beta}) \to \varepsilon \rangle \in P$, and
2. there exists an instantiation $\langle \phi, \psi \rangle$ with respect to $\langle w_s, w_t \rangle$ s.t. $\langle \phi, \psi \rangle(\langle r_s, r_t \rangle)$ yields the instantiated rule $\langle A(\boldsymbol{\rho}) \to \varepsilon, B(\boldsymbol{\sigma}) \to \varepsilon \rangle$.

By definition of the instantiation it then holds that $\boldsymbol{\rho}(w_s) = \boldsymbol{\alpha}$ and $\boldsymbol{\sigma}(w_t) = \boldsymbol{\beta}$.

The COMPLETE operation combines already established items to a new item:

$$\frac{[A_1, \boldsymbol{\rho}_1, B_1, \boldsymbol{\sigma}_1] \dots [A_m, \boldsymbol{\rho}_m, B_m, \boldsymbol{\sigma}_m]}{[A, \boldsymbol{\rho}, B, \boldsymbol{\sigma}]}$$

The side conditions are the following:

1. $\langle r_s, r_t \rangle \in P$ with $r_s = A(\boldsymbol{\alpha}) \to A_{1\boxed{l_1}}(\boldsymbol{\alpha}_1) \dots A_{m\boxed{l_m}}(\boldsymbol{\alpha}_m)$ and $r_t = B(\boldsymbol{\beta}) \to B_{1\boxed{l_1}}(\boldsymbol{\beta}_1) \dots B_{m\boxed{l_m}}(\boldsymbol{\beta}_m)$, and
2. there exists an instantiation $\langle \phi, \psi \rangle$ with respect to $\langle w_s, w_t \rangle$ s.t. $\langle \phi, \psi \rangle(\langle r_s, r_t \rangle)$ yields the instantiated rule

$$\langle A(\boldsymbol{\rho}) \to A_{1\boxed{l_1}}(\boldsymbol{\rho}_1) \dots A_{m\boxed{l_m}}(\boldsymbol{\rho}_m), B(\boldsymbol{\sigma}) \to B_{1\boxed{l_1}}(\boldsymbol{\sigma}_1) \dots B_{m\boxed{l_m}}(\boldsymbol{\sigma}_m) \rangle$$

The GOAL item is an item which covers the complete input pair:

$$[S_s, \langle\langle 0, |w_s| \rangle\rangle, S_t, \langle\langle 0, |w_t| \rangle\rangle]$$

Complexity

Parsing with LCFRS can be performed in $\mathcal{O}(n^{v(u+1)})$, where n is the length of the input, u the rank and v the fan-out of the grammar (section 2.1.4). This result can be transferred to bitext parsing with SLCFRS as lined out in the following.

The most complex case of CYK bitext parsing for a $(u, v_{v_s|v_t})$-SLCFRS is the COMPLETE operation with a rule $r = \langle r_s, r_t \rangle$ which has rank u and all non-terminals occurring in r_s have a fan-out of

v_s and all non-terminals occurring in r_t have a fan-out of v_t. In this case, uv_s source variables from the RHS are combined into v_s arguments on the LHS, and uv_t target variables are combined into v_t arguments. This amounts to $uv_s + v_s + uv_t + v_t = v_s(u+1) + v_t(u+1) = (v_s + v_t)(u+1)$ independent indices. Indices on the source side range from 1 to $|w_s|$ and on the target side from 1 to $|w_t|$. With this we obtain a time complexity of $\mathcal{O}(|w_s|^{v_s(u+1)} |w_t|^{v_t(u+1)})$. Given our definition of $v = v_s + v_t$ and with the approximation $n = \max(|w_s|, |w_t|)$, we can simplify the expression to $\mathcal{O}(n^{v(u+1)})$.

The same complexity result is achieved when using LCFRS with a dedicated source/target argument boundary instead of SLCFRS for translation modeling (cf. p. 136). An SLCFRS G_1 with fan-out $v_{v_s|v_t}$ is essentially an LCFRS G_2 with fan-out v. The special argument boundary separates the source side arguments (of maximal size v_s) from the target side arguments (of maximal size v_t) of the non-terminals of G_2. Source side arguments are only matched against w_s, while target side arguments are only matched against w_t. Then the same reasoning as for SLCFRS above holds, leading to the known complexity of $\mathcal{O}(n^{v(u+1)})$, with the approximation $n = \max(|w_s|, |w_t|)$.

Note that when using SLCFRS as the base formalism for machine translation, bitext parsing is not performed. In order to obtain the set of translations of a given string w_s generated by an SLCFRS G, monolingual LCFRS parsing using the source projection grammar G_s is conducted. By keeping track of the applied rules, source derivations can be mapped to target derivations as a post-processing step, thereby providing the translations. This method is the same as for translating with SCFG (section 2.2.5) and will be elaborated on extensively in section 5.3.

4.3 EMPIRICAL INVESTIGATION

We have just seen that the complexity for bitext parsing and for translating with SLCFRS is determined by the fan-out v and the rank u of the grammar (section 4.2.3). For the very same reason, SCFGs used for SMT are usually of rank $u = 2$ (section 2.2.5). If we also restrict the space of translation grammars to consider to SLCFRSs of maximally rank $u = 2$, then the fan-out v of the grammar is the key factor for the parsing complexity.

In this section, an empirical investigation is presented. It aims at finding out which fan-out v is necessary to fully cover the alignment configurations that occur in manually aligned parallel corpora. It continues the investigation presented in section 3.2, which studied the alignment capacity of 2-SCFG by means of alignment validation.

4.3.1 *Alignment Validation*

Recall the bottom-up hierarchical aligner presented in section 3.2.1. We use a modified version it. The all-accepting grammar for our new scenario is an $v_{v_s|v_t}$-SLCFRS.

EXAMPLE 4.23 (All-accepting grammar). This example shows the rules for an all-accepting SLCFRS $G = (N_s, N_t, T_s, T_t, V_s, V_t, P, M_s^{(1)}, M_t^{(1)})$ in normal form with fan-out $v_{v_s|v_t}$. $M_s^{(k)}, A_s^{(k)} \in N_s$ and $M_t^{(l)}, A_t^{(l)} \in N_t$ are generic non-terminals; $M_s^{(k)}$ and $M_t^{(l)}$ represent pre-terminals. We use the superscript here to differentiate between non-terminals of the same label (and interpretation), but different fan-outs. The fan-out of the non-terminals is constrained by v_s respectively v_t, i. e. $1 \leq k \leq v_s$ and $1 \leq l \leq v_t$.

$$(1) \quad \langle M_s^{(k)}(\boldsymbol{\alpha}) \to \varepsilon\,, M_t^{(l)}(\boldsymbol{\beta}) \to \varepsilon \rangle$$

where $\boldsymbol{\alpha} \in (T_s^*)^k$ with $1 \leq k \leq v_s$, and $\boldsymbol{\beta} \in (T_t^*)^l$ with $1 \leq l \leq v_t$.

$$(2) \quad \langle A_s^{(k)}(\boldsymbol{\alpha}) \to M_{s\boxed{1}}^{(k)}(\boldsymbol{\alpha})\,, A_t^{(l)}(\boldsymbol{\beta}) \to M_{t\boxed{1}}^{(l)}(\boldsymbol{\beta}) \rangle$$

where $\boldsymbol{\alpha} \in V_s^k$ with $1 \leq k \leq v_s$ and $\boldsymbol{\beta} \in V_t^l$ with $1 \leq l \leq v_t$.

$$(3) \quad \begin{aligned} &\langle A_s^{(k)}(\boldsymbol{\alpha}) \to A_{s\boxed{1}}^{\prime(k')}(\boldsymbol{\alpha'})\; A_{s\boxed{2}}^{\prime\prime(k'')}(\boldsymbol{\alpha''})\,, \\ &\quad A_t^{(l)}(\boldsymbol{\beta}) \to A_{t\boxed{1}}^{\prime(l')}(\boldsymbol{\beta'})\; A_{t\boxed{2}}^{\prime\prime(l'')}(\boldsymbol{\beta''}) \rangle \end{aligned}$$

where $\boldsymbol{\alpha} \in (V_s^+)^k$, $\boldsymbol{\alpha'} \in V_s^{k'}$, $\boldsymbol{\alpha''} \in V_s^{k''}$ with $1 \leq k \leq v_s$, $1 \leq k' \leq v_s$, $1 \leq k'' \leq v_s$ and $\boldsymbol{\beta} \in (V_t^+)^l$, $\boldsymbol{\beta'} \in V_t^{l'}$, $\boldsymbol{\beta''} \in V_t^{l''}$ for $1 \leq l \leq v_t$, $1 \leq l' \leq v_t$, $1 \leq l'' \leq v_t$.

Rule (1) builds pre-terminal constituents from terminals. To create a general purpose constituent from a pre-terminal constituent, rule (2) would be used. Hierarchical structures are created with rule (3). □

The bottom-up hierarchical aligner builds initial constituents labeled M based on the m translation units $\langle D_s^{(j)}, D_t^{(j)} \rangle$ $(1 \leq j \leq m)$ of a given word alignment $\mathcal{A}$ for a sentence pair $\langle s, t \rangle$. Constituents are then combined with each other obeying the constraints of the simulated all-accepting grammar G. In the case at hand those are G being of the formalism SLCFRS, G having a rank of $u = 2$ and G having a specific fan-out $v_{v_s|v_t}$. We again specify the hierarchical aligner as a deduction system. It is a generalization of the deduction system provided in section 3.2.1 for SCFG. Indeed, the deduction rules are almost the same. The only difference is in the side conditions, where the number of blocks that a bit vector $\boldsymbol{\rho}$ may span is not restricted to 1 anymore, but more general to v_s for source bit vectors and v_t for target bit vectors. This is due to the relationship of SCFG and SLCFRS with SCFG being a specific case of SLCFRS with fan-out $v = 2_{1|1}$. The interpretation is that the span of a non-terminal can cover several blocks, i. e. the non-terminal has a discontinuous yield.

For our convenience, the modified deduction rules are shown in the following. The definitions of the UNARY operation and the GOAL item stay the same. The source and target fan-out of the non-terminals M and A is implicitly given by the number of blocks of $\boldsymbol{\rho}_s$ and $\boldsymbol{\rho}_t$ respectively.

SCAN:

$$\frac{}{[M, \boldsymbol{\rho}_s, \boldsymbol{\rho}_t]} \quad \text{a translation unit } \langle D_s^{(j)}, D_t^{(j)} \rangle$$

where $\boldsymbol{\rho}_s(i) = 1$ if $i \in D_s^{(j)}$, otherwise $\boldsymbol{\rho}_s(i) = 0$, and $\boldsymbol{\rho}_t(i) = 1$ if $i \in D_t^{(j)}$, otherwise $\boldsymbol{\rho}_t(i) = 0$.

UNARY:

$$\frac{[M, \boldsymbol{\rho}_s, \boldsymbol{\rho}_t]}{[A, \boldsymbol{\rho}_s, \boldsymbol{\rho}_t]} \quad b(\boldsymbol{\rho}_s) \leq v_s, b(\boldsymbol{\rho}_t) \leq v_t$$

BINARY:

$$\frac{[A, \boldsymbol{\rho}_s^1, \boldsymbol{\rho}_t^1], [A, \boldsymbol{\rho}_s^2, \boldsymbol{\rho}_t^2]}{[A, \boldsymbol{\rho}_s^3, \boldsymbol{\rho}_t^3]} \quad \begin{array}{c} \boldsymbol{\rho}_s^1 \cap \boldsymbol{\rho}_s^2 = 0^n, \boldsymbol{\rho}_t^1 \cap \boldsymbol{\rho}_t^2 = 0^{n'}, \\ \boldsymbol{\rho}_s^3 = \boldsymbol{\rho}_s^1 \cup \boldsymbol{\rho}_s^2, \boldsymbol{\rho}_t^3 = \boldsymbol{\rho}_t^1 \cup \boldsymbol{\rho}_t^2, \\ b(\boldsymbol{\rho}_s^3) \leq v_s, b(\boldsymbol{\rho}_t^3) \leq v_t \end{array}$$

UNARYMIXED:

$$\frac{[M, \boldsymbol{\rho}_s^M, \boldsymbol{\rho}_t^M], [A, \boldsymbol{\rho}_s^A, \boldsymbol{\rho}_t^A]}{[A, \boldsymbol{\rho}_s, \boldsymbol{\rho}_t]} \quad \begin{array}{c} \boldsymbol{\rho}_s^M \cap \boldsymbol{\rho}_s^A = 0^n, \boldsymbol{\rho}_t^M \cap \boldsymbol{\rho}_t^A = 0^{n'}, \\ \boldsymbol{\rho}_s = \boldsymbol{\rho}_s^M \cup \boldsymbol{\rho}_s^A, \boldsymbol{\rho}_t = \boldsymbol{\rho}_t^M \cup \boldsymbol{\rho}_t^A, \\ b(\boldsymbol{\rho}_s) \leq v_s, b(\boldsymbol{\rho}_t) \leq v_t \end{array}$$

BINARYMIXED:

$$\frac{[M, \boldsymbol{\rho}_s^M, \boldsymbol{\rho}_t^M], [A, \boldsymbol{\rho}_s^1, \boldsymbol{\rho}_t^1], [A, \boldsymbol{\rho}_s^2, \boldsymbol{\rho}_t^2]}{[A, \boldsymbol{\rho}_s^3, \boldsymbol{\rho}_t^3]} \quad \begin{array}{c} \boldsymbol{\rho}_s^M \cap \boldsymbol{\rho}_s^1 = 0^n, \boldsymbol{\rho}_s^1 \cap \boldsymbol{\rho}_s^2 = 0^n, \\ \boldsymbol{\rho}_s^2 \cap \boldsymbol{\rho}_s^M = 0^n, \boldsymbol{\rho}_t^M \cap \boldsymbol{\rho}_t^1 = 0^{n'}, \\ \boldsymbol{\rho}_t^1 \cap \boldsymbol{\rho}_t^2 = 0^{n'}, \boldsymbol{\rho}_t^2 \cap \boldsymbol{\rho}_t^M = 0^{n'}, \\ \boldsymbol{\rho}_s^3 = \boldsymbol{\rho}_s^M \cup \boldsymbol{\rho}_s^1 \cup \boldsymbol{\rho}_s^2, \\ \boldsymbol{\rho}_t^3 = \boldsymbol{\rho}_t^M \cup \boldsymbol{\rho}_t^1 \cup \boldsymbol{\rho}_t^2, \\ b(\boldsymbol{\rho}_s^3) \leq v_s, b(\boldsymbol{\rho}_t^3) \leq v_t \end{array}$$

GOAL:

$$[A, \boldsymbol{\rho}_s, \boldsymbol{\rho}_t]$$

with $\boldsymbol{\rho}_s(i) = 1$ for all $0 \leq i < |\boldsymbol{s}|$ and $\boldsymbol{\rho}_t(i) = 1$ for all $0 \leq i < |\boldsymbol{t}|$.

The SCAN rule creates pre-terminal items from the translation units of the input sentence pair. The UNARY operation builds A items from pre-terminal M items. Two A items are combined to a larger A item with the BINARY rule. In contrast, the UNARYMIXED rule combines an M item with an A item, simulating a mixed SLCFRS rule of rank 1, and the BINARYMIXED rule simulates a mixed SLCFRS rule of rank 2. All operations, except SCAN, are subject to the constraint that the number of blocks in the source respective target bit vector may not exceed v_s respective v_t.

The alignment structure $\mathcal{A}$ can be deduced with a $(2, v_{v_s|v_t})$-SLCFRS if a GOAL item is found by using any sequence of the five rules. In addition, $\mathcal{A}$ can be deduced with a $v_{v_s|v_t}$-NF-SLCFRS if a GOAL item is found by using only the SCAN, UNARY and BINARY rules.

As in the previous experiments, a chart and an agenda are populated in order to compute the items.

4.3.2 *Experiments*

Experiments are conducted in the same manner as in section 3.2.1, including the same data sets which are described there. Now we additionally vary the fan-out $v_{v_s|v_t}$ of the all-accepting grammar that is

		fan-out v	
		$2_{1\|1}$	$4_{2\|2}$
Martin et al.	*en-ro (30)*	45.07	97.85
	en-hi (40)	82.73	100.00
	en-iu (40)	40.66	95.60
Padó & Lapata	*en-de (15)*	73.74	100.00
Mihalcea & Pedersen	*en-fr*	67.56	98.88
Graça et al.	*en-fr*	73.00	100.00
	en-pt	76.00	100.00
	en-es	82.00	100.00
	pt-fr	73.00	97.00
	pt-es	90.00	99.00
	es-fr	74.00	100.00
CDT	*da-en (25)*	72.90	98.93
	da-de (25)	64.87	98.42
	da-es (25)	66.61	97.68
	da-it (25)	69.01	97.65
Holmqvist & Ahrenb.	*en-sv (30)*	82.83	99.78
Schoenemann	*en-de (40)*	29.15	94.74
Lambert et al.	*en-es (40)*	47.15	97.83
Macken	*en-nl (30)*	57.14	98.86

Table 7: Alignment reachability scores for $(2, v)$-NF-SLCFRS. The scores for 2-NF-SCFG from table 4 are repeated here under the column with the title $v = 2_{1|1}$.

		fan-out v			
		$2_{1\|1}$	$4_{2\|2}$	$3_{1\|2}$	$3_{2\|1}$
Martin et al.	*en-ro (30)*	95.07	**100.00**	99.38	97.84
	en-hi (40)	96.36	**100.00**	100.00	100.00
	en-iu (40)	100.00	**100.00**	100.00	100.00
Padó & Lapata	*en-de (15)*	94.41	**100.00**	100.00	98.88
Mihalcea & Pedersen	*en-fr*	95.30	**100.00**	99.78	99.33
Graça et al.	*en-fr*	95.00	**100.00**	100.00	99.00
	en-pt	98.00	**100.00**	100.00	100.00
	en-es	96.00	**100.00**	100.00	100.00
	pt-fr	92.00	**100.00**	100.00	95.00
	pt-es	99.00	**100.00**	100.00	100.00
	es-fr	91.00	**100.00**	100.00	94.00
CDT	*da-en (25)*	97.80	**100.00**	99.90	99.84
	da-de (25)	94.94	**100.00**	100.00	100.00
	da-es (25)	97.50	**100.00**	98.93	99.64
	da-it (25)	97.95	**100.00**	99.90	99.71
Holmqvist & Ahrenb.	*en-sv (30)*	95.60	**100.00**	99.79	99.68
Schoenemann	*en-de (40)*	76.11	**100.00**	96.76	90.69
Lambert et al.	*en-es (40)*	94.85	**100.00**	98.65	98.37
Macken	*en-nl (30)*	94.86	**100.00**	99.62	98.48

Table 8: Alignment reachability scores for $(2, v)$-SLCFRS. The scores for 2-SCFG from table 4 are repeated here under the column with the title $v = 2_{1|1}$.

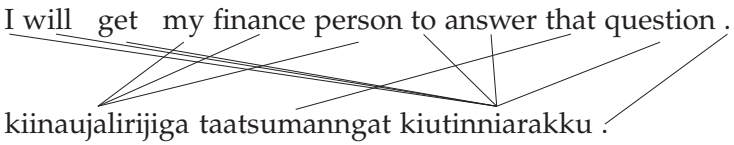


Figure 50: English-Inuktitut alignment example. The translation unit ⟨I will get , to answer , question ; kiutinniarakku⟩ has two gaps on the source side and no gap on the target side. Accordingly, an NF-SLCFRS rule of fan-out $4_{3|1}$ is required to derive it.

being simulated. If the bottom-up hierarchical aligner reaches a goal item for a given alignment structure, the alignment can be generated with the formalism in question. Alignment reachability scores are reported for the individual data sets.

Table 7 shows the results for NF-SLCFRS and table 8 for SLCFRS of different fan-outs. The alignment reachability scores for NF-SCFG and SCFG from section 3.2.1 are repeated for convenience.

The numbers in the tables demonstrate that alignment reachability increases considerably when allowing for discontinuous constituents. With a fan-out of $4_{2|2}$, even NF-SLCFRS induces all alignments present in six of the data sets, and reaches scores >97 for the other data sets, except two of them for which scores are still >94.7. The sentence pairs that cannot be induced with a $(2, 4_{2|2})$-NF-SLCFRS all display translation units that require three (or very rarely four) blocks on at least the source or the target side. An interesting observation is that only the English-Inuktitut data can nevertheless be generated with fan-out 4, by distributing the allowed discontinuity unequally: with a NF-SLCFRS of fan-out $4_{3|1}$, the alignment reachability is 100. This is not surprising given the fact that Inuktitut is a polysynthetic language, thus several English words may correspond to one Inuktitut token. See figure 50 for an example from the data set.

For grammars without the normal-form constraint, alignment reachability is generally higher than with the normal-form constraint (cf. columns with equal fan-out in table 7 and 8). This is in line with our expectations, as NF-SLCFRS and SLCFRS do not have the same alignment capacity (section 4.2.2). The grammars which are used in practice for the translation modeling in hierarchical phrase-based translation typically are not required to be in normal form. Quite the contrary, they are usually highly lexicalized (cf. section 2.2.5).

Our experiments reveal that with a $(2, 4_{2|2})$-SLCFRS, all occurring alignment configurations are induced. The last two columns of table 8 show that for some data sets, a fan-out of 3 is enough to induce all alignments. This is encouraging since such grammars are a minimal extension to SCFG towards mild context-sensitivity.

The presented experiments show that by moving from synchronous grammars with only continuous constituents to grammars which allow two blocks per constituent, all manual alignments in a variety of data sets can be generated. Given the parsing complexity that comes with allowing discontinuities, this is a promising finding. It has already been shown for monolingual LCFRS parsing that restricting the fan-out to 2 drastically reduces parsing times (Maier et al., 2012).

4.4 RELATED WORK

4.4.1 *Empirical Investigations of Alignments*

The work carried out in Wellington et al. (2006) by the group around I. D. Melamed is most similar to our investigation concerning the empirical alignment capacity of synchronous grammars of different fan-outs. They investigate how many aligned sentence pairs require gaps in the constituents in order to generate a hierarchical alignment. They find that 5% of their Chinese-English sentence pairs cannot be covered without gaps. The parse failure rates for the other language pairs lie between 0% and 2%. With allowing only one gap on either the source or the target side, all alignments can be derived. Their reported failure rates are rather low compared to ours which can be attributed to their experimental methodology, which differs considerably from ours. This is spelled out in section 3.2.3.

Other similar investigations, some of them also using a hierarchical aligner, are concerned with the alignment capacity of variants of synchronous context-free formalisms, but not beyond (Zens and Ney, 2003; Søgaard and Wu, 2009; Søgaard and Kuhn, 2009; Søgaard, 2010). They are described in more detail in section 3.2.3.

4.4.2 *Translation Modeling*

This section presents related work which is also concerned with translation modeling beyond SCFG. The proposal to employ SLCFRS for that purpose was certainly not the first step towards mild context-sensitivity for translation modeling. We can determine three main strands of motivation for proposing more powerful translation models in the literature:

INDUCING COMPLEX ALIGNMENT CONFIGURATIONS Translation models based on SCFG or phrases are not expressive enough to induce all alignment configuration which occur in aligned data. For an adequate modeling of translational equivalence thus more powerful models are required. More details can be found in chapter 3 since this is our motivation for proposing SLCFRS. The same motivation has lead to two other proposals: using Range Concatenation Grammar (RCG) and its ability to copy substrings (Søgaard, 2008b) and a phrase-based approach which allows for discontinuous phrases (Galley and Manning, 2010).

ADEQUATE MODELING OF MONOLINGUAL SYNTAX Following up on the finding that certain phenomena in natural language syntax cannot be modeled with CFG (see section 2.1.3), the early work of translation modeling beyond SCFG is motivated by an adequate representation of the source and the target languages from a linguistic point of view. This inevitably leads to the use of mildly context-sensitive grammar formalisms in their synchronous version, e. g. Synchronous Tree-Adjoining Grammar (STAG). Some formalisms have been even designed solely in the context of translation modeling, e. g. Generalized Multitext Grammar (GMTG). Many of these proposals have appeared before the breakthrough of SMT, so no large-scale systems and experimental results are available.

NON-CONSTITUENT PHRASES IN SYNTAX-BASED MODELS Even though syntax-based SMT models which make use of parse trees, e. g. string-to-tree or tree-to-tree models, have been argued to have advantages over phrase-based models, e. g. more advanced modeling of reordering and more well-formed translations, this advance was often not reflected in experimental results and BLEU scores. One often

cited reason is that in syntax-based models using SCFG, or related formalisms like Synchronous Tree-Substitution Grammar (STSG), only constituent phrases, i. e. words in the yield of a node in the parse tree, can be modeled. One possible solutions is the usage of a more powerful grammar formalism like Multi Bottom-up Tree Transducer (MBOT) or Synchronous Tree Sequence Substitution Grammar (STSSG).

In the remainder of this section, an overview over the approaches and formalisms which fall into these three categories will be provided. We will also look into their relation to (S)LCFRS and whether they induce the complex alignment configurations.

Range Concatenation Grammar

A. Søgaard (Søgaard, 2008b, 2011) proposes to apply a grammar formalism for translation modeling which is even more expressive than LCFRS, namely Range Concatenation Grammar (RCG), and to exploit its ability to copy substrings during the derivation (see p. 57) in order to induce complex alignment configurations. This approach allows for the induction of all possible alignment configurations. However, it has certain downsides, which are already mentioned in Søgaard and Kuhn (2009); for example the use of intersection to induce complex alignment configurations possibly misses generalizations, and no tight probability estimation is possible for such grammars.

Throughout this work, RCG syntax has been used for the notation of LCFRS, so the reader should be familiar with it. The difference in the definition of RCG and LCFRS or equivalently Simple Range Concatenation Grammar (SRCG) is that the arguments α_i, $1 \leq i \leq dim(A)$, of RHS non-terminals $A(\alpha_1, \ldots, \alpha_{dim(A)})$ can be sequences of terminals and variables (as opposed to just one variable), i. e. $\alpha_i \in (T \cup V)^*$, and that the occurrence of variables in the rewriting rules is less constrained for RCG than for LCFRS (cf. def. 2.29, p. 26 for LCFRS). In a *bottom-up non-erasing* RCG $G = (N, T, V, P, S)$, which is the one proposed for machine translation, all variables which occur in the RHS of a rule r must also occur in the LHS of r, for all $r \in P$. In the translation grammar proposed by Søgaard, a start predicate of fan-out 2 is used, the two arguments representing the source and the target

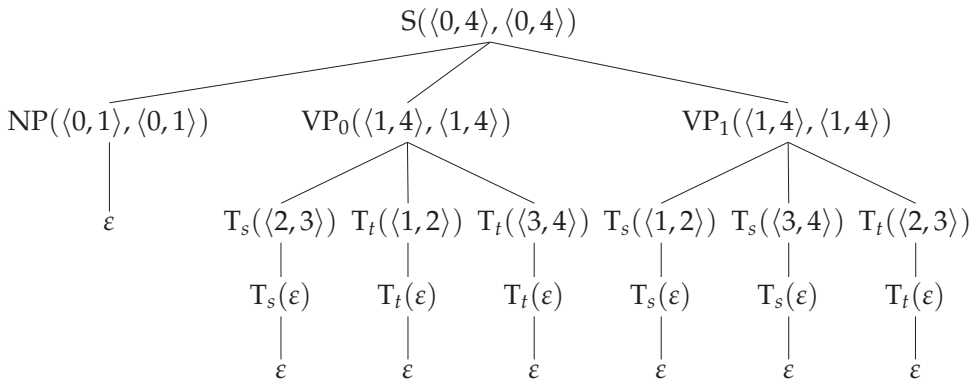


Figure 51: RCG derivation for ex. 4.24

sequence. Furthermore, the maximal fan-out of the grammar is 2 and an argument in the LHS of a rule contains at most two variables.

Example 4.24 (RCG for translation). This RCG fragment, inspired by an example in Søgaard (2011), shows rules which can be used to derive the bonbon configuration in figure 34. Rule (1) is non-linear, i. e. beyond LCFRS. Using the copying mechanism, each translation unit which is part of a complex alignment configuration is derived independently of and in parallel to the other translation units (VP_0 and VP_1).

$$\begin{aligned}
&S(Y_1Y_2, Z_1Z_2) && \rightarrow NP(Y_1, Z_1)VP_0(Y_2, Z_2)VP_1(Y_2, Z_2) && (1)\\
&VP_0(\text{ne } Y_1 \text{ pas}, Z_1 \text{ not } Z_2) && \rightarrow T_s(Y_1)T_t(Z_1)T_t(Z_2) && (2)\\
&VP_1(Y_1 \text{ mange } Y_2, \text{does } Z_1 \text{ eat}) && \rightarrow T_s(Y_1)T_s(X_2)T_t(Z_1) && (3)\\
&NP(\text{Pierre , Pierre}) && \rightarrow \varepsilon && (4)\\
&T_s(aY_1) && \rightarrow T_s(Y_1) \quad \text{for all } a \in T_s && (5)\\
&T_t(aZ_1) && \rightarrow T_t(Z_1) \quad \text{for all } a \in T_t && (6)\\
&T_s(\varepsilon) && \rightarrow \varepsilon && (7)\\
&T_t(\varepsilon) && \rightarrow \varepsilon && (8)
\end{aligned}$$

Figure 51 shows the derivation of

⟨Pierre ne mange pas , Pierre does not eat⟩

□

Discontinuous Phrase-based Translation

In Galley and Manning (2010), a phrase-based translation model and decoder are proposed with the objective to be able to generate the complex alignment configurations. They extend the phrase-based approach (cf. section 2.2.4) in that *discontinuous phrase pairs* are also allowed. This approach is not syntax-based and therefore does not rely on a specific grammar formalism. It outperforms a phrase-based system and a hierarchical phrase-based system. More details are provided in chapter 5.4.1.

In a way our proposal to use SLCFRS as the formalism for translation modeling is the tree-based counterpart to their approach. Methods to integrate linguistic constituency information into the so far only formal tree-based approach can be directly transferred from the SCFG-based approaches to SLCFRS. In contrast, it is not obvious how to include such information into the phrase-based systems.

Generalized Multitext Grammar

The group around I. D. Melamed has been an early advocate of translation modeling beyond Context-Free Grammar. In Wellington et al. (2006), they note the necessity for discontinuous constituents for deriving IO alignments with synchronous grammars of rank 2. Melamed proposes to use GMTGs (Melamed et al., 2004; Melamed, 2004) as the formalism for translation modeling. GMTG is a synchronous formalism that features discontinuous constituents. It is weakly equivalent to LCFRS. Just like SCFG and SLCFRS, it implements synchronous rewriting by an indexation of the RHS non-terminals. A GMTG has a specific *dimension D* which corresponds to the number of languages which are synchronously described. While SCFG and SLCFRS define correspondences between pairs of languages, D can be larger than 2. Discontinuous constituents in a dimension are expressed by tuples of non-terminals with more than one element on the LHS and by using the same index several times in the RHS. Ex. 4.25 shows GMTG rules and their SLCFRS counterparts.

EXAMPLE 4.25 (Generalized Multitext Grammar). The following GMTG grammar fragment of two dimensions, taken from Melamed

et al. (2004), generates ⟨*The doctor treats his teeth*, *El médico le examino los dientes*⟩.

$$
\begin{aligned}
&[(\mathrm{S}),(\mathrm{S})] && \rightarrow [(\mathrm{NP}^1\mathrm{VP}^2),(\mathrm{NP}^1\mathrm{VP}^2)]\\
&[(\mathrm{VP}),(\mathrm{VP})] && \rightarrow [(\mathrm{V}^1\mathrm{NP}^2),(\mathrm{NP}^2\mathrm{V}^1\mathrm{NP}^2)]\\
&[(\mathrm{NP}),(\mathrm{NP})] && \rightarrow [(\text{The doctor}),(\text{El médico})]\\
&[(\mathrm{NP}),(\mathrm{NP},\mathrm{NP})] && \rightarrow [(\text{his teeth}),(\text{le},\text{los dientes})]\\
&[(\mathrm{V}),(\mathrm{V})] && \rightarrow [(\text{treats}),(\text{examino})]
\end{aligned}
$$

The equivalent SLCFRS rules are:

$$
\begin{aligned}
&\langle \mathrm{S}(XY) \rightarrow \mathrm{NP}_{\boxed{1}}(X)\mathrm{VP}_{\boxed{2}}(Y)\ , \mathrm{S}(XY) \rightarrow \mathrm{NP}_{\boxed{1}}(X)\mathrm{VP}_{\boxed{2}}(Y)\rangle\\
&\langle \mathrm{VP}(XY) \rightarrow \mathrm{V}_{\boxed{1}}(X)\mathrm{NP}_{\boxed{2}}(Y)\ , \mathrm{VP}(XYZ) \rightarrow \mathrm{V}_{\boxed{1}}(Y)\mathrm{NP}_{\boxed{2}}(X,Z)\rangle\\
&\langle \mathrm{NP}(\text{The doctor}) \rightarrow \varepsilon && ,\mathrm{NP}(\text{El médico}) \rightarrow \varepsilon\rangle\\
&\langle \mathrm{NP}(\text{his teeth}) \rightarrow \varepsilon && ,\mathrm{NP}(\text{le},\text{los dientes}) \rightarrow \varepsilon\rangle\\
&\langle \mathrm{VP}(\text{treats}) \rightarrow \varepsilon && ,\mathrm{VP}(\text{examino}) \rightarrow \varepsilon\rangle
\end{aligned}
$$

□

GMTG allows for *independent rewriting*, i. e. rewriting in one dimension continues, while in another dimension no syntactic structure is generated. This mechanism is demonstrated in ex. 4.26. Neither SCFG (def. 2.60, p. 76) nor SLCFRS (def. 4.2, p. 131) allow for independent rewriting because of the strict synchronization of the indexed RHS non-terminals.[1] While independent rewriting surely is necessary to capture certain linguistic generalization in manually designed, linguistically adequate translation grammars, it might be problematic in a data-driven, statistical setting as it has the capacity to generate target words from nothing and to delete source words without much evidence because of the limited application context of such a rule. This is also one of the reasons why synchronous grammars used in SMT are typically ε-free and unaligned words are usually "attached" to their neighboring words. Furthermore note that automatically learning grammars which allow for independent rewriting is a challenge, in particular for hierarchical phrase-based systems, because of the larger rule space.

1 This is true for the definitions given in this work. The one for SCFG is widely accepted. One could however think about a relaxation of the synchronization in order to allow for independent rewriting.

EXAMPLE 4.26 (Independent rewriting). GMTG rules which demonstrate independent rewriting. In the first rule, D has no co-indexed counterpart in the target dimension, which is why the second rule, with an empty target dimension, can be applied subsequently. The two rules could be used for describing the relation between English and a language which systematically does not use determiners.

$$[(\mathrm{NP}), (\mathrm{NP})] \rightarrow [(\mathrm{D}^1\mathrm{N}^2), (\mathrm{N}^2)]$$
$$[(\mathrm{D}), ()] \rightarrow [(\mathrm{the}), ()]$$

□

While our motivation for going beyond CFG is the ability to derive certain alignment configurations beyond CFG, Melamed's incentive for defining GMTG lies in linguistically motivated syntactic translation grammars and the general observation that discontinuous constituents are necessary for monolingual modeling of syntax.

Synchronous Tree-Adjoining Grammar

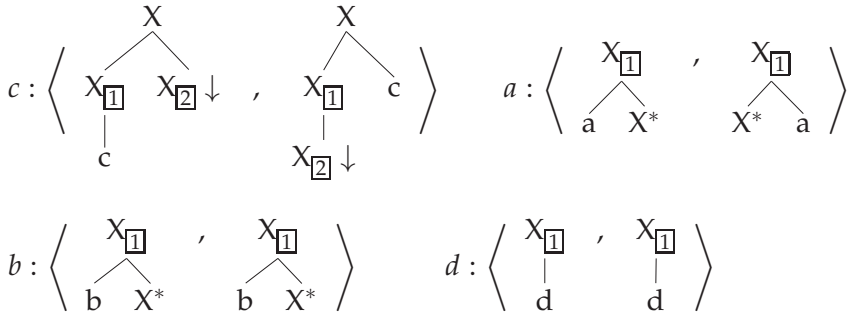
Synchronous Tree-Adjoining Grammar (STAG) (Shieber and Schabes, 1990; Shieber, 1994) generalizes Tree-Adjoining Grammar (TAG) (see p. 55) in the same way as SCFG and SLCFRS generalize CFG and LCFRS respectively. It defines the derivation of tree pairs using the TAG operations *substitution* and *adjunction* on pairs of trees. Since TAG it mildly context-sensitive and it has been found suitable for the modeling of many natural language phenomena, the idea of using STAG for translation modeling naturally has been pursued by several researchers (e. g. Abeillé et al., 1990; Harbusch and Poller, 2013; Dras, 1999). Later, this body of work has been extended to probabilistic versions of STAG and the more restricted Synchronous Tree-Insertion Grammar (STIG)[2] for statistical machine translation (e. g. Nesson et al., 2006; DeNeefe and Knight, 2009; Liu et al., 2011). Shieber (2007) also argues for using probabilistic STAG and its formal relatives as the formalism in statistical machine translation systems because of their appealing properties with respect to expressivity, trainability and computational tractability.

2 Tree-Insertion Grammar (TIG) does not allow *wrapping adjunction*, i. e. auxiliary trees where the foot node is not the left-most or right-most frontier node, and is attractive because of the resulting parsing complexity of $\mathcal{O}(n^3)$.

Approaches to translation modeling and SMT which rely on tree transducers (see p. 81) and whose grammar formalism equivalent is beyond SCFG have also appeared in the literature. In particular, tree transducer models for STAG have been proposed (e. g. Shieber, 2006; Maletti, 2010a).

To the best of our knowledge, none of the above-mentioned approaches have chosen STAG for translation modeling explicitly because of its alignment capacity. Usually an appropriate modeling of the monolingual syntax is the main motivation for choosing STAG. However, Søgaard and Kuhn (2009) point out that 2-STAGs are expressive enough to induce CDTUs and IO alignments, but not multi-gap discontinuous translation units (DTUs) with more than two gaps on one side. 2-STAGs also have the alignment capacity to induce the bon-bon configuration, as we will show. A 2-STAG is an STAG in which each elementary tree pair has at most two linked non-terminal nodes. This is the kind of STAG often used in SMT for efficiency reasons. Ex. 4.27 and ex. 4.28 demonstrates how STAG induces complex alignment configurations.

EXAMPLE 4.27 (STAG for IO alignment). Consider the following STAG elementary tree pairs.[3] This grammar fragment can be used to derive the IO alignment in figure 30(i).



A derivation would start with the tree pair labeled *c*. Tree pair *d* substitutes into the linked non-terminal $X_{\boxed{2}}$ of *c*. Tree pair *b* adjoins into the linked non-terminal $X_{\boxed{1}}$ of *c*. The latter leads to a new linked node into which tree pair *a* is adjoined. Figure 52 shows the resulting derived tree pair. To visualize the IO alignment, the target tree has been copied and mirrored below the source tree. The non-terminals' subscripts and superscripts in the derived tree denote the origin of the

3 This example has been taken from Søgaard and Kuhn (2009) and adapted since their example does not actually work.

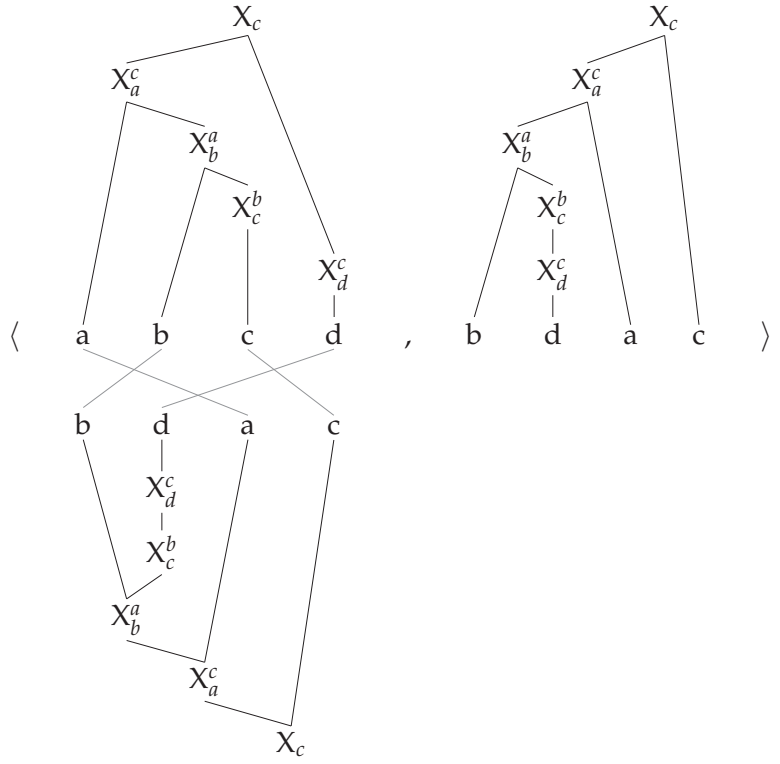
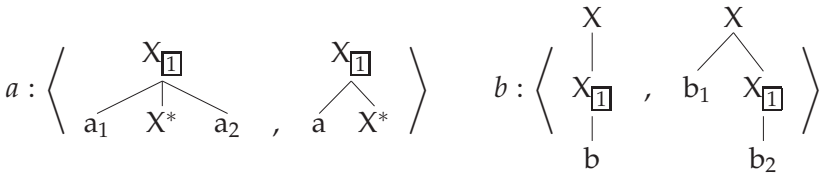


Figure 52: STAG derived tree inducing an IO alignment

non-terminal with respect to an elementary tree pair.[4] Note especially how the material of the elementary tree pair c is separated by the two adjunctions. □

Example 4.28 (STAG for bonbon alignment). As a second illustration for how STAG induces complex alignment configurations, consider the following STAG elementary tree pairs.



4 TAG non-terminal nodes are sometimes viewed as consisting of two halves, where a root node is only a lower half, while foot nodes and substitution nodes are upper halves. During adjunction, an inner node is split into two halves, which are then completed again by the half of the root node and the half of the foot node of the auxiliary tree.

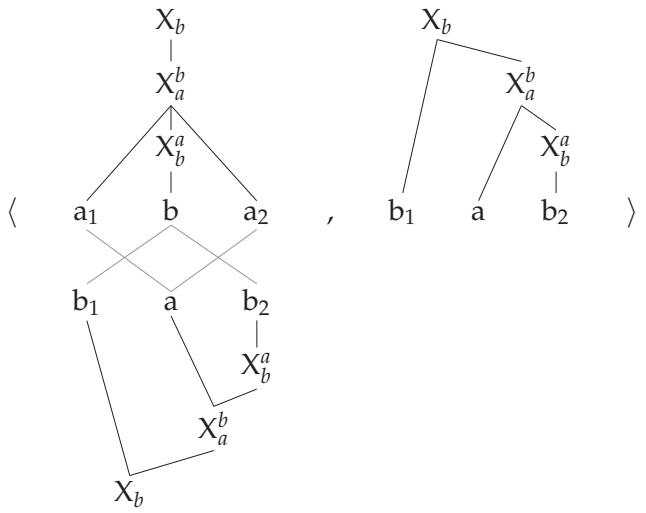


Figure 53: STAG derived tree inducing a bonbon alignment

The derivation starts with the elementary tree pair b. The auxiliary tree pair a adjoins to the linked non-terminal $X_{\boxed{1}}$ creating the derived tree in figure 53. Note how all and only the terminals which form one translation unit (given the alignment) are generated with the application of one synchronous operation. □

The languages that can be generated with TAG are a subset of the languages that can be generated with 2-LCFRS.[5] The same relation holds for the translations of STAG and SLCFRS. As STAG is also capable of modeling the complex alignment configurations under the assumptions given in section 3.1.1, it is also a potential candidate for translation modeling beyond CFG, which is the topic of this chapter. However, going from SCFG to SLCFRS is somewhat more natural and direct than going to STAG. This is because the basic units (the rules) and the applied operations are similar for SCFG and SLCFRS. Corresponding notions and concepts, methods and algorithms, e. g. for parsing, therefore carry over in a rather straightforward manner. This is different for STAG where the basic units are trees, the operations between the trees are particular, and derivation tree and derived tree do not coincide. Note also the additional structure which has to be stipulated in the two examples in order to induce the two complex align-

5 More concretely, TAG is weakly equivalent to well-nested 2-LCFRS.

ment configurations. This makes automatic grammar learning more complex.

Range Concatenation Transducer

Besides Søgaard (2008b), RCG as a formalism for machine translation had also been proposed earlier by Bertsch and Nederhof (2001) in the form of a *Range Concatenation Transducer (RCT)*, a combination of two RCGs. Non-terminals take two sequences of arguments, input arguments and output arguments, and they have the following form:

$$A(\alpha_1, \ldots, \alpha_p)(\beta_1, \ldots, \beta_{p'})$$

where $A \in N$ with the input fan-out $p \geq 1$ and the output fan-out $p' \geq 1$, and $\alpha_i \in (V \cup T)^*$ for $1 \leq i \leq p$ and $\beta_i \in (V \cup T)^*$ for $1 \leq i \leq p'$.

Bertsch and Nederhof (2001) show that, in the general case, the recognition problem for a pair of strings and a RCT is undecidable, but that it is tractable for a restricted subclass, namely *Simple Range Concatenation Transducer (SRCT)*. An SRCT is the combination of two SRCGs to a transducer as seen in the previous paragraph. Since SRCG and LCFRS are equivalent, and both SRCT and SLCFRS define a synchronization between non-terminals, it is immediate that SRCT and SLCFRS are equivalent in terms of their strong generative capacity. Any SRCT can be converted to an equivalent SLCFRS and vice versa in a straight forward way. Roughly, when going from SRCT to SLCFRS, each transduction rule needs to be split into a source LCFRS rule, using the source arguments, and a target LCFRS rule, using the output arguments. Non-terminal labels are identical in the source and the target projection, and non-terminals which originate from one SRCT non-terminal are co-indexed. Consider ex. 4.29 as an example. For the other direction, the two LCFRS rules in each SLCFRS rule need to be merged to one transduction rule. If non-terminal labels of synchronized non-terminals are not identical, we create a compound non-terminal alphabet. Ex. 4.30 shows an example.

EXAMPLE 4.29 (Simple Range Concatenation Transducer I). The following SRCT grammar fragment generates ⟨*The doctor treats his teeth*, *El*

médico le examino los dientes⟩. The equivalent SLCFRS rules are shown in ex. 4.25, p. 155.

$$\begin{aligned}
&\mathrm{S}(XY)(XY) \rightarrow \mathrm{NP}(X)(X)\,\mathrm{VP}(Y)(Y)\\
&\mathrm{VP}(XY)(XYZ) \rightarrow \mathrm{V}(X)(Y)\,\mathrm{NP}(Y)(X,Z)\\
&\mathrm{NP}(\text{The doctor})(\text{El médico}) \rightarrow \varepsilon\\
&\mathrm{NP}(\text{his teeth})(\text{le},\text{los dientes}) \rightarrow \varepsilon\\
&\mathrm{VP}(\text{treats})(\text{examino}) \rightarrow \varepsilon
\end{aligned}$$

□

Example 4.30 (Simple Range Concatenation Transducer II). The following SRCT rules generate $\{\langle \mathrm{a}^n\mathrm{b}^m\mathrm{c}^n\mathrm{d}^m, \mathrm{a}^n\mathrm{b}^m\mathrm{d}^m\mathrm{c}^n\rangle \mid n, m > 0\}$. The non-terminal label alphabet of this grammar fragment is $\{\mathrm{S/S}, \mathrm{A/C}, \mathrm{B/D}\}$. The equivalent SLCFRS rules are shown in ex. 4.7, p. 133.

$$\begin{aligned}
&\mathrm{S/S}(Y_1Y_2Y_3Y_4)(Z_1Z_2Z_3) \rightarrow \mathrm{A/C}(Y_1,Y_3)(Z_1,Z_3)\,\mathrm{B/D}(Y_2,Y_4)(Z_2)\\
&\mathrm{A/C}(\mathrm{a}Y_1,\mathrm{c}Y_2)(\mathrm{a}Z_1,Z_2\mathrm{c}) \rightarrow \mathrm{A/C}(Y_1,Y_2)(Z_1,Z_2)\\
&\mathrm{B/D}(\mathrm{b}Y_1,\mathrm{d}Y_2)(\mathrm{b}Z_1\mathrm{d}) \rightarrow \mathrm{B/D}(Y_1,Y_2)(Z_1)\\
&\mathrm{A/C}(\mathrm{a},\mathrm{c})(\mathrm{a},\mathrm{c}) \rightarrow \varepsilon\\
&\mathrm{B/D}(\mathrm{b},\mathrm{d})(\mathrm{bd}) \rightarrow \varepsilon
\end{aligned}$$

□

The work by Bertsch and Nederhof (2001) remained theoretical. To the best of our knowledge, RCTs for translation modeling have not been picked up in the machine translation community. Note that the modeling of specific alignment configurations is not a topic in Bertsch and Nederhof (2001). They propose RCT as an interesting extension to RCG and RCG parsing, and focus rather on the appropriate monolingual syntactic modeling of language and language pairs.

Synchronous Tree Sequence Substitution Grammar

Another series of work of translation modeling beyond SCFG has been contributed by M. Zhang and colleagues, in particular for tree-to-tree machine translation. They use a grammar formalism called *Synchronous Tree Sequence Substitution Grammar (STSSG)* for translation modeling, including implementations for grammar learning and decoding. STSSG is similar to STSG, with the difference that the source

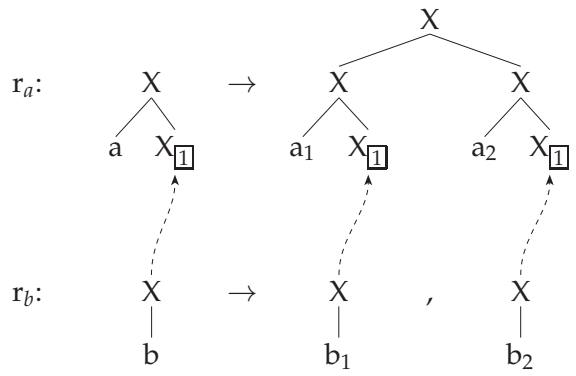


Figure 54: Non-contiguous STSSG/MBOT rules for ex. 4.31

and the target side potentially consist of several tree fragments instead of just one each. Roughly speaking, an STSSG is a finite set of rules of the form

$$(t_1, \ldots, t_n) \rightarrow (t'_1, \ldots, t'_m)$$

where $(t_1, \ldots, t_n)$ is an ordered sequence of tree fragments for the source side and $(t'_1, \ldots, t'_m)$ is an ordered sequence of tree fragments for the target side. Non-terminal leaves in $(t_1, \ldots, t_n)$ are linked with the non-terminal leaves in $(t'_1, \ldots, t'_m)$ in a many-to-many relation. Using a substitution operation just as in STSG, a source tree for an input sentence is derived, while the tree fragments on the target side at the same time derive a translation. Thereby linked non-terminals have to be replaced simultaneously with the tree fragments of a rule, and the order of the tree fragments has to be preserved in the derived tree.

Two different flavors of STSSG have been proposed. In the original formulation (Zhang et al., 2008b,c), the tree fragments of one tree sequence need to be adjacent in the final derived tree, i. e. they have to substitute into non-terminal leaf nodes which are consecutive in the frontier of the tree fragment into which they substitute. Sun et al. (2009) give up on this contiguity constraint and use a flavor of STSSG which allows for *non-contiguous tree sequences* in the rules. We will refer to it as *non-contiguous STSSG*. Consider ex. 4.31 for illustration.

Example 4.31 (STSSG for CDTU). Figure 54 shows two STSSG rules translating ab to $a_1b_1a_2b_2$. They derive the CDTU in figure 32(i). Rule r_a induces the translation unit $\langle a; a_1, a_2 \rangle$ and rule r_b generates the translation unit $\langle b; b_1, b_2 \rangle$. This example shows a non-contiguous STSSG.

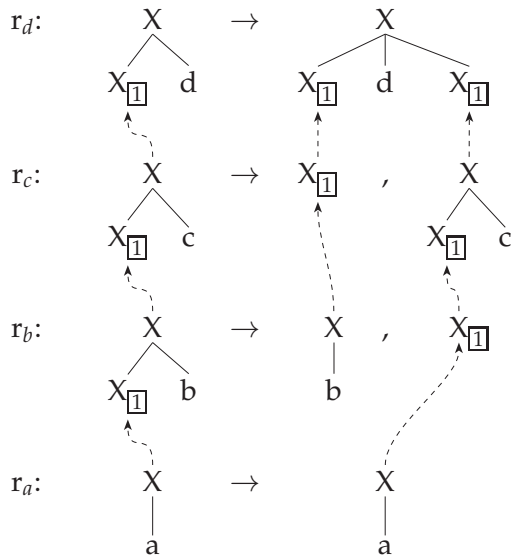


Figure 55: Non-contiguous STSSG/MBOT rules which derive an IO alignment

Under the contiguity constraint, rule r_a is not a valid STSSG rule since the two co-indexed non-terminal leaves on the target side are separated by the terminal a_2. □

While the original publications (Zhang et al., 2008b,c; Sun et al., 2009) provide some more details than what we have presented here, the formal properties of STSSG itself have not been studied. However, MBOT, which will be presented in the next section, is a special case of non-contiguous STSSG, in which only tree sequences of length 1 are allowed on the source side (Maletti, 2011). It thus follows from the results of the generative capacity of MBOTs (Gildea, 2012) that non-contiguous STSSG has the same weak generative capacity as LCFRS on the source and the target side. It is able to induce complex alignment configurations. The resulting synchronous parse tree structures are different from those produced by SLCFRS. Ex. 4.32 shows how an IO alignment is derived. In contrast to MBOT, non-contiguous STSSG can derive CDTUs which are discontinuous on the source side, i. e. the bottom-up version of the CDTU in figure 32(i). The rules for that are the same as in figure 54, only that source and target sides are swapped.

EXAMPLE 4.32 (STSSG for IO alignment). Figure 55 shows a STSSG derivation for the IO alignment in figure 30(i). This STSSG is non-

contiguous. In comparison to the SLCFRS derivation in figure 47(i), note how rule r_d introduces the discontinuity by means of linking the source non-terminal leaf X to two target non-terminal leaves and how rules r_c and r_b pass it further down. □

The motivation for defining STSSG has not been an adequate modeling of alignment, but rather to improve the translation accuracy of tree-to-tree translation. In syntax-based models, the translation rules are constrained not only by the word alignment but also by a syntactic parse tree on the source and/or the target side. SCFG or STSG rules can only describe *syntactic phrases*, i. e. words which are in the yield of a node in the parse tree together with the corresponding tree fragment. This severely restricts the space of possible translation rules. By allowing for sequences of tree fragments, which do not necessarily correspond to a syntactic constituent in the tree, i. e. non-syntactic phrases, STSSG is an attempt to bridge the gap between syntax-based and (hierarchical) phrase-based translation models. While the contiguous definition of STSSG is well-suited for translating between non-isomorphic trees in which contiguous sequences of words are aligned to each other, the non-contiguous version also allows to capture cases where the sequences which translate to each other are non-contiguous.

Multi Bottom-up Tree Transducer

Arguing that STSSG is too powerful for SMT and therefore leads to noisy translation rules, a group around A. Maletti has worked on using (local, linear, non-deleting extended) *Multi Bottom-up Tree Transducers (MBOTs)* (Arnold and Dauchet, 1982; Lilin, 1981) for SMT, primarily for string-to-tree translation (Maletti, 2010b). One can view MBOTs as STSSGs in which the length of the sequence of tree fragments on the source side is constrained to 1 (Maletti, 2011). We will use this perspective here instead of introducing the tree transducer notation. Accordingly the rules have the following form

$$t \to (t'_1, \ldots, t'_m)$$

where t is a tree fragment for the source side, and $(t'_1, \ldots, t'_m)$ is an ordered sequence of tree fragments for the target side. Each non-terminal leaf in t is linked with one or several non-terminal leaves in $(t'_1, \ldots, t'_m)$, while each non-terminal leaf in $(t'_1, \ldots, t'_m)$ is linked with exactly one non-terminal leaf in t. See the referenced literature and

Engelfriet et al. (2009) for details and a formal definition. Figures 54 and 55 are also examples for MBOTs since, in each rule, the source tree sequence has length 1, and also the linking relation is as required (1-to-n with $n \geq 1$).

Gildea (2012) shows that the class of translations that are generated by MBOTs is a subclass of the translations generated by LCFRS. In particular, on the source side, the language of an MBOT is context-free, while it has the same weak generative capacity as LCFRS on the target side. Complex alignment configurations which require discontinuous constituents on the source side consequently lie beyond the alignment capacity of MBOTs. This is the case for CDTUs which are discontinuous on the source side. In figure 54, the MBOT rules for the derivation of a CDTU which is discontinuous on the target side are shown. Note that because of the asymmetry of the MBOT, the source and the target side of an MBOT rule cannot just be swapped to translate into the inverse direction.

While the first publications of MBOT for SMT were of a rather theoretical nature, an SMT system using weighted MBOT has been implemented (Braune et al., 2013). Details and experimental results will be provided in section 5.4.2. Besides the favorable algorithmic properties of MBOT, the motivation for using it in an SMT system is similar as the one for STSSG: extending translation models based on STSG in order to build better syntax-based translation systems. A. Maletti and colleagues thereby focus on string-to-tree translation, acknowledging the need for discontinuities when constraining the target sentence with a parse tree, as noted by Wellington et al. (2006) (Seemann et al., 2015a).

Miscellaneous

Given the evidence that natural language exceeds the power of context-free formalisms and also TAG, Nederhof and Vogler (2012) introduce *Synchronous (Simple) Context-Free Tree Grammar (SCFTG)* as a formalism for machine translation. Simple Context-Free Tree Grammars of rank k are weakly equivalent to well-nested $(k+1)$-LCFRS (Kanazawa, 2009). SCFTG is thus at least as powerful as STAG, but in addition some of the generative capacity of LCFRS is added. To the best of the author's knowledge, SCFTG has merely been of theoretical interest at the intersection of machine translation and formal grammar theory.

Machine translation has also been pursued within the *Grammatical Framework*[6] (Ranta, 2004), whose formal back-end is equivalent to Parallel Multiple Context-Free Grammar (PMCFG) (Ljunglöf, 2004), a more expressive variant of Multiple Context-Free Grammar (MCFG). The machine translation approach chosen is rather rule-based than statistical, and thus follows an approach which is different from the work presented in this and the following chapter. Instead of low-level synchronous rules, rich monolingual PMCFGs are used for parsing the surface string into an abstract representation and generating a target surface string from there.

It was also noted in various contexts that discontinuous constituents emerge when binarizing synchronous grammars of continuous yields, e. g. SCFG, with rank ≥ 4 (Melamed, 2003; Rambow and Satta, 1999). See also ex. 2.69, p. 80. Grammars of maximally rank 2 are desirable for parsing and translation for efficiency reasons. As a remedy, in order to efficiently parse SCFGs and its relatives, grammars are, for example, stipulated to be of rank 2 by means of the extraction algorithm itself (e. g. Chiang, 2007), or synchronously binarized as far as possible, ignoring the non-binarizable rules during decoding (e. g. Zhang et al., 2006). It was also worked out that, since translation grammars are highly lexicalized and the terminals serve as anchors of the rule over the input sentence, not all grammar rules of rank ≥ 2 are equally problematic in terms of parsing complexity (Hopkins and Langmead, 2010). In contrast, Melamed (2003) argues for a more powerful formalism which is able to model discontinuous constituents (GMTG, see p. 155).

4.5 CONCLUSION

This chapter has introduced and defined the concept of Synchronous Linear Context-Free Rewriting Systems. It is a generalization of SCFG to constituents which span tuples of strings, i. e. which may be discontinuous on the source and/or the target side. This is the crucial characteristic which allows the derivation of the complex alignment configurations as they have been described in the previous chapter. Sample grammar fragments for those configurations were provided in section 4.2.2.

6 `http://www.molto-project.eu/`, accessed on February 13, 2018.

SLCFRS is furthermore the extension of LCFRS or SRCG to the synchronous case. Notions have been taken from the parsing literature, and definitions have been extended and adapted accordingly.

The investigation of the empirical alignment capacity of different variants of SCFG from section 3.2 was continued in this chapter for SLCFRS. Our main focus of attention was the fan-out v of a grammar which is required to hierarchically align the same variety of manually word-aligned data sets. The reason for that is that the bitext parsing complexity and also the translation complexity are directly dependent on v. We found that with an SLCFRS of rank 2 and fan-out $4_{2|2}$ all alignments occurring in the data can be generated. This means that in order to cover the full range of alignments and translations in the data sets with a *Hiero*-like grammar, the smallest possible extension from SCFG towards mild context-sensitivity is sufficient: each pair of non-terminals spans a pair of tuples of strings where the tuple length is at most 2, while for SCFG the tuples' length is 1.

We reviewed previous approaches to translation modeling beyond phrase-based and SCFG-based models, including their motivation for doing so as well as their alignment capacity with respect to complex alignment configurations. Besides the alignment capacity of a formalism, its expressiveness for adequately modeling the syntax of the individual languages and the possibility to model non-syntactic phrases in syntax-based translation systems are motives for using grammar formalisms beyond SCFG.

While GMTG and RCT are the most similar formalisms to SLCFRS from a formal point of view, the motivations for proposing and designing them are very different from our approach. The results by Galley and Manning (2010) are motivating for our direction of research. By realizing a discontinuous phrase-based system, they are likewise able to induce the complex alignment configurations, without following a hierarchical, tree-based approach. In their Chinese-to-English experiments, their system outperforms a standard phrase-based and a hierarchical system in terms of BLEU score, thus showing that being able to induce the complex alignment configurations improves translation quality.

5

HIERARCHICAL MACHINE TRANSLATION WITH DISCONTINUOUS PHRASES

The innovation we put forward in this chapter is to use *weighted Synchronous Linear Context-Free Rewriting System (SLCFRS)* as the translation model formalism for statistical machine translation (SMT). Conceptually, this grammar formalism is very close to Synchronous Context-Free Grammar (SCFG), with the addition that non-terminals span tuples of strings instead of just strings on either side of the bitext. Just as an SCFG, an SLCFRS can be used for synchronous parsing of parallel sentences as well as for translating monolingual sentences. For the latter, the source projection of the synchronous grammar is used to parse the input text, thereby generating target side derivations whose yields are translations.

The foundations for this chapter have been laid in previously presented material. On the one hand, in chapter 4, the grammar formalism SLCFRS has been introduced and shown to derive the alignment configurations which occur in aligned data, including the *complex alignment configurations*. On the other hand, a description of the most commonly used definitions and techniques in the context of tree-based SMT using SCFG was provided in section 2.2.5.

We will continue by defining and implementing a statistical machine translation system which uses a translation model based on SLCFRS. As this is a tree-based (or syntax-based) approach, naturally, the interaction between individual translational correspondences is hierarchical. The focus in this chapter will be on string-to-string models, i. e. on formal, not on linguistically motivated, translation grammars. In particular, we will follow the approach of *hierarchical phrase-based (Hiero) systems* (cf. p. 83). They use the same kind of phrases as a phrase-based system, but allow for a hierarchical combination of the phrases.

Since SLCFRS is a grammar formalism which is able to model discontinuous constituents, with our approach, the notion of a phrase is broadened by also allowing *discontinuous phrases*. They are necessary for the modeling of certain alignment structures as shown

in chapter 3. In summary, we present an approach for hierarchical machine translation with discontinuous phrases.

The translation system which is described in this chapter has already been presented in less detail in the following conference publication:

> Kaeshammer, M. (2015). Hierarchical machine translation with discontinuous phrases. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 228–238, Lisbon, Portugal. Association for Computational Linguistics.

5.1 MODEL

Recall the general SMT framework which has been presented in section 2.2.1. The hierarchical machine translation model with discontinuous phrases is situated in the same context.

Given a source sentence f and an SLCFRS $G = (N_s, N_t, T_s, T_t, V_s, V_t, P, S_s, S_t)$, the best translation $\hat{e}$ is the target yield of the highest scoring SLCFRS derivation d whose source yield is f. More precisely, repeated from equation (2.6):

$$\hat{e} \approx \mathfrak{e}\left(\underset{d \text{ s.t. } \mathfrak{f}(d)=f}{\arg\max}\ P(e, d|f)\right) \tag{5.1}$$

where e is the target yield $\mathfrak{e}(d)$ of the derivation d in scope. For the notion of *derivation* as defined for SLCFRS, see def. 4.10, p. 135.

5.1.1 *Model Definition*

Generally, given f and an SLCFRS G, many derivations will have f as the source side yield, leading to many (potentially different) target side yields, meaning possible translations e; i. e. $\mathcal{T}(G, f)$ will be large. As it is standard in SMT, we adopt a log-linear model over derivations d to weight those translation options:

$$P(e, d|f) \propto \prod_i \phi_i(d)^{\lambda_i} \tag{5.2}$$

ϕ_i are features defined on the derivations of the SLCFRS G, λ_i are feature weights to be set during tuning. The model definition then follows the one that is usually used for SCFG (see p. 82ff.).

Most of the features that we define are local to the synchronous rules which are used during the derivation d. We define them as products of feature values on those rules:

$$\phi_i(d) = \prod_{r \in d} \phi_i(r) \tag{5.3}$$

The most notable exception is, as for SCFG-based models, an n-gram language model providing a feature $\phi_{LM}(d) = P_{LM}(e)$ for the probability of seeing the target sentence e as derived by d. $P(e, d|f)$ then becomes:

$$P(e, d|f) \propto P_{LM}(e)^{\lambda_{LM}} \prod_{i \neq LM} \prod_{r \in d} \phi_i(r)^{\lambda_i} \tag{5.4}$$

The features $i \neq LM$ can conveniently be formalized as features in a *weighted SLCFRS*. The weight of an SLCFRS derivation d is defined as the product of the weights of the rules which are used in d.

Definition 5.1 (Weighted Synchronous Linear Context-Free Rewriting System). A *weighted Synchronous Linear Context-Free Rewriting System* is a tuple

$$G = (N_s, N_t, T_s, T_t, V_s, V_t, P, S_s, S_t, \omega)$$

where $(N_s, N_t, T_s, T_t, V_s, V_t, P, S_s, S_t)$ is an SLCFRS and $\omega : P \to \mathbb{R}_{\geq 0}$ is a weight function which maps from rules to real numbers. □

Definition 5.2 (Weight of a derivation (Weighted SLCFRS)). Let $G = (N_s, N_t, T_s, T_t, V_s, V_t, P, S_s, S_t, \omega)$ be a weighted SLCFRS and $\langle w_s, w_t \rangle \in T_s^* \times T_t^*$ a pair of strings. Let $\Gamma_s, \Gamma_t, \Delta_s, \Delta_t \in I^*$ and let Γ_s and Γ_t be synchronous.

1. Let $r \in P$. The weight of one derivation step $\langle \Gamma_s, \Gamma_t \rangle \Rightarrow^r_{G, \langle w_s, w_t \rangle} \langle \Delta_s, \Delta_t \rangle$ is defined as

$$\omega(\langle \Gamma_s, \Gamma_t \rangle \Rightarrow^r_{G, \langle w_s, w_t \rangle} \langle \Delta_s, \Delta_t \rangle) = \omega(r)$$

2. Let $r_1, \ldots, r_m \in P$, $m \in \mathbb{N}$. Let $\langle \Gamma_s, \Gamma_t \rangle \Rightarrow^{r_1}_{G, \langle w_s, w_t \rangle} \ldots \Rightarrow^{r_m}_{G, \langle w_s, w_t \rangle} \langle \Delta_s, \Delta_t \rangle$ be a derivation d. The weight of d is defined as

$$\omega(\langle \Gamma_s, \Gamma_t \rangle \Rightarrow^{r_1}_{G, \langle w_s, w_t \rangle} \ldots \Rightarrow^{r_m}_{G, \langle w_s, w_t \rangle} \langle \Delta_s, \Delta_t \rangle) = \prod_{j=1}^{m} \omega(r_j)$$

□

Using the less formal notation (see p. 135), we write for the latter

$$\omega(d) = \prod_{r \in d} \omega(r) \tag{5.5}$$

In the context of the log-linear modeling, we define as it has been done for SCFG:

$$\omega(r) = \prod_{i \neq LM} \phi_i(r)^{\lambda_i} \tag{5.6}$$

This results in the following model definition

$$P(e, d|f) \propto P_{LM}(e)^{\lambda_{LM}} \omega(d) \tag{5.7}$$

and overall search objective

$$\hat{e} \approx \mathfrak{e}\left(\underset{d \text{ s.t. } \mathfrak{f}(d)=f}{\arg\max} P_{LM}(\mathfrak{e}(d))^{\lambda_{LM}} \prod_{j=1}^{m} \prod_{i \neq LM} \phi_i(r_j)^{\lambda_i} \right) \tag{5.8}$$

with d being a derivation consisting of the rules $r_1 \dots r_m$. Note that this objective is the same as for SCFG-based translation (cf. equation (2.15)). They however differ in the notion of the derivation d and the rules r, and potentially in the features ϕ_i.

As for SCFG, this definition based on weighted SLCFRS has the advantageous characteristic that finding the translation, i. e. derivation, with the best weight $\omega(d)$ or even the k-best translations can be solved with dynamic programming algorithms. In section 5.3 we will also describe how the language model feature can be integrated into the search for the best translation.

5.1.2 *Features*

Let $r = \langle r_s, r_t \rangle$ be an SLCFRS rule. Besides the language model feature $P_{LM}(e)$, we use the following standard features $\phi_i(r)$, which have been proven successful in phrase-based and SCFG-based models:

- conditional translation probabilities in both directions $P(r_s|r_t)$ and $P(r_t|r_s)$,
- lexical weighting scores $lex(r_s|r_t)$ and $lex(r_t|r_s)$ (Koehn et al., 2003) that assess how well the terminals in the rule r translate to each other,

- a rule count penalty $\exp(1)$ with the help of which a preference for derivations using more or fewer rules can be learned, and
- a target sentence length penalty $\exp(-|w_t|)$, where $|w_t|$ is the number of terminals that occur in r_t, which allows the model to learn a preference for longer or shorter translations.

In addition, we would like to characterize the amount of expressivity beyond context-freeness of the applied rules. To that end, we devise the following new features:

- g_s, the *source gap degree* $\exp(\text{fan-out}(r_s) - 1)$, and
- g_t, the *target gap degree* $\exp(\text{fan-out}(r_t) - 1)$,

where $\text{fan-out}(r')$, the fan-out of an LCFRS rule r', is defined as the fan-out of the left-hand side (LHS) non-terminal of r'. The gap degree features allow to learn a preference for or against using the more powerful SLCFRS rules for the source and the target side, as opposed to using SLCFRS rules of fan-out $2_{1|1}$ which do not generate gaps and which correspond to SCFG rules. The term *gap degree* (def. 2.12, p. 20) has been coined by Maier and Lichte (2011) as a measure for the amount of discontinuity in syntactic structures. Note that in contrast to their definition, in which the gap degree of a syntactic structure is the maximum of any of its nodes, the value of the source and target gap degree feature of a derivation d will be the sum of all source, respective target, gaps which are present in the derivation.

We also use *glue rules*, as proposed by Chiang (2007), see p. 85. They allow for a monotone combination of synchronous continuous constituents as in a phrase-based model, and model the beginning and end of sentences. Here they are shown in the SLCFRS syntax:

$$\begin{aligned}
&\langle \mathrm{S}(Y_1 Y_2) \to \mathrm{S}_{\boxed{1}}(Y_1)\mathrm{X}_{\boxed{2}}(Y_2)\,, \mathrm{S}(Z_1 Z_2) \to \mathrm{S}_{\boxed{1}}(Z_1)\mathrm{X}_{\boxed{2}}(Z_2)\rangle \\
&\langle \mathrm{S}(\langle s\rangle) \to \varepsilon \qquad\qquad\quad\ \ , \mathrm{S}(\langle s\rangle) \to \varepsilon\rangle \\
&\langle \mathrm{S}(Y_1\langle /s\rangle) \to \mathrm{S}_{\boxed{1}}(Y_1) \qquad , \mathrm{S}(Z_1\langle /s\rangle)) \to \mathrm{S}_{\boxed{1}}(Z_1)\rangle
\end{aligned}$$

$\langle s\rangle$ and $\langle /s\rangle$ are special terminal symbols for the purpose of modeling beginning and end of sentences. S is the source start symbol S_s as well as the target start symbol $S_t$of the grammar G. A glue rule feature ϕ_{glue} is set to $\exp(1)$ for the first of the glue rules and to $\exp(0)$ for all other rules. Its weight λ_{glue} is also calibrated during tuning. The other two rules do not have any dedicated feature.

5.2 GRAMMAR LEARNING

The synchronous grammar rules are learned automatically in a similar fashion as those for SCFG-based Hiero machine translation, described in Chiang (2005) and Chiang (2007). See p. 83ff.

5.2.1 *Rule Extraction and Scoring*

The SLCFRS rules for the translation model are extracted from a corpus of parallel sentences that have already been word-aligned (cf. section 2.2.3). Following Och and Ney (2004) and Chiang (2005), we extract all rules that are consistent with the given word alignment $\mathcal{A}$ of a sentence pair $\langle f_1^J, e_1^I \rangle$ in a two-step procedure. First, *inital phrase pairs* are extracted. They correspond to terminal rules. Second, hierarchical rules are created by replacing phrase pairs that are contained within other phrase pairs with non-terminals and variables as will be detailed in the following.

The crucial difference to previous work on translation with SCFG is that initial phrases do not have to be continuous. Instead, a phrase is a set of word indices, as in Galley and Manning (2010), i. e. a phrase $\bar{s}$ of a sentence s_1^L is $\bar{s} \subset \{1, 2, \ldots, L\}$. Given $\langle f_1^J, e_1^I \rangle$ and a corresponding word alignment $\mathcal{A}$, a phrase pair $\langle \bar{f}, \bar{e} \rangle$ with $\bar{f} \subset \{1, \ldots, J\}$ and $\bar{e} \subset \{1, \ldots, I\}$ is called *consistent* with $\mathcal{A}$ if the following holds:

$$\forall (j, i) \in \mathcal{A} : j \in \bar{f} \leftrightarrow i \in \bar{e} \qquad \text{and} \qquad \exists j \in \bar{f}, i \in \bar{e} : (j, i) \in \mathcal{A}$$

I. e. words that are part of this phrase pair $\langle \bar{f}, \bar{e} \rangle$ may not be aligned to words outside this phrase pair, and $\langle \bar{f}, \bar{e} \rangle$ must contain at least one alignment link. A *hierarchical phrase pair* is one initial phrase pair $\langle \bar{f}, \bar{e} \rangle$ (called the base phrase pair) together with a set of initial phrase pairs $\{\langle \bar{f}_1, \bar{e}_1 \rangle, \langle \bar{f}_2, \bar{e}_2 \rangle, \ldots, \langle \bar{f}_m, \bar{e}_m \rangle\}$, $m \in \mathbb{N}$, (called *holes*) for which the following must hold:

$$\bar{f}_i \subset \bar{f} \text{ and } \bar{e}_i \subset \bar{e}, \quad 1 \leq i \leq m, \text{ and}$$

$$\bar{f}_i \cap \bar{f}_j = \emptyset \text{ and } \bar{e}_i \cap \bar{e}_j = \emptyset, \quad 1 \leq i, j \leq m.$$

I. e. each hole is a subset of the base phrase pair and the holes are mutually non-overlapping. Additional constraints may be defined on initial and hierarchical phrase pairs and the corresponding rules, see section 5.2.2.

For each initial phrase pair $\langle \bar{f}, \bar{e} \rangle$, a terminal SLCFRS rule of the following form is created and added to the set of extracted rules P:

$$\langle \mathrm{X}(\boldsymbol{\rho}_{\bar{f}}(f)) \to \varepsilon, \mathrm{X}(\boldsymbol{\rho}_{\bar{e}}(e)) \to \varepsilon \rangle$$

$\boldsymbol{\rho}_{\bar{f}}$ and $\boldsymbol{\rho}_{\bar{e}}$ are range vectors (def. 2.32, p. 27). They are applied to the source sentence f and the target sentence e respectively, to obtain the corresponding yields. $\boldsymbol{\rho}_{\bar{f}}$ and $\boldsymbol{\rho}_{\bar{e}}$ are constructed via the block sets $\Omega_{\bar{f}}$ and $\Omega_{\bar{e}}$ respectively (def. 2.7, p. 19) in the following manner: For $1 \leq i \leq |\Omega_x|$, the ith block o_i of Ω_x defines the ith range of $\boldsymbol{\rho}_x$ with $\boldsymbol{\rho}_x(i).l = \min(o_i) - 1$ and $\boldsymbol{\rho}_x(i).r = \max(o_i)$ for $x \in \{\bar{f}, \bar{e}\}$.

From each hierarchical phrase pair

$$(\langle \bar{f}, \bar{e} \rangle, \{\langle \bar{f}_1, \bar{e}_1 \rangle, \langle \bar{f}_2, \bar{e}_2 \rangle, \ldots, \langle \bar{f}_m, \bar{e}_m \rangle\}),$$

an SLCFRS rule of the form

$$\begin{aligned} \langle \mathrm{X}(\alpha_1, \ldots, \alpha_{|\Omega_{\bar{f}}|}) &\to \mathrm{X}_{\boxed{1}}(Y_1^{(1)}, \ldots, Y_{|\Omega_{\bar{f}_1}|}^{(1)}) \cdots \mathrm{X}_{\boxed{m}}(Y_1^{(m)}, \ldots, Y_{|\Omega_{\bar{f}_m}|}^{(m)}), \\ \mathrm{X}(\beta_1, \ldots, \beta_{|\Omega_{\bar{e}}|}) &\to \mathrm{X}_{\boxed{1}}(Z_1^{(1)}, \ldots, Z_{|\Omega_{\bar{e}_1}|}^{(1)}) \cdots \mathrm{X}_{\boxed{m}}(Z_1^{(m)}, \ldots, Z_{|\Omega_{\bar{e}_m}|}^{(m)}) \rangle \end{aligned}$$

is created as explained in the following. We assume any order for the set of holes. For $1 \leq k \leq m$, the kth hole corresponds to one synchronous right-hand side (RHS) non-terminal labeled X, synchronized via the index annotation k. Furthermore, for $1 \leq j \leq |\Omega_{\bar{f}_k}|$, the jth block in the block set $\Omega_{\bar{f}_k}$ is associated with a variable $Y_j^{(k)}$, and for $1 \leq i \leq |\Omega_{\bar{e}_k}|$, the ith block in the block set $\Omega_{\bar{e}_k}$ is associated with a variable $Z_i^{(k)}$.

Finally, we need to explain the LHS arguments $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. The source LHS arguments $\alpha_{j'}$, $1 \leq j' \leq |\Omega_{\bar{f}}|$, are assembled in the following way. We start with an empty $\alpha_{j'}$ and assume the j'th block $o_{j'}$ of $\Omega_{\bar{f}}$ to be ordered according to the natural increasing order of the integers in $o_{j'}$. For each element $j'' \in o_{j'}$, we check whether j'' is contained in the source side $\bar{f}_k$ of the kth hole, for $1 \leq k \leq m$. If yes, the variable $Y_j^{(k)}$ which is associated with the block o_j of $\Omega_{\bar{f}_k}$ in which j'' is contained is appended to $\alpha_{j'}$, unless $\alpha_{j'}$ currently ends in $Y_j^{(k)}$. If j'' is not part of any hole, then the yield $\langle j'' - 1, j'' \rangle(f)$ is appended to $\alpha_{j'}$. We proceed accordingly with the target LHS arguments $\beta_{i'}$, $1 \leq i' \leq |\Omega_{\bar{e}}|$. We start with an empty $\beta_{i'}$ and assume the i'th block $o_{i'}$ of $\Omega_{\bar{e}}$ to be ordered according to the natural increasing order of the integers in $o_{i'}$. For each

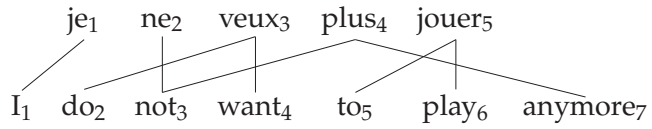


Figure 56: Word alignment for ex. 5.3

element $i'' \in o_{i'}$, we check whether i'' is contained in the target side $\bar{e}_k$ of the kth hole, for $1 \leq k \leq m$. If yes, the variable $Z_i^{(k)}$ which is associated with the block o_i of $\Omega_{\bar{e}_k}$ in which i'' is contained is appended to $\beta_{i'}$, unless $\beta_{i'}$ currently ends in $Z_i^{(k)}$. If i'' is not part of any hole, then the yield $\langle i''-1, i'' \rangle(e)$ is appended to $\beta_{i'}$.

Note that instead of having a separate rule creation procedure for initial phrase pairs, we could equally treat them as hierarchical phrase pairs with an empty set of holes.

EXAMPLE 5.3 (SLCFRS rule extraction). Consider the word-aligned sentence pair in figure 56. In the following we list some initial phrase pairs and SLCFRS rules yielded by the extraction procedure.

Initial phrase pairs:

1. jouer $\{5\}$ | to play $\{5,6\}$
2. veux $\{3\}$ | do ... want $\{2,4\}$
3. ne veux plus $\{2,3,4\}$ | do not want ... anymore $\{2,3,4,7\}$
4. ne veux plus jouer $\{2,3,4,5\}$ | do not want to play anymore $\{2,3,4,5,6,7\}$
5. ...

Rules:

1. $\langle \mathrm{X}(\text{jouer}) \to \varepsilon,\ \mathrm{X}(\text{to play}) \to \varepsilon \rangle$
2. $\langle \mathrm{X}(\text{veux}) \to \varepsilon,\ \mathrm{X}(\text{do}\,,\ \text{want}) \to \varepsilon \rangle$
3. $\langle \mathrm{X}(\text{ne veux plus}) \to \varepsilon,\ \mathrm{X}(\text{do not want}\,,\ \text{anymore}) \to \varepsilon \rangle$
4. $\langle \mathrm{X}(\text{ne veux plus jouer}) \to \varepsilon,\ \mathrm{X}(\text{do not want to play anymore}) \to \varepsilon \rangle$
5. $\langle \mathrm{X}(\text{ne } Y_1 \text{ plus}) \to \mathrm{X}_{\boxed{1}}(Y_1),\ \mathrm{X}(Z_1 \text{ not } Z_2\,,\ \text{anymore}) \to \mathrm{X}_{\boxed{1}}(Z_1, Z_2) \rangle$

6. $\langle \mathrm{X}(\text{ne veux plus } Y_1) \rightarrow \mathrm{X}_{\boxed{1}}(Y_1),$
$\mathrm{X}(\text{do not want } Z_1 \text{ anymore}) \rightarrow \mathrm{X}_{\boxed{1}}(Z_1)\rangle$
7. $\langle \mathrm{X}(\text{ne } Y_1 \text{ plus } Y_2) \rightarrow \mathrm{X}_{\boxed{1}}(Y_1)\mathrm{X}_{\boxed{2}}(Y_2),$
$\mathrm{X}(Z_1 \text{ not } Z_2 Z_3 \text{ anymore}) \rightarrow \mathrm{X}_{\boxed{1}}(Z_1, Z_2)\mathrm{X}_{\boxed{2}}(Z_3)\rangle$
8. ...

Rule #1 for example is obtained from phrase pair #1. Phrase pair #2 is discontinuous on the target side. The range vectors created from it are $\boldsymbol{\rho}_{\bar{f}} = \langle\langle 2,3\rangle\rangle$ and $\boldsymbol{\rho}_{\bar{e}} = \langle\langle 1,2\rangle, \langle 3,4\rangle\rangle$ yielding rule #2. It is an SLCFRS rule where the LHS non-terminal has a source fan-out of $v_s = 1$ and a target fan-out of $v_t = 2$ in accordance with the lengths of $\boldsymbol{\rho}_{\bar{f}}$ and $\boldsymbol{\rho}_{\bar{e}}$. Rule #5 is a hierarchical rule. It was created from the initial phrase pair #3 by substituting phrase pair #2. For this rule, $m = 1$,

$$\Omega_{\bar{f}} = \{\{2,3,4\}\},$$
$$\Omega_{\bar{e}} = \{\{2,3,4\},\{7\}\},$$
$$\Omega_{\bar{f}_1} = \{\{3\}\},$$
$$\Omega_{\bar{e}_1} = \{\{2\},\{4\}\}.$$

The one block of $\Omega_{\bar{f}_1}$ is mapped to the variable Y_1; the first block of $\Omega_{\bar{e}_1}$ is mapped to the variable Z_1 and the second block to Z_2. $\boldsymbol{\alpha}$ originates from $\langle(\langle 1,2\rangle(f))Y_1(\langle 3,4\rangle(f))\rangle$, $\boldsymbol{\beta}$ from $\langle Z_1(\langle 2,3\rangle(e))Z_2, (\langle 5,6\rangle(e))\rangle$. □

Phrase pairs #1 and #4 in ex. 5.3 are equally extracted by the standard phrase extraction algorithm used for phrase-based systems. Furthermore, rules #1, #4 and #6 are also generated by the rule extraction procedure of a hierarchical phrase-based, i. e. SCFG-based, system. Rule #6, for example, would usually be written down as

$$\mathrm{X} \rightarrow \langle \text{ne veux plus } \mathrm{X}_{\boxed{1}} \text{ , do not want } \mathrm{X}_{\boxed{1}} \text{ anymore}\rangle$$

However, just as Galley and Manning (2010), we extract many more phrase pairs, namely those which capture discontinuous translation units, e. g. phrase pairs #2 and #3.[1] On top of that, we furthermore extract rules which are *discontinuous* and *hierarchical* at the same time.

1 See the related work in section 5.4.1 for a more detailed comparison of our hierarchical translation system to the phrase-based one by Galley and Manning (2010).

They capture relationships between possibly discontinuous translation units. Rule #5 is such an example. It represents the translation unit $\langle$ne, plus; not, anymore$\rangle$, which is discontinuous on the source and the target side, and its interaction with a second translation unit that is discontinuous on the target side, namely constituent $X_{\boxed{1}}$ with a source fan-out of 1 and a target fan-out of 2. The LHS non-terminal also has a target fan-out of 2, where the gap originates from the position of the infinitival verb in the example training sentence. Rule #7 is similar, however it additionally explicitly models the relationship to the constituent which might represent the infinitival verb ($X_{\boxed{2}}$).

The proposed extraction procedure yields a *monotone* (def. 4.5, p. 133) and *ε-free* SLCFRS (def. 4.6, p. 133), both characteristics which simplify parsing. Source and target side rules are extracted in a way which guarantees that the variables which present the arguments of one RHS non-terminal occur in the same relative order in the arguments of the LHS non-terminal. Furthermore, no ε-rules are created during the extraction procedure: Each LHS argument is created from a block of a block set, which is a partition, and therefore by definition non-empty.

For *rule scoring*, i. e. to estimate the parameters of the translation probabilities $P(r_s|r_t)$ and $P(r_t|r_s)$ and also of the lexical weighting $lex(r_s|r_t)$ and $lex(r_t|r_s)$, we use the same heuristic methodology which is generally applied for phrase-based and hierarchial phrase-based translation grammar learning (Koehn et al., 2003; Och and Ney, 2004; Chiang, 2007). Since no SLCFRS derivations are observable in the training data, we cannot obtain real counts for how often a rule was used in the training data. Instead a distribution is hypothesized based on the extracted rules and their counts, and taken as the real, observed data. Then $P(r_s|r_t)$ and $P(r_t|r_s)$ are estimated as relative frequencies:

$$P(r_s|r_t) = \frac{count(\langle r_s, r_t\rangle)}{count(r_t)} \qquad P(r_t|r_s) = \frac{count(\langle r_s, r_t\rangle)}{count(r_s)}$$

For the calculation of the lexical weighting scores, a word alignment $\mathcal{A}$ between the source terminal symbols $f \in T_s$ which occur in r_s and the target terminal symbols $e \in T_t$ which occur in r_t, and lexical translation probabilities $t(f|e)$ and $t(e|f)$, which are a byproduct of word-aligning the training corpus, are required. Then:

$$lex(r_s|r_t) = \prod_{j \text{ s.t. } (j,i)\in\mathcal{A}} \frac{\sum_{i \text{ s.t. } (j,i)\in\mathcal{A}} t(f_j|e_i)}{|\{i \text{ s.t. } (j,i) \in \mathcal{A}\}|}$$

$lex(r_t|r_s)$ is calculated accordingly. If an extracted rule $\langle r_s, r_t \rangle$ occurs with different alignments $\mathcal{A}$ in the training data, we use the most frequent alignment. This is in accordance with the *Moses* implementation, but different than described in Koehn et al. (2003).

5.2.2 *Practical Considerations*

Extraction Constraints

Enumerating all discontinuous initial phase pairs is exponential in the maximal phrase length. Therefore, in addition to the constraints which are usually applied for SCFG extraction (e. g. maximal phrase length, number of non-terminals and others, see p. 84), we also filter the extracted grammar rules according to the following criteria. They alleviate the problem of spurious ambiguity, help to keep the grammar at a manageable size and allow to implement parsing efficiently.

- The size of a gap, i. e. the number of words which are in a gap, is limited, e. g. to 10.
- Unaligned blocks are not allowed, i. e. each block in an initial phrase pair needs to contain at least one alignment link.
- Phrase pairs with more than one block on either side are only allowed in synchronous spans of $\langle f_1^J, e_1^I \rangle$ which contain at least one complex alignment configuration (section 3.1).
- The source non-terminals as well as the target non-terminals of the SLCFRS rules are restricted to a maximal fan-out. Or, putting it differently, the number of blocks in the initial phrase pairs is limited. We set this maximum to 2 on each side. This is motivated by the experimental results presented in section 4.3. They show that all manual alignments in various data sets can be covered with a $(2, 4_{2|2})$-SLCFRS.

Rule Text Format

The SLCFRS rules are written to a text file during the extraction procedure, and also the complete translation grammar itself is stored in a text file. We accordingly devise a new SLCFRS rule text format. It is

inspired by the format used in *Moses*[2], and extended to also support non-terminals with arguments. In this format, each line contains one rule $\langle r_s, r_t \rangle \in P$ of the grammar $G = (N_s, N_t, T_s, T_t, V_s, V_t, P, S_s, S_t)$ in the following notation:

$$r_s \texttt{ ||| } r_t$$

where r_s and r_t are both LCFRS rules. They are formatted as A B C each, with $x \in \{s, t\}$, and:

A. the arguments of the LHS non-terminal of r_x, delimited by the gap symbol `{,}`, i. e. $(T_x \cup V_x \cup \{\texttt{\{,\}}\})^+$. For ease of notation, a mapping function f_x from variables V_x to natural numbers $\mathbb{N}_0$ is used, such that a variable $v \in V_x$ is then formatted as $\texttt{\{}f_x(v)\texttt{\}}$. E. g. `the {0} cat {,} {1}`.

B. the RHS of r_x, i. e. a sequence of non-terminals with their argument variables, also applying the mapping function f_x. To be able to easily distinguish non-terminals from terminals, non-terminals are framed by square brackets. E. g. `[X(0,2)] [X(1)]`.

C. the LHS non-terminal symbol of r_x, also framed by square brackets. E. g. `[S]`.

The RHS non-terminals in B of r_s and r_t are synchronized by their order in this format, not by indices. See ex. 5.4 for illustration.

EXAMPLE 5.4 (Rule text format). This example shows two SLCFRS rules and the same rules in the SLCFRS rule text format. The second rule would be printed to one line and only contains a line break here because of the narrow line width.

$\langle \mathrm{X}(\text{veux}) \to \varepsilon,\ \mathrm{X}(\text{do}\,,\ \text{want}) \to \varepsilon \rangle$
$\langle \mathrm{X}(\text{ne } Y_1 \text{ plus } Y_2) \to \mathrm{X}_{\boxed{1}}(Y_1)\mathrm{X}_{\boxed{2}}(Y_2),$
$\mathrm{X}(Z_1 \text{ not } Z_2 Z_3 \text{ anymore}) \to \mathrm{X}_{\boxed{2}}(Z_3)\mathrm{X}_{\boxed{1}}(Z_1, Z_2) \rangle$

```
veux [X] ||| do {,} want [X]
ne {0} plus {1} [X(0)] [X(1)] [X] |||
                          {0} not {1} {2} [X(0,1)] [X(2)] [X]
```

The mapping functions of the second rule are $f_s : Y_1 \to 0, Y_2 \to 1$ and $f_t : Z_1 \to 0, Z_2 \to 1, Z_3 \to 2$. Note how the order of the RHS of the

2 See `http://www.statmt.org/moses/?n=Moses.SyntaxTutorial`, accessed on May 15, 2016.

target projection of the second rule is reversed in the rule text format in order to represent the synchronization of the SLCFRS rule. □

Training Implementation

The implementation of the SLCFRS training is similar to the training script of the *Moses* toolkit.[3] It implements a similar sequence of steps and reuses some well-established ideas. However, our training script *slcfrs-extract* supports less options and is focused on one specific task, namely, learning a Hiero translation grammar which supports discontinuous phrases.

Word alignment, symmetrization and the calculation of the lexical translation probabilities are performed using an external tool (see section 2.2.3). Our training script *slcfrs-extract* takes the parallel tokenized corpus, the word alignments of this corpus and the lexical translation tables in both directions, $t(e|f)$ and $t(f|e)$, as input. The rule extraction itself, as described in section 5.2.1, is implemented, in *Python*. It outputs two files, one for each translation direction. Those files are then sorted with *Unix sort* such that identical rules are found in subsequent lines and that rules with identical source sides but different target sides are also found next to each other. Given such input, the rule scoring can be implemented efficiently (again in *Python*). In the end, the two translation tables - one containing $P(r_s|r_t)$ and the lexical weighting feature scores, the other containing $P(r_t|r_s)$ - are merged.

The training in *slcfrs-extract* can make use of several CPU cores to speed up training. The time-consuming steps of rule extraction and rule scoring are parallelized by splitting the corresponding input files into smaller chunks which are processed in parallel and then merged together again afterwards.

5.3 DECODING

The decoder presented in this section closely follows the methodology of current string-to-string SCFG decoders (see section 2.2.5): exhaustive monolingual parsing using the source projection of the grammar and then beam search to find the overall best derivation. The difference is that it is able to produce source and target discontinuities as

3 `https://github.com/moses-smt/mosesdecoder/blob/master/scripts/training/train-model.perl`, accessed on May 16, 2016.

they are represented in SLCFRS rules. The objective is the general SMT objective as defined in equation (2.6): finding the target sequence e generated by the highest scoring derivation d according to the model definition in section 5.1 for a given input sentence f.

5.3.1 *LCFRS Parsing*

We parse the input sentence with a bottom-up CYK parser using the source projection grammar $G_s = proj_s(G)$ of the SLCFRS translation grammar $G = (N_s, N_t, T_s, T_t, V_s, V_t, P, S_s, S_t)$. This corresponds to monolingual Linear Context-Free Rewriting System (LCFRS) parsing (see section 2.1.4), with some additions to ensure valid derivations on the target side.

Using the rules in P, parse items are built. They are of the form $[A, B, \boldsymbol{\rho}, k]$ where $A \in N_s$ is a source side non-terminal symbol and $B \in N_t$ is a target side non-terminal symbol. $\boldsymbol{\rho}$ is a range vector in f of dimension $dim(A)$ identifying which part of the input has been derived by this item. Besides the target side label B itself, the item is also aware of its dimension $k = dim(B)$ in order to ensure valid target side derivations. The details of the parser are presented by means of a weighted deduction system (cf. p. 37).

The SCAN rule creates items for terminal rules, i. e. rules where the source and the target RHS are ε.

$$\textsc{scan:}\quad \frac{}{\omega(\langle r_s, r_t\rangle) : [A, B, \boldsymbol{\rho}, dim(B)]}$$

with the following side conditions:

1. $\langle r_s, r_t\rangle \in P$ with $r_s = A(\boldsymbol{\alpha}) \to \varepsilon$ and $r_t = B(\boldsymbol{\beta}) \to \varepsilon$,
2. there exists a ϕ s.t. $\phi(A(\boldsymbol{\alpha})) = A(\boldsymbol{\rho})$ where ϕ is an instantiation of r_s with respect to f.

By definition of the instantiation, it holds that $\boldsymbol{\rho}(f) = \boldsymbol{\alpha}$.

The COMPLETE operation combines m antecendent items to a new item.

$$\textsc{complete:}\quad \frac{w_1 : [A_1, B_2, \boldsymbol{\rho}_1, k_1] \quad \ldots \quad w_m : [A_m, B_m, \boldsymbol{\rho}_m, k_m]}{w_1 \cdot \ldots \cdot w_m \cdot \omega(\langle r_s, r_t\rangle) : [A, B, \boldsymbol{\rho}, dim(B)]}$$

with the following side conditions:

1. $\langle r_s, r_t \rangle \in P$ with $r_s = A(\boldsymbol{\alpha}) \to A_{1\boxed{l_1}}(\boldsymbol{\alpha}_1) \ldots A_{m\boxed{l_m}}(\boldsymbol{\alpha}_m)$ and $r_t = B(\boldsymbol{\beta}) \to B_{1\boxed{l_1}}(\boldsymbol{\beta}_1) \ldots B_{m\boxed{l_m}}(\boldsymbol{\beta}_m)$ and $m > 0$, and
2. for $0 < i \leq m$, $k_i = dim(B_j)$ where B_j is the target non-terminal with index l_i, and
3. there exists a ϕ s.t. $A(\boldsymbol{\rho}) \to A_{1\boxed{l_1}}(\boldsymbol{\rho}_1) \ldots A_{m\boxed{l_m}}(\boldsymbol{\rho}_m)$ is an instantiation of r_s with respect to f.

The GOAL item is an item which covers the complete input f:

$$\text{GOAL:} \quad w : [S_s, S_t, \langle\langle 0, |f|\rangle\rangle, 1]$$

Intuitively, the condition about the instantiation means that when creating a new item using a specific rule r, the variables and arguments in r need to be consistently replaced with ranges of the input sentence f. Roughly, this means that terminals and variables are instantiated with ranges such that for ranges that are adjacent in an argument of the LHS non-terminal, the concatentation of the two ranges has to be defined, i. e. $r_1 = l_2$ for $\langle l_1, r_1 \rangle$ and $\langle l_2, r_2 \rangle$. See def. 2.34, p. 28 for the formal definition of rule instantiation for LCFRS.

In the presented weighted deduction system, $\omega(\langle r_s, r_t \rangle)$ is the weight of the rule $r = \langle r_s, r_t \rangle$ as defined by the model, see equation (5.6). For a weighted item $w : I$, w is the weight of the parse item I, i. e. the weight of the SLCFRS derivation which lead to I (def. 5.2, p. 171 and equation (5.5)). Note that while in the presented deduction system the weight function is

$$f(w_1, \ldots, w_m) = \omega(r) \cdot \prod_{j=1}^{m} w_j \,,$$

with w_j being the weight of the jth antecedent, $1 \leq j \leq m$, one could equally well use the logarithm of the rule weights together with the weight function

$$f(w_1, \ldots, w_m) = \log \omega(r) + \sum_{j=1}^{m} w_j \,.$$

See also equation (2.7). In fact, an implementation would always use log probabilities or weights for efficiency reasons and to avoid underflow problems when computing with very small probabilities. As the

```
 1: G: an SLCFRS grammar
 2: f: the input string of length J
 3: C = ∅
 4: for all l s.t. 0 < l ≤ J do
 5:     for all range vectors ρ for f of yield length |ρ| = l do
 6:         for all items I = [A, B, ρ, dim(B)] which can be deduced with
            SCAN or COMPLETE using items I_1 ... I_m from C and a rule r of
            rank m from P do
 7:             if I ∉ C then
 8:                 add I to C
 9:             end if
10:             add a hyperedge e with H(e) = I and T(e) = (I_1, ..., I_m) and
                label it r
11:         end for
12:     end for
13: end for
```

Algorithm 7: SLCFRS parsing procedure for translation

weights in the deduction system do not include the score for the language model feature $P_{LM}(e)$, this deduction system cannot be used to find the best translation according to the model in section 5.1. We will turn to the search for the derivation with the best weight including the language model score in the next section.

As for SCFG-based translation (section 2.2.5), the presented deduction system is used to compute the parse hypergraph of the input f using the translation grammar G. This is done by deducing the items in an ordered manner. Smaller items, i. e. items which cover less input words from f, come before larger items to make sure that every item is created after all its possible antecedents. Items are stored in a chart $\mathcal{C}$. Equal items are combined, thereby retaining their origin via hyperedges. The pseudocode is provided in algorithm 7.

In contrast to context-free rules, formally the order of the RHS nonterminals of r of an LCFRS or SLCFRS does not matter. Nevertheless, we assume for the ease of formulation and implementation that each RHS is an ordered list, e. g. ordered by the natural order of the indices with which the non-terminals are annotated to indicate the synchronization. The order of the items in the tail $I_1, \ldots, I_m$ then corresponds to this order.

Rules which behave identically in the described parsing procedure are bundled together. Such a *rule bundle* comprises SLCFRS rules $\langle r_s, r_t \rangle$ with identical r_s and identical LHS non-terminal B of identical $dim(B)$ of r_t.

The parsing procedure in algorithm 7 does not make use of the weights specified in the weighted deduction system because the complete parse hypergraph of f needs to be computed anyway to then extend it to the actual search hypergraph in the next step. Nevertheless, the weights of the weighted SLCFRS can be used as estimates for the final weights in order to perform, for example, early pruning of unlikely translation hypothesis along hyperedges with very low weights.

EXAMPLE 5.5 (SLCFRS deduction and parse hypergraph for translation). Let us consider an SLCFRS translation grammar fragment G as specified by the following rules

$$
\begin{aligned}
r_1 &: \langle \mathrm{X}(\text{ne}, \text{plus}) \to \varepsilon \;, \mathrm{X}(\text{not}, \text{anymore}) \to \varepsilon \rangle \\
r_2 &: \langle \mathrm{X}(\text{ne}, \text{plus}) \to \varepsilon \;, \mathrm{X}(\text{not}, \text{any longer}) \to \varepsilon \rangle \\
r_3 &: \langle \mathrm{X}(\text{manger}) \to \varepsilon \;, \mathrm{X}(\text{to eat}) \to \varepsilon \rangle \\
r_4 &: \langle \mathrm{X}(\text{manger}) \to \varepsilon \;, \mathrm{X}(\text{to consume food}) \to \varepsilon \rangle \\
r_5 &: \langle \mathrm{X}(Y_1 \text{ veux } Y_2 Y_3) \to \mathrm{X}_{\boxed{1}}(Y_1, Y_2)\, \mathrm{X}_{\boxed{2}}(Y_3) \;, \\
&\qquad\qquad \mathrm{X}(\text{does } Z_1 \text{ want } Z_2 Z_3) \to \mathrm{X}_{\boxed{1}}(Z_1, Z_3)\, \mathrm{X}_{\boxed{2}}(Z_2) \rangle \\
r_6 &: \langle \mathrm{X}(\text{ne veux plus } Y_1) \to \mathrm{X}_{\boxed{1}}(Y_1) \;, \\
&\qquad\qquad \mathrm{X}(\text{does not want } Z_1 \text{ anymore}) \to \mathrm{X}_{\boxed{1}}(Z_1) \rangle
\end{aligned}
$$

and the input

$$f = \text{il ne veux plus manger .}$$

Rules r_1 and r_2, and r_3 and r_4 form rule bundles respectively. Figure 57 depicts the parse hypergraph which represents the parse forest which is generated when parsing $f_2 \ldots f_5$ with G using the specified deduction system. The order of the tail nodes is not represented in this picture.[4]

Using r_3 (or r_4), the item $[\mathrm{X}, \mathrm{X}, \langle\langle 4, 5\rangle\rangle, 1]$ is deduced with the SCAN operation. With the argument numbering $\xi_{r_{3_s}}(1) = \text{manger}$ and $\phi = \langle\langle 4, 5\rangle\rangle$, $\mathrm{X}(\langle\langle 4, 5\rangle\rangle) \to \varepsilon$ is an instantiation of r_{3_s} with respect to f. Rule r_1 and r_2 lead to the item $[\mathrm{X}, \mathrm{X}, \langle\langle 1, 2\rangle, \langle 3, 4\rangle\rangle, 2]$.

4 Weights are neglected in this example.

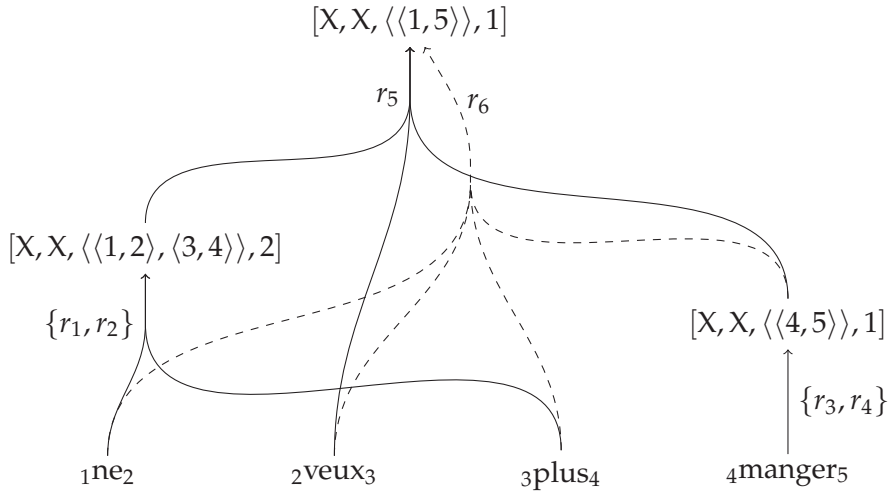


Figure 57: SLCFRS parse hypergraph for translation

The item $[X, X, \langle\langle 1,5\rangle\rangle, 1]$ can be created from the antecedents $[X, X, \langle\langle 1,2\rangle, \langle 3,4\rangle\rangle, 2]$ and $[X, X, \langle\langle 4,5\rangle\rangle, 1]$ and the rule r_5 using the COMPLETE deduction rule. With $\xi_{r_{5_s}}(1) = Y_1$, $\xi_{r_{5_s}}(2) = \text{veux}$, $\xi_{r_{5_s}}(3) = Y_2$, $\xi_{r_{5_s}}(4) = Y_3$ and

$$\phi = \langle\langle 1,2\rangle, \langle 2,3\rangle, \langle 3,4\rangle, \langle 4,5\rangle\rangle \ ,$$

$X(\langle 1,5\rangle) \rightarrow X_{\boxed{1}}(\langle 1,2\rangle, \langle 3,4\rangle)\, X_{\boxed{2}}(\langle 4,5\rangle)$ is an instantiation of r_{5_s} with respect to f. Note also that the fan-out of the *first* RHS target non-terminal $X_{\boxed{1}}$ in r_5 is 2, just as the target dimension k of the *first* antecedent, whereas the fan-out of the *second* RHS target non-terminal $X_{\boxed{2}}$ is 1 which fits the target dimension of the *second* antecedent.

The application of the rule r_6 (dashed hyperedge) leads to the same item $[X, X, \langle\langle 1,5\rangle\rangle, 1]$. Overall the hypergraph represents six derivations of this item. □

5.3.2 *Integration of the Language Model*

As explained in section 5.1, the n-gram language model feature is not local to the SLCFRS rules and can therefore not be integrated into the deduction system in the same manner as the other features. When using SCFG as the base formalism for SMT, the same issue arises (see section 2.2.5). For SCFG, we have described how this increases the search space and detailed one of the most popular approaches to deal

with it, which is cube pruning. We will now describe how we extend this approach to SLCFRS decoding.

While the deduction rules and parse items which have been presented for parsing with a grammar G depend on whether G itself is a (S)CFG or (S)LCFRS, the parse hypergraph $\langle V, E \rangle$ (or parse forest) created during parsing an input f with G is a general concept applying to both CFG and LCFRS. This is because in both cases the derivations correspond to context-free trees (cf. section 2.1.1 for LCFRS). In such a hypergraph, a hyperedge $e \in E$ represents the successful application of a rule r (or of several rules if e is associated with a rule bundle) deriving the parse item $H(e)$ from the antecedents in $\boldsymbol{T}(e)$, no matter whether G is a (S)CFG or a (S)LCFRS.

Recall the objective

$$\hat{e} \approx \mathfrak{e}\left(\underset{d \text{ s.t. } \mathfrak{f}(d)=f}{\arg\max}\ P_{LM}(\mathfrak{e}(d))^{\lambda_{LM}}\omega(d)\right), \tag{5.9}$$

cf. equations (5.7) and (5.8). In order to make its computation tractable, we need to take advantage of the structure of the search space. During deductive parsing, we have already taken advantage of the independence assumptions in G resulting in the parse hypergraph. Roughly speaking, it covers $\omega(d)$ in the equation, including all features except the language model. We now extend it to the final weighted search hypergraph where the weights include the language model score, thereby taking into account the independence assumptions of language modeling using n-grams, see equation (2.3). This leads to the definition of *search items*, i. e. nodes in the search hypergraph, which extend the parse items by the information which is necessary to compute language model scores when combining subderivations.

Consider calculating a (partial) language model score for a translation hypothesis. Whether this hypothesis is generated by an SCFG or an SLCFRS, results in the following major difference: in the case of SLCFRS, the translation hypothesis, i. e. the target sequence, is not necessarily continuous. It rather is a tuple of continuous blocks of target words. Let us consider, e. g., *does not want . . . anymore* as the target yield of an SLCFRS derivation, and assume the usage of a bigram language model for now ($n = 2$). Only once the gap between *want* and *anymore* is filled, the language model probability which is con-

tributed by *anymore* can be determined.[5] Even though the language model probability for *want* can be fully determined in this case, as its previous word is known, the identity of *want* is crucial for computing the language model score of the word which will fill the gap towards the left, i. e. directly right of *want*. This means that in terms of language model scoring, each continuous block in the target yield of an SLCFRS non-terminal can be considered analogously to an SCFG non-terminal and the target side words which are generated by it. For the given example, these are *does not want* and *anymore*.

For SCFG decoding, one *language model state* s, roughly consisting of the $n-1$ left and right target boundary words, is recorded in each item in the search hypergraph. Accordingly, for SLCFRS decoding, we store a list of such language model states $(s_1, \ldots, s_k)$ in the search items, one for each block in the target yield of the corresponding node. The length of this list corresponds to the k in the parse item $[A, B, \boldsymbol{\rho}, k]$ which is the fan-out of the target non-terminal B. The list of language model states is then used when computing the language model score along a hyperedge e from the arguments of the LHS non-terminal of r_t of the synchronous rule $\langle r_s, r_t \rangle$ which is associated with e. When encountering a variable $Z_j^{(i)} \in V_t$ which constitutes the jth argument of the ith RHS non-terminal of r_t, we use the jth language model state s_j of the ith antecedent $T_i(e)$.

Example 5.6 (SLCFRS search hypergraph for translation). We continue with ex. 5.5, p. 185. Figure 58 shows the corresponding search hypergraph which expands the parse hypergraph. It assumes the use of a bigram language model ($n = 2$). The search items therefore record the one rightmost and leftmost word of each block of each translation hypothesis. □

Exploring the corresponding search hypergraph to find the best derivation amounts to a theoretical time complexity of

$$\mathcal{O}(|P| J^{v_s \cdot (u+1)} |T|^{v_t u \cdot 2(n-1)})$$

with $|P|$ the number of rules in the SLCFRS G, J the length of the input sequence f, $|T|$ the number of possible target side words, u the rank of G, typically $u = 2$, and n the order of the language model. The

5 Actually, the gap does not need to be completely filled. It just needs to be filled in a way such that the $n-1$ words left of *anymore* are fixed.

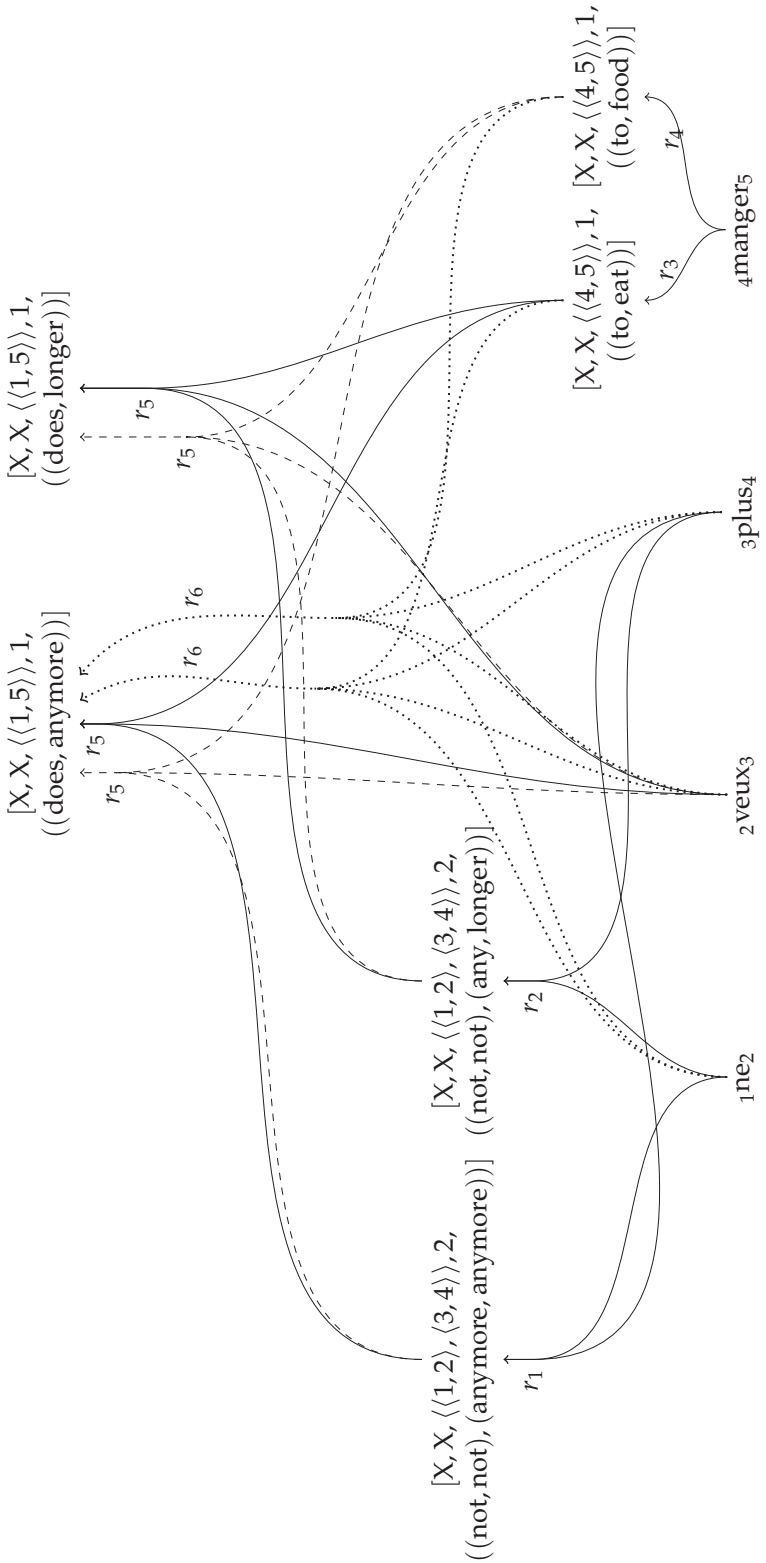


Figure 58: SLCFRS search hypergraph for translation. Some hyperedges are dashed for better readability.

new factors in the exponents compared to the complexity of SCFG decoding are v_s, the source fan-out of G, and v_t, the target fan-out of G. In fact, SCFG decoding is the special case in which $v_s = v_t = 1$. To make the search feasible, we implement cube pruning as specified in algorithm 6.

Finally, to obtain the translation associated with a derivation d of a search item, e. g. $D_1(\textsc{goal})$, the hyperedges along the derivation d are followed and the translation is assembled according to the target projections of the rules $r \in d$.

5.3.3 *Implementation Details*

The decoder, called *discodec*, is implemented in C++.

Language Model Toolkit

discodec includes the code from the *KenLM Language Model Toolkit*.[6] *KenLM* is one of the most popular toolkits for n-gram language modeling, amongst others because of fast and low-memory queries, and fast and scalable model estimation. Other SMT toolkits such as *Moses*, *cdec* and *Phrasal* also distribute *KenLM* and build it with the decoder.

We use *KenLM*'s type `ChartState` for the language model states which we store with the search items and code against the interface provided in `lm/left.hh`.

Terminal Instantiation

In the implementation, in order to build the parse hypergraph according to algorithm 7, we first perform a terminal instantiation step. This step replaces all terminals in all rules with all possibles ranges with respect to the input sentence f.

A *terminal instantiation* of a rule r is a partial instantiation of r (def. 2.34, p. 28) in which only occurrences of terminals and ε are instantiated, but no variables. Recall the definition of the argument numbering ξ_r of r (def. 2.33, p. 28).

6 The author of *KenLM*, K. Heafield, recommends to distribute the language modeling code with the decoder. The documentation of how to integrate *KenLM* and use its interface is provided here: `http://kheafield.com/code/kenlm/developers/`. `kenlm.tar.gz` was downloaded in April 2014.

DEFINITION 5.7 (Terminal instantiation). Let $G = (N, T, V, P, S)$ be an LCFRS. Let $w \in T^*$ be a string. For a rule $r \in P$ with its argument numbering ξ_r, the following is defined:

1. A *terminal instantiation* of r with respect to w is given by a ξ_r^{max}-dimensional range vector ϕ' where $\phi'(i)$, $1 \leq i \leq \xi_r^{max}$, contains the range to which $\xi_r(i)$ is bound. If a $\phi'(i)$ is still unbound, it contains the symbol *?* for *unknown*. Thereby, for $0 \leq j < |w|$,
   a) if $\xi_r(i)$ is an occurrence of a terminal t, it must be mapped on a range $\phi'(i) = \langle j, j+1 \rangle$ with $w_{j+1} = t$, and
   b) if $\xi_r(i)$ is an occurrence of ε, it must be mapped on a range $\phi'(i) = \langle j, j \rangle$.
2. Applying ϕ' to a non-terminal $A(\alpha)$, notated as $\phi'(A(\alpha))$, is defined as a mapping of all occurrences of terminals and ε in α to elements of ϕ' such that each $\xi_r(i)$ which is an occurrence of a terminal or ε is mapped to $\phi'(i)$, for $1 \leq i \leq \xi_r^{max}$. If the result is defined, i. e. if the range images of adjacent terminals and ε can be concatenated, then it is called a *terminal instantiated non-terminal*, and if $A(\alpha)$ is the LHS of a rule r, then the result of the mapping is called a *terminal instantiated rule*. □

The given definition of terminal instantiation also covers ε-rules. Note however that the grammars we deal with for machine translation are ε-free.

EXAMPLE 5.8 (Terminal Instantiation). Let $w = {}_0a_1b_2c_3a_4b_5c_6$ be our input and $r = \mathrm{A}(abX_1, abX_2) \to \mathrm{A}(X_1, X_2)$ an LCFRS rule. The following range vectors ϕ'_1 and ϕ'_2 are valid terminal instantiations of r with respect to w:

$$\phi'_1 = \langle \langle 0,1 \rangle, \langle 1,2 \rangle, ?, \langle 3,4 \rangle, \langle 4,5 \rangle, ? \rangle$$
$$\phi'_2 = \langle \langle 3,4 \rangle, \langle 4,5 \rangle, ?, \langle 0,1 \rangle, \langle 1,2 \rangle, ? \rangle$$

ϕ'_1 applied to r yields $\mathrm{A}(\langle 0,2 \rangle X_1, \langle 3,5 \rangle X_2) \to \mathrm{A}(X_1, X_2)$, and ϕ'_2 applied to r yields $\mathrm{A}(\langle 3,5 \rangle X_1, \langle 0,2 \rangle X_2) \to \mathrm{A}(X_1, X_2)$.

The following range vector ϕ'_3 does not lead to a terminal instantiated rule since the concatenations of $\phi'_3(1)$ and $\phi'_3(2)$, and $\phi'_3(4)$ and $\phi'_3(5)$ fail.

$$\phi'_3 = \langle \langle 0,1 \rangle, \langle 4,5 \rangle, ?, \langle 3,4 \rangle, \langle 1,2 \rangle, ? \rangle$$

□

1: G: an SLCFRS grammar
2: f: the input string of length J
3: $\mathcal{C}_P = \emptyset$
4: **for all** $r_s = A(\boldsymbol{\alpha}) \rightarrow A_{1\boxed{l_1}}(\boldsymbol{\alpha}_1) \ldots A_{m\boxed{l_m}}(\boldsymbol{\alpha}_m)$ s.t. $\langle r_s, r_t \rangle \in P$ **do**
5: Let ξ_{r_s} be the argument numbering of r_s
6: **for all** terminal instantiations ϕ' of dimension $\xi_{r_s}^{max}$ with respect to f s.t. $\phi'(A(\boldsymbol{\alpha}))$ is defined **do**
7: **if** $(\langle r_s, r_t \rangle, \phi')$ passes filter **then**
8: add $(\langle r_s, r_t \rangle, \phi')$ to $\mathcal{C}_P$
9: **end if**
10: **end for**
11: **end for**

Algorithm 8: Parsing initialization

Algorithm 8 shows how the terminal instantiation is applied during parsing initialization. The result of this step are *terminal instantiated rules* (r, ϕ') for f. They are stored in the rule chart $\mathcal{C}_P$.

For terminal rules, this step covers the SCAN operation (algorithm 7, line 6). This means that parse items can be directly created from those terminal instantiated terminal rules, by applying ϕ' to r, and they are added to $\mathcal{C}$. During the actual parsing, we are then only concerned with how variables are instantiated when combining items from $\mathcal{C}$ and (terminal instantiated) rules from $\mathcal{C}_P$ with the COMPLETE operation. Obviously, we perform the terminal instantiation only once per rule bundle.

Rule Filtering

As our rules are all monotone and ε-free, we can make certain assumptions about rule instantiations. For efficient processing, we filter out terminal instantiated rules which will never lead to a successful parse during the initialization, see algorithm 8, line 7. The details of the rule filtering are presented in algorithm 9.

Due to monotonicity, only rule instantiations ϕ of r_s for which the following holds need to be considered, for $0 < i_1 < i_2 \leq \xi_{r_s}^{max}$:

$$\phi(i_1).r \leq \phi(i_2).l$$

This in turn also holds for the terminal instantiations ϕ'. In the implementation, we actually only generate terminal instantiations meeting

```
 1: f: the input string of length J
 2: (⟨r_s, r_t⟩, φ'): a rule which is terminal instantiated with respect to f
 3: cnt = 0
 4: last = 0
 5: for all i s.t. 0 < i ≤ ξ_{r_s}^{max} do
 6:     if ξ_{r_s}(i) is a variable then
 7:         cnt = cnt + 1
 8:     end if
 9:     if i > 1 and ξ_{r_s}(i) and ξ_{r_s}(i − 1) do not belong to the same argument
        of r_s then
10:         cnt = cnt + 1
11:     end if
12:     if ξ_{r_s}(i) is an occurrence of a terminal then
13:         if last + cnt > φ'(i).l then
14:             return false
15:         end if
16:         cnt = 0
17:         last = φ'(i).r
18:     end if
19: end for
20: if last + cnt > J then
21:     return false
22: end if
23: return true
```

Algorithm 9: Filter for terminal instantiated rules

this condition in the first place, but they would also be filtered by algorithm 9 (see lines 13 and 17).

Due to ε-freeness, we assume that each variable in a rule must be mapped to a range whose yield's length is at least one. In addition, during parsing, we do not allow for empty gaps to avoid spurious ambiguity. We therefore assume that each gap must later be filled by a range whose yield's length is at least one. These constraints are realized by counting the number of variables (line 7) and gaps (line 10) between two occurrences of terminals, holding the current value in the variable named *cnt*, and making sure that the range between the ranges to which the terminals are mapped is large enough to accommodate *cnt* variables and filled gaps.

EXAMPLE 5.9 (Rule Filtering). Let us consider

$$r_s = \mathrm{A}(abX_1, abX_2) \rightarrow \mathrm{A}(X_1, X_2)$$

The terminal instantiation $\mathrm{A}(\langle 0,2\rangle X_1, \langle 3,5\rangle X_2)$ with respect to $f = {}_0a_1b_2c_3a_4b_5c_6$ does not pass the filter since, to account for X_1 and the gap later in the derivation, at least an input range of length 2 would be necessary between the first b and the second a, but only the c is available. The terminal instantiation $\mathrm{A}(\langle 0,2\rangle X_1, \langle 4,6\rangle X_2)$ with respect to $f = {}_0a_1b_2c_3c_4a_5b_6$ does not pass the filter either. While it has two cs available to potentially fill X_1 as well as the gap, no input material is left at the end to fill X_2, i. e. the condition in line 20 applies. The terminal instantiation $\mathrm{A}(\langle 0,2\rangle X_1, \langle 4,6\rangle X_2)$ with respect to $f = {}_0a_1b_2c_3c_4a_5b_6c_7$ passes the filter. □

Pruning Techniques

We furthermore implement different standard pruning methods, such as limiting the number of rules in a rule bundle, and limiting the number of incoming hyperedges for one parse item by pruning those with low SLCFRS weights.

Non-terminal S

In the hierarchical phrase-based models, the non-terminal S has a special meaning. It is used only in the glue rule and the rules which model the start and the end of the sentence, see section 5.1.2. We adopt this interpretation and make sure that items for the non-terminal S are only created for range vectors $\langle\langle 0, j\rangle\rangle$ with $0 < j \leq J$.

$(2,2)$-LCFRS Parser

The grammar that we extracted has a specific form, namely rank 2 and fan-out $4_{2|2}$, see section 5.2.2. We thus implement a specific parser for $(2,2)$-LCFRS instead of a generic LCFRS parser as presented in the deduction system. Accordingly, the range vector $\boldsymbol{\rho}$ of an item I has the form $\langle\langle i_1, j_1\rangle, \langle i_2, j_2\rangle\rangle$, where i_2 and j_2 are undefined if the yield of $\boldsymbol{\rho}$ is continuous. Such range vectors can be stored and retrieved more efficiently than general range vectors, i. e. for full LCFRS. In the latter case they are typically implemented as bit strings of the size of the input sentence, e. g. in Maier (2013, section 6.3.1). Also parsing time

complexity is directly dependent on the fan-out v_s of the monolingual grammar G_s: $\mathcal{O}(|G_s|\, J^{v_s \cdot (u+1)})$ with rank $u = 2$ and fan-out $v_s = 2$ in our specific case.

Furthermore, our extracted grammar also has a target fan-out of $v_t = 2$. This means that the list of language model states has a maximal length of 2. However, this limit is not hard-coded in the implementation.

k-best Translations

Obtaining the *k*-best translations for a given input sentence is essential for parameter tuning, i. e. for optimizing the weights of the log-linear model. We implement *k*-best extraction on the search hypergraph that results from cube pruning (section 5.3.2), using the lazy strategy from Huang and Chiang (2005). The pseudocode is shown in algorithm 5.

5.4 RELATED WORK

In this section, we review previous work on statistical machine translation systems which are as well expressive enough to model complex alignment configurations and allow for discontinuities.

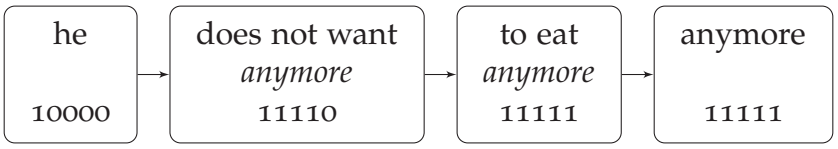
5.4.1 *Non-hierarchical Machine Translation with Discontinuous Phrases*

The most notable work in this area is Galley and Manning (2010). Building upon the idea of a translation model proposed by Simard et al. (2005), a phrase-based translation system is proposed which allows for *discontinuous phrase pairs*, thereby enabling the system to generate the complex alignment configurations. See section 5.2.1 for a definition of discontinuous phrase pairs. Galley and Manning (2010) use the suffix array technique of Lopez (2007), originally devised for hierarchical phrase-based translation, to compactly store the training data and retrieve all applicable phrase pairs for each input sentence. For decoding, a conventional string-based beam search algorithm is extended which employs different stacks to store the translation hypotheses according to the number of input words which are covered. Translation hypotheses are still generated from left to right, but the continuous target-side blocks of discontinuous phrase pairs are appended independently. Each translation hypothesis features a list of

waiting blocks of discontinuous phrases which still need to be appended since the first block is already part of the hypothesis. Before expanding a hypothesis in the conventional way, the beam search algorithm may select one or several of the waiting blocks to append them first.

EXAMPLE 5.10 (Discontinous phrase-based translation). This example demonstrates the approach of Galley and Manning (2010). We show one particular decoder search path for the French input sequence

$f =$ il ne veux plus manger



It makes use of the following phrase pairs, where #3 is discontinuous on the target side:

1. il | he
2. manger | to eat
3. ne veux plus | does not want ... anymore

Each (partial) translation hypothesis is depicted here as the generated target string, the waiting phrasal blocks (*anymore*) and a bit string indicating which source words are already covered. Scores and the other competing hypotheses are omitted. □

Besides the standard features of phrase-based translation (see section 2.2.4), Galley and Manning (2010) add two features to softly constrain the usage of the discontinuous phrase pairs: (a) a feature which is the sum of the lengths of all target gaps, and (b) a feature which is the number of discontinuous phrase pairs which are in cross-serial discontinuous translation unit (CDTU) or bonbon configurations. They also need to adapt the computation of the distortion feature due to the discontinuous phrase pairs.

The proposed discontinuous phrase-based approach has been implemented as part of the open-source SMT toolkit *Phrasal*.[7] It is compared against a conventional phrase-based system (*Moses*) and a hierarchical phrase-based system (*Joshua*), i. e. one that uses a 2-SCFG

7 *Phrasal* (Green et al., 2014) is documented at `http://nlp.stanford.edu/phrasal/`.

for translation modeling and a CYK-based decoder, for Chinese-to-English translation. In the conducted experiments, the discontinuous phrase-based system outperforms Moses by 0.77 and Joshua by 1.03 BLEU points. This advantage is attributed by the authors to the support of discontinuous phrases, more specifically the enlarged space of alignment configurations that can be induced compared to phrase-based and hierarchical phrase-based systems, and benefits of the extended phrase pair inventory since the decoder chooses larger phrase pairs than the Moses decoder.

In some sense, our work of a hierarchical machine translation system with discontinuous phrases is the hierarchical, tree-based counterpart to the phrase-based approach of Galley and Manning (2010). Our building blocks, the possibly discontinuous initial phrase pairs, e. g. in ex. 5.3, p. 176, are also the building blocks of the discontinuous phrase-based system proposed in Galley and Manning (2010). In addition, we however also make use of *hierarchical discontinuous rules*, e. g. rules #5, repeated here for your convenience, and #7 in ex. 5.3.

$$\langle \mathrm{X}(\text{ne } Y_1 \text{ plus}) \rightarrow \mathrm{X}_{\boxed{1}}(Y_1),\ \mathrm{X}(Z_1 \text{ not } Z_2\text{ , anymore}) \rightarrow \mathrm{X}_{\boxed{1}}(Z_1, Z_2) \rangle$$

Hierarchical discontinuous rules are not used in the other two prevalent state-of-the-art SMT approaches, namely neither in (discontinuous) phrase-based systems nor in tree-based (2-SCFG) systems. Our proposed translation grammar rules unify two types of *gaps* of previous approaches:

a) *gaps* in the sense of *non-terminals* which are inserted into longer phrases when hierarchical rules are created, as in Chiang (2007); their purpose is a better generalization of the translation rules, and

b) *gaps* in the sense of *discontinuities* in the yield of a translation rule, on the source side, on the target side or both, driven by the idea of allowing for more flexible phrases such that the generated alignment structures are not restricted.

The idea to use discontinuous phrase pairs in a phrase-based system has already been suggested in Simard et al. (2005). However, in their model, each gap symbol represents exactly one word, making this approach less general and more prone to data sparsity than the one by Galley and Manning (2010).

The model and system by Crego and Yvon (2009) is also inspired by Simard et al. (2005) in that it allows phrases to have gaps. Their work is however part of the *n-gram-based SMT* framework (Mariño et al., 2006; Crego et al., 2005), which is a very different approach to SMT compared to the predominantly used framework which is in focus in this thesis as presented in section 2.2.

5.4.2 *Tree-based Machine Translation Beyond CFG*

Other grammar formalisms beyond SCFG which have been proposed for translation modeling in the literature have been presented in section 4.4.2. In this section, we will pick up again those formalisms which indeed serve as the basis for SMT systems.

GenPar

The idea of *SMT by parsing* (Melamed, 2004) has been implemented in the John Hopkins University Summer Workshop 2005. The resulting toolkit *GenPar* uses Generalized Multitext Grammar (GMTG) (see p. 155) as its translation model formalism and is available for prototyping.[8] Even though it was supposed to be a reference implementation for further research in the area of tree-based SMT, GenPar seems to not have gained momentum. The latest available version is from the year 2006.[9] To the best of our knowledge, no GenPar system has been trained on a reasonably large data set for comparison with other SMT toolkits and approaches. The baselines published in the project report (Burbank et al., 2005) and the few experimental results based on GenPar, e. g. Vičič and Brodnik (2008) for Slovenian-English, only use a few thousand sentence pairs for training and appear to be rather weak.

Beyond GenPar, we do not know of any efforts of building a state-of-the-art SMT toolkit and system which models discontinuities with GMTG. In our view, a *state-of-the-art SMT system* makes use of at least the standard features of the log-linear model for SMT, integrates the language model scoring during decoding, e. g. using cube pruning, is trained on large-scale data sets, or provides competitive translation results.

8 `http://nlp.cs.nyu.edu/GenPar/GenPar.html`, accessed on January 7, 2016.
9 `http://nlp.cs.nyu.edu/GenPar/ReleaseLog`, accessed on January 23, 2017.

MBOT/STSSG

Both the non-contiguous version of Synchronous Tree Sequence Substitution Grammar (STSSG), which is able to induce the complex alignment configurations as it generates LCFRS translations (see p. 162), and the somewhat less expressive Multi Bottom-up Tree Transducer (MBOT) (see p. 165) serve as base formalisms in SMT systems. They have been proposed to improve syntax-based approaches to machine translation which make use of syntactic annotation, and are primarily used for *tree-to-tree* and *string-to-tree translation* respectively.

Non-contiguous STSSG and the corresponding SMT system are described in Sun et al. (2009). Extending previous work for Synchronous Tree-Substitution Grammar (STSG) learning, they provide an STSSG rule extraction algorithm which operates on aligned sentence pairs including a parse tree for each of the source and the target sentences. In order to not extract spurious rules, they require either the source or the target side of a rule to be contiguous. The decoder is not strictly speaking an STSSG decoder in the same sense as we have described decoding as parsing with other synchronous formalisms. It is a bottom-up, three step heuristic, filling up increasingly large spans, similar to what a chart decoder would do. The target sides of the STSSG rules, however, do not provide hard constraints for the reordering of the target constituents. The gaps in the rules, i. e. the non-contiguities, are ignored during language model scoring. No theoretical complexity results or practical runtime numbers are provided.

The authors report results for Chinese-to-English experiments on a rather small training data set.[10] In particular, they evaluate the impact of different rule types on translation quality. They find that systems which use the non-contiguous rules outperform systems which employ contiguous STSSG. Allowing for non-contiguity only on the source side is more effective than allowing for non-contiguity only on the target side. This is in-line with the observations of Galley and Manning (2010), even though their rules are non-hierarchical and not constrained by parse trees (see section 5.4.1). While the latter authors speculate that making effective use of gaps on the target side is more difficult than on the source side, because the first is a generation task, the effect might also be due to the specific language pair which was used in both sets of experiments.

10 The FBIS corpus consists of roughly 250k Chinese-English parallel sentences.

Sun et al. (2009) also find that when allowing maximally one gap on each side in the rules, as compared to no gaps, the BLEU score increases significantly. However, allowing more gaps does not further increase the translation quality. This finding further corroborates our earlier experiments where we concluded that allowing for one gap or discontinuity on each side in a synchronous formalism is a good fit for a translation modeling formalism. Sun et al. (2009)'s experiments however differ considerably from ours in section 4.3 in that they perform actual model training and decoding using automatic alignments, they only investigate one language pair and translation direction, and they use a tree-to-tree approach, i. e. constrain the extracted rules by parse trees on both sides.

Using MBOT as the base formalism for SMT is described in Braune et al. (2013) and Seemann et al. (2015a). Recall that MBOT does not allow discontinuities on the source side and therefore does not induce all complex alignment configurations. The decoding is implemented as a branch of *Moses*' syntax-based system. As the source-side derivations are still context-free, no adaptions are necessary for the parsing part of the decoder. The representation of the rules and the hypothesis expansion are adjusted as well as the language model scoring, similar to our approach in section 5.3.2. Cube pruning is used as well to make language model scoring of partial hypothesis feasible.

In contrast to Sun et al. (2009), larger-scale experiments are conducted using the *Europarl* corpus (Koehn, 2005) for English to German and the *MultiUN* corpus (Eisele and Chen, 2010) for English to Arabic and Chinese. In string-to-tree decoding setups, the MBOT-based system outperforms the SCFG-based baseline system for all three language pairs, but not necessarily a phrase-based *Moses* system (Seemann et al., 2015a). In Seemann et al. (2015b), additionally, string-to-string, i. e. hierarchical phrase-based, systems are trained for English to German and Chinese. For both language pairs, the MBOT-based system outperforms its string-to-tree and tree-to-tree counterpart as measured by BLEU. However, it does not improve over the SCFG-based baseline, although the difference in BLEU score is small. An analysis shows that very few discontinuous rules have been used when decoding the test set. These results stand in contrast to the results of Galley and Manning (2010) who achieve significant improvements over a phrase-based and an SCFG-based system by allowing for discontinuous phrases for Chinese-to-English translation (see sec-

tion 5.4.1). One obvious difference between the two approaches is that the phrase-based approach offers discontinuous phrases on both sides, whereas MBOT only offers gaps on the target side and thus is not able to induce all complex alignment configurations. The experiments of Galley and Manning (2010) furthermore indicate that allowing source discontinuities is more crucial than allowing target discontinuities. We can however only speculate about the reasons for the contrary findings of Galley and Manning (2010) and Seemann et al. (2015b). Besides the necessity for gaps being a characteristic of the language (pair), it might also be *generally* more beneficial to allow for gaps on the source side than on the target side since this allows to match rather distant words in one (discontinuous) phrase.

In Seemann and Maletti (2015), string-to-tree experiments for English to Polish and English to Russian, both morphologically rich languages with a rather free word order, are performed. For the trees on the target side, potentially non-projective dependency parses are transformed to projective dependency trees and then to constituency structures. While the corresponding MBOT systems yield a considerable improvement over SCFG-based baseline systems respectively, again, corresponding phrase-based and hierarchical phrase-based systems yield better results in terms of BLEU. No results for string-to-string systems using MBOT as the translation grammar formalism are reported.

Others

Using a restricted, but non-linear form of Range Concatenation Grammar (RCG) to model translational equivalence beyond the alignment capacity of 2-SCFG has been advocated by Søgaard (2008b). See section 4.4.2, p. 153f., for the details. Even though there has been a suggestion for learning corresponding grammar rules from parallel corpora (Søgaard, 2008a), to the best of our knowledge, no corresponding SMT system has been built.

Synchronous Tree-Adjoining Grammar (STAG), in particular Synchronous Tree-Insertion Grammar (STIG), serves as the base formalism in some SMT systems (e. g. Nesson et al., 2006; DeNeefe and Knight, 2009; Carreras and Collins, 2009; Liu et al., 2011). They mostly use syntactic annotation (i. e. parse trees) on the source and/or the target side and usually outperform an SCFG/STSG-based baseline.

Nesson et al. (2006) is an exception to that pattern: A probabilistic STIG featuring only one non-terminal X is induced using EM to estimate the parameters. The search space is a priori pruned based on given automatically learned word alignments. Experiments are conducted for German-to-English translation, but using a very small data set (∼15k training sentences and 100 test sentences).

5.5 CONCLUSION AND FUTURE WORK

In this chapter, we presented the first hierarchical phrase-based machine translation system which features discontinuous phrases both on the source and on the target side. It is at the same time the first SMT system which uses SLCFRS as the translation grammar formalism. Since SLCFRS is a direct extension to SCFG, previous work on hierarchical phrase-based translation, in particular the model definition, training and decoding, could be extended to SLCFRS in a more or less direct manner.

The described training and decoding procedures have been implemented. In chapter 6, a corresponding machine translation system will be trained and evaluated using the described implementation.

The SMT system as presented in this chapter has certain *shortcomings* which we point our here. Addressing them remains future work.

The learning of phrase-based and hierarchical phrase-based models in their original formulation generally involves a set of heuristically fixed constraints (see section 2.2.5). With our more powerful model, the size of the model space as well as the number of parameters grows, meaning that we have added more heuristics in order to render learning of the translation model feasible. For SCFG-based translation models, non-parametric Bayesian methods have been proposed to learn synchronous rules without imposing heuristic constraints or relying on a previously generated word alignment (e. g. Levenberg et al., 2012). It would be interesting to work on learning SLCFRS rules in a similar fashion.

Galley and Manning (2010) address the problem of the exponential number of discontinuous phrase pair by using a suffix array technique which had been developed for tree-based models (Lopez, 2007). The training corpus is represented in a specific data structure, and only translation rules which apply to a test sentence at hand are extracted

on the fly. A related strategy could possibly be used to represent the training data for SLCFRS translation rules.

The implementation of the LCFRS parser of our decoder is specific to grammars of source fan-out 2. This makes sense given that our extracted grammars are heuristically constrained to fan-out $4_{2|2}$, which we empirically motivated. However, when using more general methods for grammar extraction which do not constrain the fan-out, grammars of higher source fan-out might be learned. This would require a more general parser implementation.

Finally, our approach to language model scoring of discontinuous target sequences might put discontinuous sequences at a disadvantage in comparison to continuous sequences.[11] Let us consider *not ... anymore* and *no longer* as competing translation options for, e. g., *nicht mehr*. The continuous sequence *no longer* will obtain the more precise language model score as *longer* can be attributed a bigram probability, while both *not* and *anymore* still lack their left context and therefore can only be scored with their unreliable unigram probability. Only when *not ... anymore* is part a larger segment, its language model score will be more precise. The risk is that it might have been already pruned by then. A remedy would be to develop a language model which scores dependent words even though they do not form a continuous block, for example along the lines of Sennrich (2015).

11 The same issue has been noted for the MBOT-based translation approaches (Braune, 2015, sections 5.3.3 and 5.6).

6 EVALUATION

In this chapter, the machine translation system presented in chapter 5, which implements the translation modeling approach proposed in chapter 4, is evaluated. To that end, experiments for German-to-English translation are conducted. The results are discussed and put into context.

The experimental setup and the main results of section 6.2 have been previously published in

> Kaeshammer, M. (2015). Hierarchical machine translation with discontinuous phrases. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 228–238, Lisbon, Portugal. Association for Computational Linguistics.

6.1 SETUP

We run experiments for German-to-English, based on data that has been used in the WMT 2014 translation task.[1] For training of the translation models, we use the parallel sentences from Europarl and the News Commentary Corpus up to a length of 30 words (1.3M sentence pairs). For the estimation of the language model, we use the *KenLM Language Model Toolkit*.[2] We train a 3-gram language model on all available monolingual English data (Europarl, News Commentary, News Crawl, 92.7M sentences). From the available development data, we use `newstest2013` as the development test set (max. 25 words). From the rest, we randomly select 3000 sentence pairs of a maximal length of 25 words as development set. We further refine this set to sentences without out-of-vocabulary source words by decoding the development set once and selecting the corresponding sentences. We thus end up with

1 Web page of the shared task: `http://www.statmt.org/wmt14/translation-task.html`. Data downloaded on September 08, 2014.

2 *KenLM* is documented at `http://kheafield.com/code/kenlm/`. `kenlm.tar.gz` was downloaded in April 2014.

1694 sentence pairs for tuning. As our test set, we use the cleaned test set that has been made available (2280 sentence pairs with a maximal length of 30 words).

We normalize the punctuation, tokenize and truecase all our data using the scripts that are available in *Moses*.[3] Furthermore, we perform compound splitting for German, also with the script provided in *Moses*.

The parallel training data is word-aligned by running multi-threaded *GIZA++* in both directions and then symmetrizing the alignments using the `grow-diag-final-and` heuristics as implemented in the *Moses* training script `train-model.perl` (step 1–4). Lexical translation probabilities are also emitted as part of this pipeline.

For the extraction of the Synchronous Linear Context-Free Rewriting System (SLCFRS) and the estimation of the rule translation probabilities, we use our training procedure *slcfrs-extract*. See section 5.2.2. For training a *Moses* hierarchical phrase-based system, we use the `-hierarchical` option in the training script. Generally, for grammar extraction, we limit the length of initial phrases (cf. section 5.2.1) and the number of words in a gap to 10. We neither allow unaligned words at edges of initial phrases nor unaligned blocks.

Before decoding a data set with our decoder *discodec*, we filter the large translation grammar with respect to the input data by extracting per-sentence grammars. These only contain rules whose terminals match the words in the sentence to translate.

Tuning the feature weights is done with minimum error rate training (Och, 2003), maximizing BLEU (Papineni et al., 2002) and using the 200 best translations. For our own decoder *discodec*, we use the very flexible implementation *Z-MERT*[4] (Zaidan, 2009). For *Moses*, we use its MERT implementation via the provided tuning script `mert-moses.pl`.

For the reported results, we set the buffer size for cube pruning to 400. We do not limit the number of words a non-terminal can span. We neither restrict the number of incoming hyperedges for the parse items nor the number of different target projections for the same source projection.

---

3 *Moses* is documented at `http://www.statmt.org/moses/`. The scripts are available under `https://github.com/moses-smt/mosesdecoder/tree/master/scripts`. Git commit 6085a60.

4 *Z-MERT* v1.50 obtained from `http://www.cs.jhu.edu/~ozaidan/zmert/`.

6.2 TRANSLATION QUALITY RESULTS

We compare different versions of our SLCFRS system *discodec* against each other. The baseline is a system which uses only Synchronous Context-Free Grammar (SCFG) rules, i. e. a hierarchical phrase-based system comparable to the one in Chiang (2007). We refer to it as DISCODEC(1,1), as it uses an SLCFRS of fan-out $2_{1|1}$.

DISCODEC(1,2) is a system which uses a grammar of fan-out $3_{1|2}$, i. e. it builds only continuous constituents on the source side, but allows for discontinuous constituents with two blocks on the target side. DISCODEC(2,1) is the analogous system which restricts the target side to continuous constituents. Finally, DISCODEC(2,2) uses an SLCFRS of fan-out $4_{2|2}$. In addition, we also provide results for systems which make use of our newly devised gap degree features g_s and g_t (cf. section 5.1.2).

6.2.1 *Automatic Evaluation*

Table 9 displays the main results. To account for the variance that is introduced by tuning, we repeat each experiment four times and report the mean of BLEU (Papineni et al., 2002) and METEOR (Denkowski and Lavie, 2011).[5] The calculation of the scores as well as significance testing is performed with *MultEval* (Clark et al., 2011).[6]

Allowing gaps on both the source and the target side leads to a decline in BLEU and METEOR compared to the baseline on both data sets; see DISCODEC(1,1) VS. DISCODEC(2,2). We hypothesize that this is due to weak probability estimates of the discontinuous rules because of data sparseness, and the additional ambiguity that is incurred by the new rules with discontinuities.

By adding the gap degree features g_s and g_t, the model has an additional way of influencing which kind of rules are used. The scores of DISCODEC(2,2) then approach and even surpass the scores of the baseline on both data sets. Especially controlling for the target gap degree turns out to be important. See the results for DISCODEC(2,2)-g_t and DISCODEC(2,2)-g_s-g_t, which show small, but consistent improvements, some of them even significant. It is worth noting that rules with target

5 See p. 68 for more information about the metrics.
6 `https://github.com/jhclark/multeval`, Git commit bd93ed1.

		devtest		test	
system	feat.	BLEU	METEOR	BLEU	METEOR
DISCODEC(1,1)		24.13	30.81	23.23	30.01
DISCODEC(2,2)		23.90	30.75	22.90	29.91
DISCODEC(2,2)	g_s	24.06	30.81	23.17	30.02
DISCODEC(2,2)	g_t	**24.20**	30.85	* 23.35	** 30.07
DISCODEC(2,2)	g_s, g_t	24.18	* **30.88**	23.32	** **30.10**
DISCODEC(1,2)		23.40	30.57	23.19	30.00
DISCODEC(2,1)		24.17	* 30.86	** **23.41**	** 30.08
MOSES		24.33	30.87	23.34	30.13

Table 9: Results for German-to-English translation: averaged BLEU and METEOR scores over four tuning runs. The feat column indicates whether additional source/target gap degree features have been used. The best-performing DISCODEC system in each column is marked in bold. Starred results indicate statistically significant improvements over the DISCODEC(1,1) baseline, at confidence $p < 0.08$ (*) and $p < 0.01$ (**).

gaps are not completely dismissed when the target gap degree feature is switched on. Usage of discontinuous rules with a target gap goes down from on average 735 rules in DISCODEC(2,2) to on average 77 rules in DISCODEC(2,2)-g_t in the test set. They are used less often, but, it seems, in a more controlled and sensible way. Section 6.3.2 will provide more details.

The importance of limiting the generation of gaps on the target side is further confirmed with the experiments in which the discontinuous rules are only used on one side. Restricting the target side derivations to continuous yields leads to small improvements in BLEU and METEOR scores. As measured by METEOR the improvement is significant on both data sets; in terms of BLEU, DISCODEC(2,1) is the best system for the test set. This is in particular interesting with respect to translation times since restricting the target side to continuous yields means removing the additional complexity that target gaps elicit for the language model integration (see section 5.3.2).

Restricting the source side derivations to continuous yields does not improve the BLEU and METEOR scores; it rather severely degrades them, see DISCODEC(1,2).

To put our results into context, we also report results for the hierarchical phrase-based system in *Moses* trained on the same data as the DISCODEC systems. We tried to use as much as possible the same settings as for our comparable system DISCODEC(1,1). However, given the number of parameters during training and decoding, the various interpretations thereof and numerous implementation details to consider, it is not too surprising that the MOSES system actually produces different translations than ours. The reported numbers merely serve as a point of reference, indicating that the translations produced by our system are not totally far off.

6.2.2 *Human Evaluation*

In addition to the automatic evaluation, we conducted a manual evaluation in form of a system comparison using our own installation of the *Appraise* tool (Federmann, 2012). We compare the baseline DISCODEC(1,1) against DISCODEC(2,1), one of the best-performing setups on the test set. For each of the two system types, we randomly selected one of the four optimized systems. We first discarded all test sentences for which the two systems provide the same translation (1323 out of 2280). From the remaining test sentences, we then selected those as our test set for manual evaluation where DISCODEC(2,1) uses at least one SLCFRS rule with a discontinuity (95 sentences). We recased and detokenized the translations before showing them to the evaluators, using the scripts available in *Moses*.

We asked two native speakers of English (e1, e2) with basic knowledge of German to evaluate our test sentences. They were shown the source sentence, a reference translation, the DISCODEC(1,1) translation and the DISCODEC(2,1) translation. The latter two were presented anonymized and in random order. The options for the evaluators were

a) translation A is better than B,

b) translation B is better than A, and

c) translations A and B are of equal quality.

	DISCODEC(1,1)	DISCODEC(2,1)	=
e1	43	49	3
e2	46	47	2

Table 10: Result of the manual system comparison

		e2		
		DISCODEC(1,1)	DISCODEC(2,1)	=
e1	DISCODEC(1,1)	**29**	13	1
	DISCODEC(2,1)	15	**33**	1
	=	2	1	0

Table 11: Confusion matrix of the decisions of the manual evaluation

We specifically asked them to use option c) as rarely as possible.

Table 10 shows the results. Evaluator e1 preferred translations generated by DISCODEC(2,1) 49 times, and evaluator e2 47 times out of 95 test sentences. While they do not demonstrate a clear preference for one of the systems, there is, however, a slight preference for the system that uses discontinuous rules, i. e. DISCODEC(2,1). In spite of the inter-annotator agreement being not very high (Cohen's $\kappa = 0.338$), the tendency for DISCODEC(2,1) is also perceivable for the translations for which the evaluators agree in their decisions, see table 11.

6.3 FURTHER ANALYSIS

6.3.1 *Translation Examples*

This section provides some actual translation examples from our test set. They have been selected because they make crucial use of the discontinuous SLCFRS rules and demonstrate how complex alignment configurations are generated. The presented translations and derivations have been produced by the systems used in the human evaluation in section 6.2.2 if not otherwise indicated. For the sake of clarity and presentation ease, "uninteresting" parts of the sentences are left out.

Source:	er wäre damit auch geeignet gewesen , he would have thereby also suitable been , um die illegale Einwanderung Richtung USA PARTICLE the illegal immigration direction/to USA zu fördern . to promote .
Reference:	it would thus be suitable to assist illegal immigration into the USA .
DISCODEC(1,1):	it would also have to be , in order to promote *the direction US illegal immigration* .
DISCODEC(2,1):	he also would have been appropriate to promote *the direction US illegal immigration* .
DISCODEC(2,2):	he would also be appropriate to encourage *illegal immigration in the US* .

Figure 59: Test sentence with translations provided by the SCFG and SLCFRS systems

Let us start with the example in figure 59. If we ignore the noun phrase (in italics), which is problematic for all three translation systems, the translation generated by DISCODEC(2,1) is meaningful, in particular it has an overall grammatical sentence structure. This is in clear contrast to the DISCODEC(1,1) translation which is not grammatical and it misses important concepts, such as *geeignet* (*suitable*). The DISCODEC(2,2) translation is also acceptable, although, in contrast to the DISCODEC(2,1) translation in perfect conditional tense, it does not convey the exact verbal tense of the source sentence, though neither does the reference translation. The missing context precludes an assessment of the correct gender of the subject pronoun (*it* or *he*).

The derivation generated by DISCODEC(2,1) is depicted in figure 60. It has a source gap degree of 1. The two SLCFRS rules which feature discontinuous constituents are the following:

1. $\langle X(\text{wäre , } Y_1 \text{ gewesen } Y_2) \rightarrow X_{\boxed{1}}(Y_1)X_{\boxed{2}}(Y_2),$
   $X(\text{would have been } Y_1 Y_2) \rightarrow X_{\boxed{1}}(Y_1)X_{\boxed{2}}(Y_2)\rangle$
2. $\langle X(Y_1 \text{ damit auch } Y_2 \text{ .}) \rightarrow X_{\boxed{1}}(Y_1, Y_2),\ X(\text{also } Y_1 \text{ .}) \rightarrow X_{\boxed{1}}(Y_1)\rangle$

Rule #1 has a fan-out of 2 on the source side. It derives the synchronous constituent labeled $X_{\boxed{6}}$ with source gap degree 1. Besides

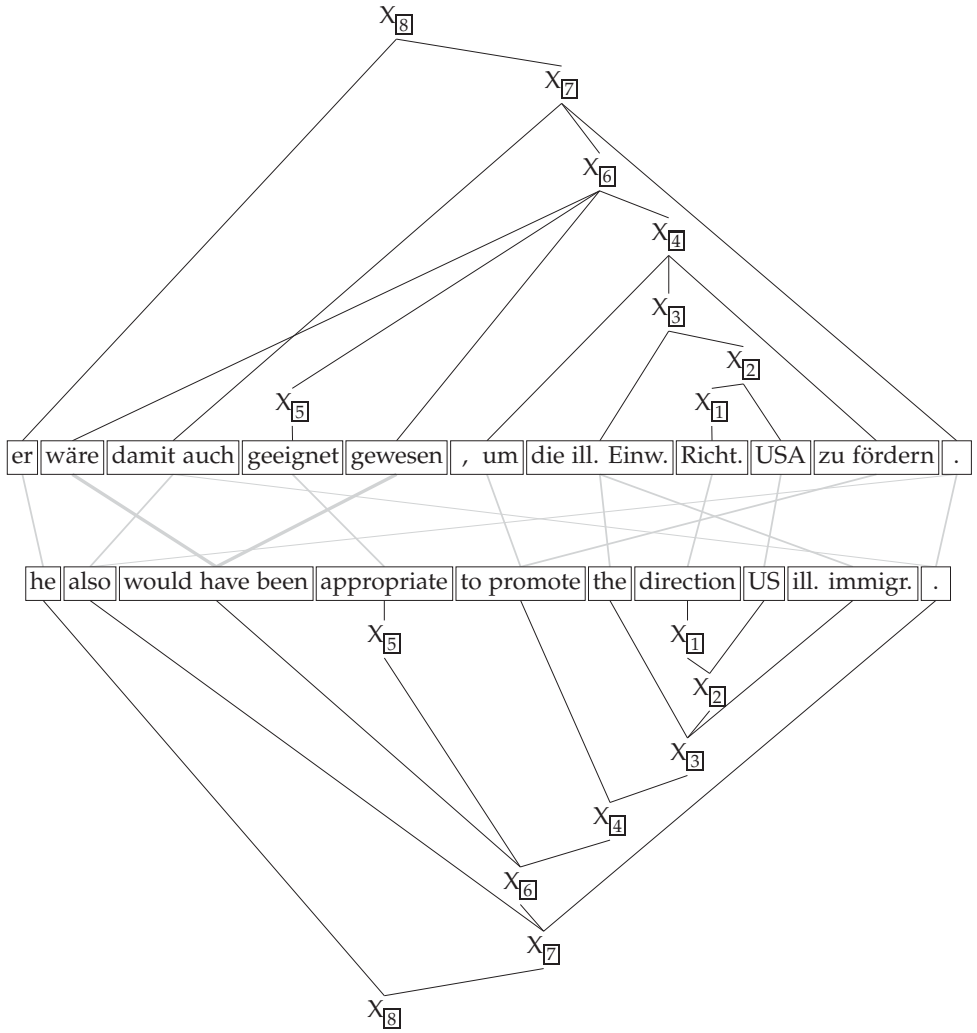


Figure 60: Derivation of the translation in figure 59 of the DISCODEC(2,1) system

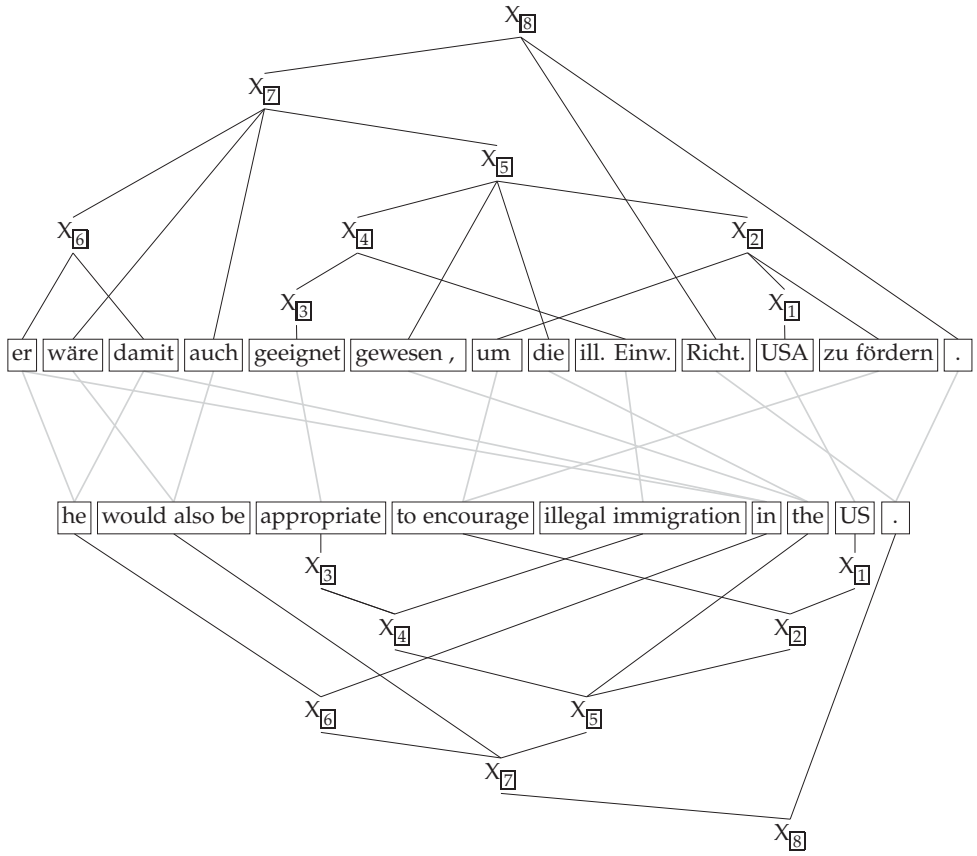


Figure 61: Derivation of the translation in figure 59 of the DISCODEC(2,2) system

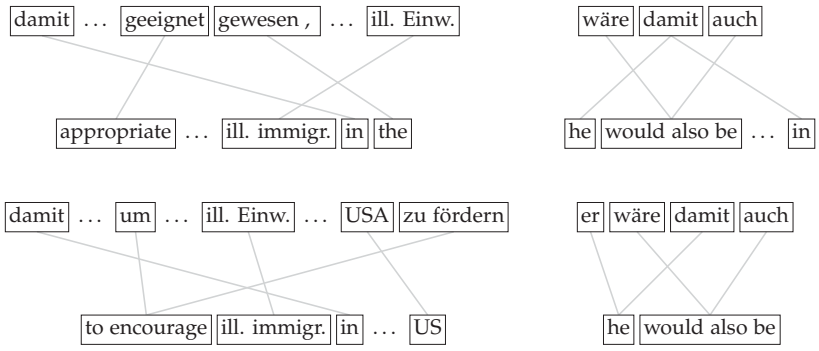


Figure 62: Four complex alignment configurations from the alignment in figure 61: an IO alignment, a bonbon configuration, an IO-DTU and a CDTU.

providing a correct verbal translation in a specific tense, it also establishes a relationship to the adjective ($X_{\boxed{5}}$) and the infinitive subordinate clause ($X_{\boxed{4}}$), thereby still leaving room for the adverb in terms of the gap on the source side. The adverb is then introduced with rule #2, leading to the constituent labeled $X_{\boxed{7}}$. This rule can be seen as capturing the different placement of the adverb *auch/also* in German and English.

Note that the alignment that is induced by the DISCODEC(2,1) derivation is also derivable with a $2_{1|1}$-SLCFRS, by putting the individual translation units together in a different order and hierarchy. For example, in an SCFG rule, the discontinuous verb phrase (bold alignment links) could be combined with the adjective and the adverb first, which leads to a continuous constituent. Then the subordinate clause would be added in a later derivation step. However, in the derivation for the best translation of DISCODEC(1,1), this does not happen because a corresponding specific rule has not been learned.

Figure 61 shows the derivation of the translation by DISCODEC(2,2). Both its source and its target gap degree is one. The source side tree features five discontinuous constituents ($X_{\boxed{2}}$, $X_{\boxed{4}}$, $X_{\boxed{5}}$, $X_{\boxed{6}}$, $X_{\boxed{7}}$), while the target side tree features four discontinuous constituents ($X_{\boxed{2}}$, $X_{\boxed{4}}$, $X_{\boxed{5}}$, $X_{\boxed{6}}$). The alignment generated by this derivation contains many complex alignment configurations. As they are difficult to spot, some of them are depicted individually in figure 62.

Even though the translation quality that this particular derivation provides is still somewhat acceptable, it demonstrates that the addi-

tional power of the discontinuous translation rules *can* lead to unexpected and unreasonable alignments which in turn *can* lead to a decrease in translation quality (cf. the results for DISCODEC(2,2) in table 9). In the example at hand this is particularly the case when there is no one-to-one correspondence between an expression in the source and in the target: the German determiner *die* of the noun *Einwanderung* is not being transferred to the target, which is correct, i. e. *immigration* does not have a determiner, however, $X_{\boxed{5}}$ instead generates the determiner of *US*. Similarly, rules #3 and #4 (below) lead to unexpected dependencies in the scope of $X_{\boxed{6}}$ and $X_{\boxed{8}}$ respectively and to alignments which one might indeed call "wrong" in the given context.

3. $\langle X(\text{er , damit}) \rightarrow \varepsilon,\ X(\text{he , in}) \rightarrow \varepsilon \rangle$

4. $\langle X(Y_1 \text{ Richtung } Y_2 \text{ .}) \rightarrow X_{\boxed{1}}(Y_1, Y_2),\ X(Y_1 \text{ .}) \rightarrow X_{\boxed{1}}(Y_1) \rangle$

Such rules probably originate from faulty word alignments in the first place, but this is an issue common to all evaluated systems. Even if a sensible use of those rules is difficult to imagine, one should not generally judge them as wrong or harmful. This decision should rather be delegated to a (good) model and its features and weights. Finally, alignments and derivations are latent in the translation process anyway. There is thus no right or wrong, as long as the surfacing translations and their quality is being optimized.

Next, let us consider the translation in figure 63. The DISCODEC(2,1) translation features a bonbon configuration using the following two rules:

5. $\langle X(\text{Ihnen , zustimmen }) \rightarrow \varepsilon,\ X(\text{agree with you}) \rightarrow \varepsilon \rangle$

6. $\langle X(Y_1 \text{ nicht stärker } Y_2) \rightarrow X_{\boxed{1}}(Y_1, Y_2),\ X(\text{not } Y_1 \text{ more}) \rightarrow X_{\boxed{1}}(Y_1) \rangle$

The object of *zustimmen/agree* is realized as a dative object in German and as a prepositional phrase in English, as captured in rule #5, leading to the synchronous constituent labeled $X_{\boxed{4}}$ of fan-out $v = 3_{2|1}$. The verbal translation unit that it generates is intertwined with the adverbial translation unit as described by rule #6, inducing a bonbon configuration.

This translation by DISCODEC(2,1) is a perfect translation, and differs from the reference translation only by using the full form of *not* instead of the contraction. In contrast to the previous example in figure 60, the alignment as shown in figure 63 cannot be derived by a

Source:	...	und	ich	könnte	Ihnen	nicht	stärker	zustimmen	.
	...	and	I	could	you	not	stronger	agree	.

Reference: ... and I couldn't agree with you more .
DISCODEC(1,1): ... and I could you not agree more .
DISCODEC(2,1): ... and I could not agree with you more .

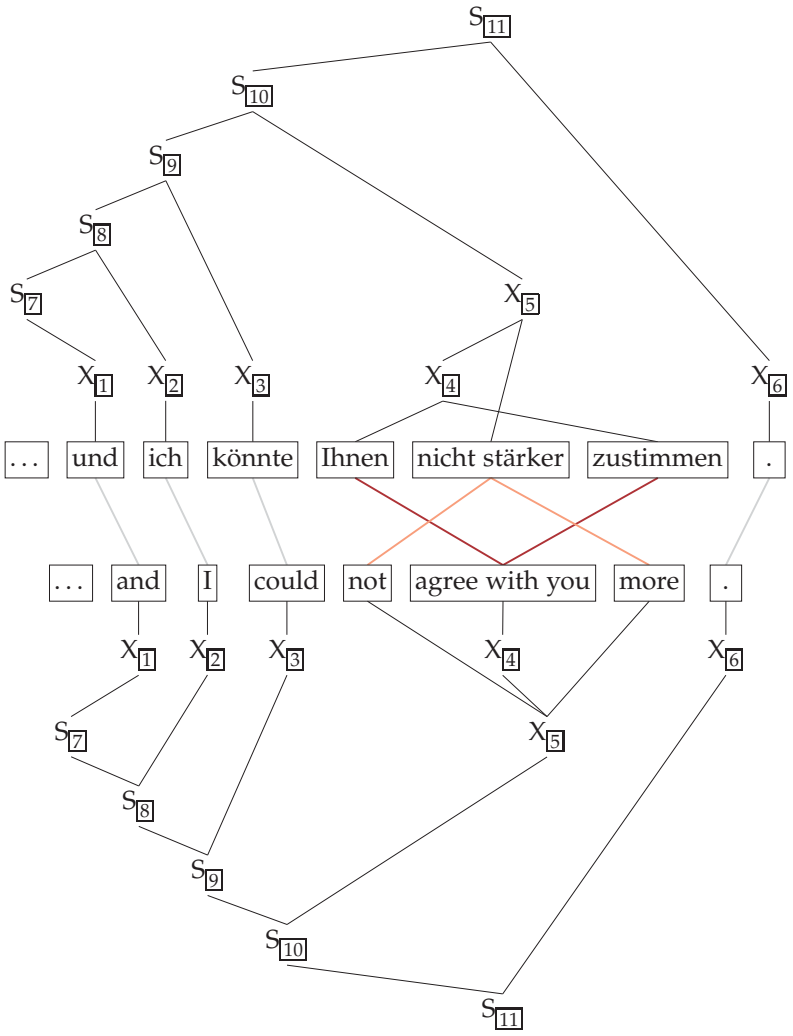


Figure 63: Test sentence with translations provided by the SCFG and the SLCFRS systems, including the derivation of the SLCFRS system DISCODEC(2,1). The bonbon configuration is highlighted.

$2_{1|1}$-SLCFRS. The bonbon configuration is beyond the alignment capacity of SCFG as it requires discontinuous constituents.

The DISCODEC(1,1) system, thus, fails to realize the object adequately and comes up with a translation which is too close to the German source. We would like to point out however that, in theory, an SCFG-based system could produce the same translation as DISCODEC(2,1) on the surface, though with a different derivation and alignment, by splitting either the verbal or the adverbial translation unit into two. Sample rules in which the object is derived independently from the verb are shown in the following:

7. $\langle \mathrm{X}(\text{Ihnen}) \rightarrow \varepsilon,\ \mathrm{X}(\text{with you}) \rightarrow \varepsilon \rangle$

8. $\langle \mathrm{X}(\text{zustimmen}) \rightarrow \varepsilon,\ \mathrm{X}(\text{agree}) \rightarrow \varepsilon \rangle$

9. $\langle \mathrm{X}(Y_1 \text{ nicht stärker } Y_2) \rightarrow \mathrm{X}_{\boxed{1}}(Y_1)\,\mathrm{X}_{\boxed{2}}(Y_2),$
$\mathrm{X}(\text{not } Y_1 Y_2 \text{ more}) \rightarrow \mathrm{X}_{\boxed{1}}(Y_2)\,\mathrm{X}_{\boxed{2}}(Y_1) \rangle$

We can only speculate why neither DISCODEC(1,1) nor DISCODEC(2,1) produced the best translation choosing those rules, as the log-linear model incorporates a plenitude of factors. Potential reasons include the larger number of rules, comparably low translation probabilities for *Ihnen*/*with you*, little evidence for applying rule #9 as the reordering is not lexically motivated, or simply a search error.

In any case, DISCODEC(2,1) has the advantage of being able to represent the source-discontinuous translation unit ⟨ *Ihnen*, *zustimmen* ; *agree with you* ⟩ as a lexical rule (#5) and the target-discontinuous translation unit ⟨ *nicht stärker* ; *not*, *more* ⟩ as a mixed rule (#6), thus directly encoding the necessary lexical dependency and word order information to produce a fluent and adequate translation.

Next, we consider the test sentence and its translations in figure 64. In this case, the DISCODEC(1,1) translation is acceptable, but the DISCODEC(2,1) translation, which features a CDTU, represents a nice variation. Both use the *to be to do something* construction to express that something shall be done. However, while the DISCODEC(1,1) translation rather literally translates the passive construction used in the German source, DISCODEC(2,1) uses active voice. Both evaluators of the human evaluation presented in section 6.2.2 preferred the DISCODEC(2,1) translation over the DISCODEC(1,1) translation. The derivation of the first is shown in figure 65.

Source:	... Ausarbeitung von Strategien , mit denen die ... development of strategies , with which the ambitionierten , gesetzlich verankerten Ziele des ambitious , legally anchored goals of the Bundesstaats zum Klimawandel erreicht werden federal state on climate change reached PASSIVE sollen . shall .
Reference:	... they devise strategies to meet the goals laid out in the state 's ambitious global warming laws .
DISCODEC(1,1):	... drafting of policies with which the ambitious , legally enshrined targets of the federal state on climate change are to be achieved .
DISCODEC(2,1):	... development of strategies , which are to achieve the ambitious objectives, enshrined by law of the federal state on climate change .

Figure 64: Test sentence with translations provided by the SCFG and the SLCFRS systems

The CDTU consists of two translation units which are both discontinuous on the source side. One, let us call it the *relative* one for the purpose of easier reference since it captures the relative pronoun, is induced by rule 10, which creates the synchronous constituent labeled $X_{\boxed{7}}$ in figure 65. The synchronous constituent labeled $X_{\boxed{8}}$ is generated by rule #11. It corresponds to the second translation unit, the *verbal* one.

10. $\langle X(\text{mit denen , werden}) \rightarrow \varepsilon,\ X(\text{which are}) \rightarrow \varepsilon \rangle$

11. $\langle X(Y_1 \text{ die } Y_2 \text{ erreicht } Y_3 \text{ sollen}) \rightarrow X_{\boxed{1}}(Y_1, Y_3)\, X_{\boxed{2}}(Y_2),$
$X(Y_1 \text{ to achieve the } Y_2) \rightarrow X_{\boxed{1}}(Y_1)\, X_{\boxed{2}}(Y_2) \rangle$

While in the previous examples of DISCODEC(2,1) derivations, the gaps of discontinuous constituents were rather short and filled with material from just one translation unit, figure 64/65 shows that long gaps can also be useful to generate acceptable translations. The gap between *mit denen* and *werden* is filled with 11 words which are part of 7 translation units.

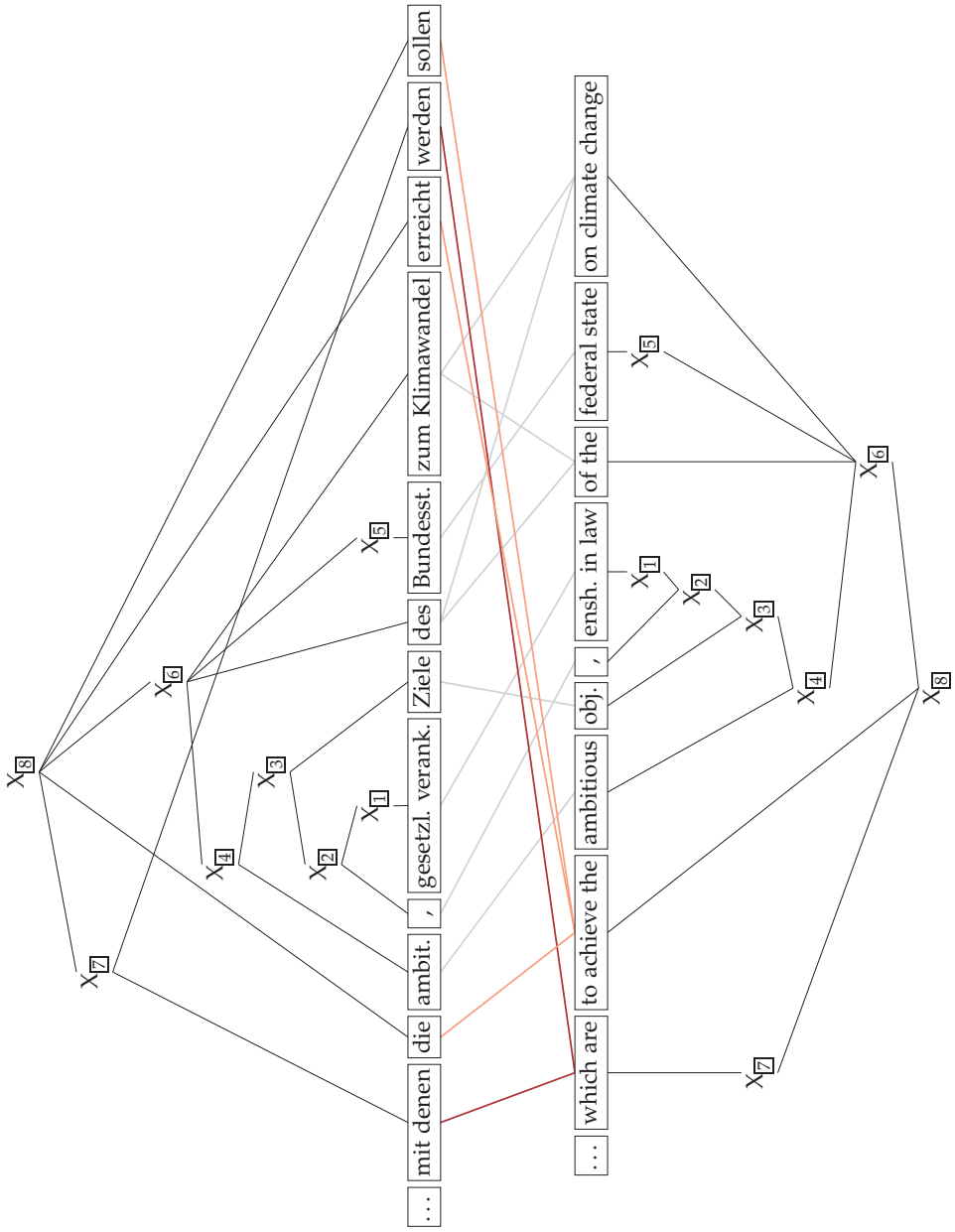


Figure 65: Derivation of the translation in figure 64 of the DISCODEC(2,1) system. The CDTU is highlighted.

This alignment is again beyond SCFG alignments, and no reordering on the target side would be possible to simplify the alignment.

Finally, let us consider the example in figure 66. The SLCFRS translation is by a DISCODEC(2,2)-g_s-g_t system. Except the missing apostrophe (*Gutachs* is an unknown word to the system), the translation is very well acceptable. The DISCODEC(1,1) translation is understandable, despite the subject being plural (*mayors*) and the adverbial placement being somewhat clumsy.

The DISCODEC(2,2)-g_s-g_t derivation does not only feature a CDTU and a gap degree of 1 on the source side, the gap also has a depth of 2 (cf. def. 2.15), i. e. it is not filled by the mother of the discontinuous node, but by its grandmother. The following three rules derive this structure:

12. $\langle \mathrm{X}(\text{diese Frage , klar beantwortet}) \to \varepsilon,$
$\mathrm{X}(\text{answered that question very clearly}) \to \varepsilon \rangle$

13. $\langle \mathrm{X}(Y_1 \text{ hat } Y_2, Y_3) \to \mathrm{X}_{\boxed{1}}(Y_1, Y_3)\, \mathrm{X}_{\boxed{2}}(Y_2),$
$\mathrm{X}(Y_1 \text{ has } Y_2) \to \mathrm{X}_{\boxed{1}}(Y_2)\, \mathrm{X}_{\boxed{2}}(Y_1) \rangle$

14. $\langle \mathrm{X}(Y_1 \text{ gestern } Y_2 \text{ .}) \to \mathrm{X}_{\boxed{1}}(Y_1, Y_2),\ \mathrm{X}(Y_1 \text{ yesterday . }) \to \mathrm{X}_{\boxed{1}}(Y_1) \rangle$

Rule #12 derives the synchronous constituent $\mathrm{X}_{\boxed{1}}$ which has a source fan-out of 2. This lexical rule links the verb *beantwortet*/*answered* with its direct object, which in the German sentence occupies the *initial field* position (cf. p. 118). The gap in-between is partially filled with the auxiliary verb and the subject using rule #13. Note how the same rule is also responsible for the reordering of the subject and the object on the English side.

The source left-hand side (LHS) non-terminal of rule #13 also has a fan-out of 2, leading to $\mathrm{X}_{\boxed{4}}$ which is still discontinuous on the source side, in order to accomodate for the adverb *gestern*. $\mathrm{X}_{\boxed{5}}$, created from rule #14, closes the gap.

This example is also a showcase of discontinuities allowing to capture larger phrases, which in turn lead to more fluent and well-formed translations: The translation unit represented by rule #12 needs to be divided into three separate translation units if discontinuities are not allowed, e. g. in an SCFG-based system: ⟨ *diese Frage* ; *that question* ⟩, ⟨ *klar* ; *(very) clearly* ⟩ and ⟨ *beantwortet* ; *answered* ⟩. While this is

Source:	diese Frage hat Gutachs Bürgermeister gestern this question has Gutachs' mayor yesterday klar beantwortet . clearly answered .
Reference:	yesterday, Gutacht*[sic]* 's Mayor gave a clear answer to this question .
DISCODEC(1,1):	Gutachs mayors yesterday clearly has answered that question .
DISCODEC(2,2)-g_s-g_t:	Gutachs mayor has answered that question very clearly yesterday .

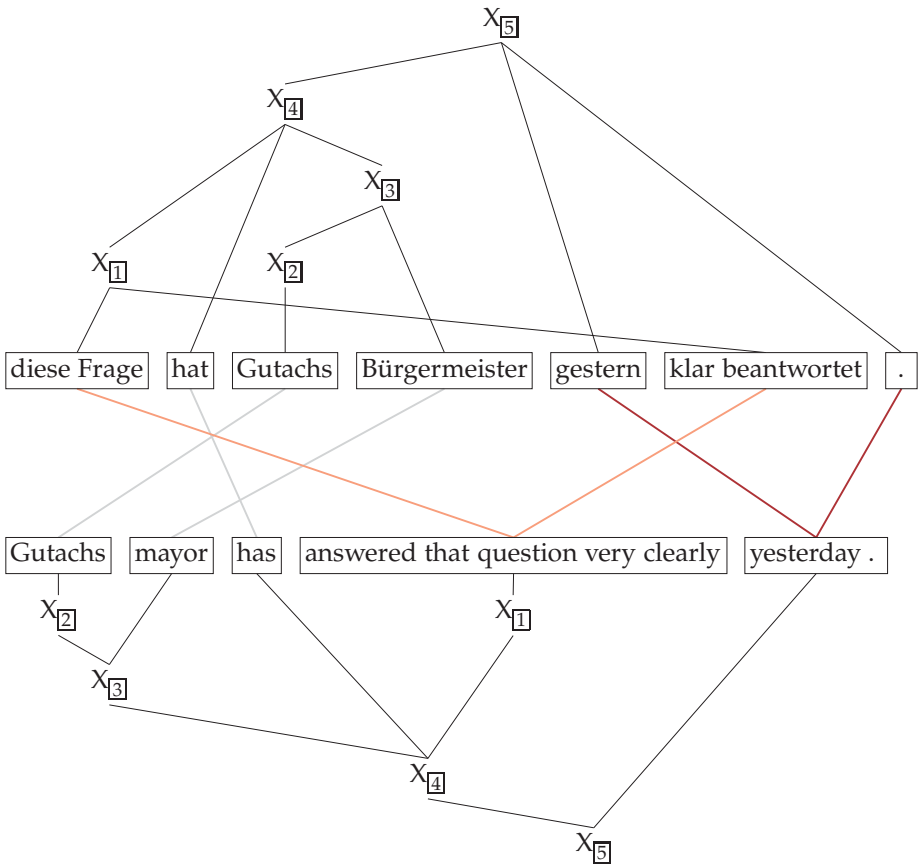


Figure 66: Test sentence with translations provided by the SCFG and the SLCFRS systems, including the derivation of the SLCFRS system DISCODEC(2,2)-g_s-g_t. The CDTU is highlighted.

not problematic in itself, more things can go wrong when reassembling several small translation units than during the reordering of a few large translation units. For instance, the DISCODEC(1,1) translation does not correctly realize the subject-verb agreement and fails at the placement of the sentential and verbal adverbs.

6.3.2 *Discontinuous Constituents and Complex Alignment Configurations*

This section will provide insights into how the discontinuous rules of the SLCFRS translation grammars are used across the conducted experiments. Table 12 shows aggregated numbers for discontinuous constituents and complex alignment configurations occurring in the test set.

Let us first concentrate on the number of *discontinuous constituents* present in the derivation of the best translation of the test sentences. While, without the gap degree features, DISCODEC(2,2) produces translations with discontinuous constituents in 18.68% of the test sentences, the number of discontinuities drastically reduces when adding one or both gap degree features or when constraining the source or the target side of the derivation to continuous constituents. Remarkably, for all systems which outperform the baseline, the average numbers of sentences which feature discontinuities are very close to each other (5.99%, 6.35% and 6.47%). This might be a characteristic of the tested language pair, or more specifically of the test data set.

The absolute value of the weights of the gap degree features tend to be rather high, see table 13. Intuitively, this means that hypotheses are usually penalized quite heavily for using discontinuous rules. Table 12 clearly shows that they are nevertheless used in top-scoring hypotheses and contribute to improving translation quality over the baseline.

The correlation coefficients r in table 13 furthermore show that the features g_s and g_t and their weights rather directly impact the number of generated discontinuous constituents.[7] This was the intention when designing the gap degree features. Thus, they work as expected.

Section 6.2.1 already provided evidence that constraining the gaps on the target side, either by excluding them formally or by penalizing

7 They should however be treated with caution, as the correlation could also be by chance, given the small number of data points.

system	feat.	BLEU	discontinuous constituents			complex alignment configurations					
			source disc.	target disc.	% of sent.	IO	CDTU	Bonbon	IO-DTU	% of sent.	% of TUs
DISCODEC(1,1)		23.23	0	0	0.00	0	0	0	0	0.00	0.00
DISCODEC(2,2)		22.90	761	735	18.68	2993	1276	388	3601	9.71	3.99
DISCODEC(2,2)	g_s	23.17	21	54	2.12	238	54	22	200	0.87	0.41
DISCODEC(2,2)	g_t	23.35	177	77	6.47	1870	196	56	936	3.13	1.27
DISCODEC(2,2)	g_s, g_t	23.32	207	45	6.35	2244	238	62	1038	3.59	1.45
DISCODEC(1,2)		23.24	0	29	1.21	71	7	3	10	0.48	0.22
DISCODEC(2,1)		23.41	160	0	5.99	1123	104	24	468	2.84	1.19

Table 12: Analysis of the translations in the test set: number of discontinuous constituents in the derivations of the best translations, and ratio of sentences which feature at least one discontinuous constituent; number of different types of complex alignment configurations, ratio of sentences which feature at least one complex alignment configuration, and ratio of translation units (TUs) which are part of a complex alignment configuration. All reported numbers are averages over the systems obtained by multiple tunings.

		experiments				
system	feat.	*1*	*2*	*3*	*4*	*r*
DISCODEC(2,2)	g_s	-0.5377	-0.3237	-0.4112	-0.4790	0.88
DISCODEC(2,2)	g_t	-0.4227	-0.2296	-0.6978	-0.4148	0.80
DISCODEC(2,2)	g_s	-0.0041	-0.1589	0.1490	-0.3068	0.83
	g_t	-0.4327	-0.2181	-0.3797	0.03673	0.79

Table 13: Weights of the indicated features which have been obtained with MERT, and their correlation with the number of discontinuous source respectively target constituents, whose means are given in table 12.

them via a feature, is crucial for building an SLCFRS-based translation system which outperforms the SCFG-based baseline. In addition, table 12 teaches us about the role of the source discontinuities. Disallowing source discontinuities (DISCODEC(1,2)) leads to a system of similar or worse performance as the baseline. To outperform the baseline, a certain amount of source discontinuities seem to be necessary, as seen with DISCODEC(2,1), DISCODEC(2,2)-g_t and DISCODEC(2,2)-g_s-g_t.

Many SLCFRS translation rules extracted by *slcfrs-extract* feature both, source and target fan-out larger than 1. For instance, in the translations of the test set by DISCODEC(2,2), only about 6% of the discontinuous rules have a LHS non-terminal fan-out of $v = 3_{2|1}$ and 3% of $v = 3_{1|2}$. The remaining discontinuous rules generate a gap on the source and the target side simultaneously. This means that there is a dependency between source and target discontinuities in the translation rules. Therefore, controlling the target gaps (via $v_t = 1$ or via g_t) simultaneously influences the number of source discontinuities.

It remains unclear, however, why g_s alone is ineffective in generating a suitable amount of source discontinuities. This might as well be an effect of the dependency between source and target discontinuities: possibly, g_s is in the first place optimized to generate only a few target gaps, which then leads to an undergeneration of source gaps during decoding.

In addition, we also analysed the derivations with respect to generated *ill-nested* structures (cf. def. 2.14, p. 21). SLCFRS rules which derive ill-nested structures are not necessary for deriving any of the

complex alignment configurations, but of course they could be used nevertheless. The grammar extraction algorithm of *slcfrs-extract* does not make reference to ill-nestedness. It certainly does not prohibit the extraction of rules which generate ill-nested structures. However, the derivations of DISCODEC(1,2), DISCODEC(2,1) and DISCODEC(2,2)-g_t are all well-nested on the source and the target side. DISCODEC(2,2)-g_s-g_t and DISCODEC(2,2)-g_s generate one ill-nested structure, on the source and the target side respectively, across all experiments. Only derivations of DISCODEC(2,2) feature a non-negligible amount of ill-nestedness, showing that the decoder is in principle able to generate them: 4.75 ill-nested constructions on the source side in the test set (on average across the experiments), and 5.75 on the target side. The derivation in figure 61 contains several ill-nested configurations: on the source side, $X_{\boxed{2}}$ and $X_{\boxed{4}}$, on the target side, $X_{\boxed{2}}$ and $X_{\boxed{4}}$ as well as $X_{\boxed{5}}$ and $X_{\boxed{6}}$ are ill-nested.

Note that DISCODEC(2,2) is also the system generating the most discontinuous constituents. Since an ill-nested structure requires (at least) two discontinuous constituents, it is natural that this system also generates the most ill-nested structures. However, since DISCODEC(2,2) fares poorly in the evaluation of the translation quality, no conclusion can be drawn about the status of ill-nested structures and their importance for translation quality.

Table 12 also shows the number of *complex alignment configurations* in the test set as generated by DISCODEC. The numbers are obtained by generating a word alignment for each sentence pair from the synchronous SLCFRS derivation, and then analysing the alignments to identify complex configurations. As described in section 3.1.1, the terminals in each synchronous rule form a translation unit, and the words of one translation unit are all linked to each other. For the purpose of counting, we generate only one link for adjacent terminals in the rules, or, when thinking in terms of translation units, we align continuous blocks of translation units rather than words.

For illustration, consider the example in figure 67. The three rules used in the derivation are the following:

1. $\langle \mathrm{X}(\text{Forschung}) \to \varepsilon,\ \mathrm{X}(\text{research}) \to \varepsilon \rangle$
2. $\langle \mathrm{X}(\text{es , noch weitere } Y_1 \text{ notwendig}) \to \mathrm{X}_{\boxed{1}}(Y_1),$
   $\mathrm{X}(\text{more } Z_1, \text{ needed}) \to \mathrm{X}_{\boxed{1}}(Z_1) \rangle$

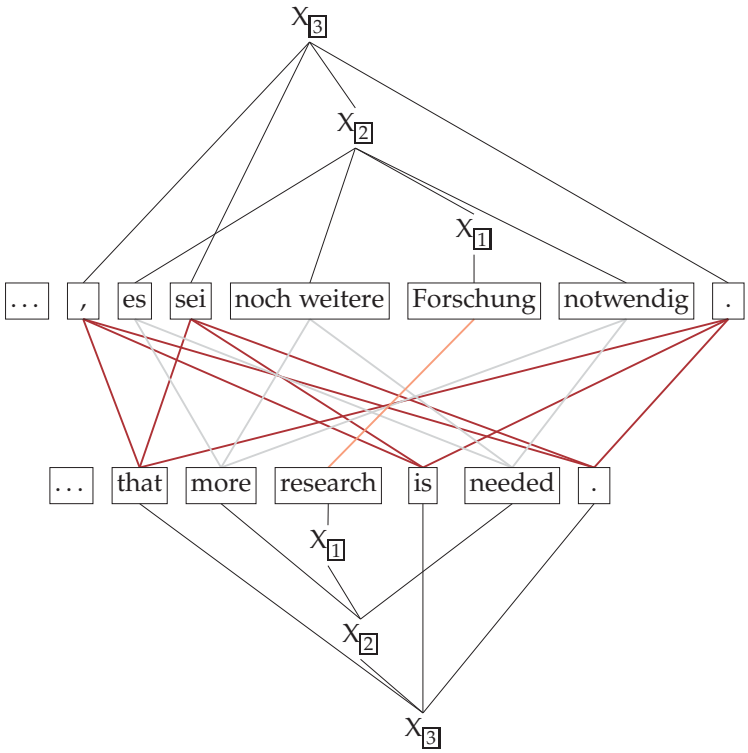


Figure 67: Partial derivation of a test sentence with complex alignment configurations by the SLCFRS system DISCODEC(2,2)-g_s-g_t

3. $\langle \mathrm{X}(\text{","}\ Y_1\ \text{sei}\ Y_2\ .) \rightarrow \mathrm{X}_{\boxed{1}}(Y_1, Y_2),\ \mathrm{X}(\text{that}\ Z_1\ \text{is}\ Z_2\ .) \rightarrow \mathrm{X}_{\boxed{1}}(Z_1, Z_2)\rangle$[8]

Rule #2 creates a synchronous constituent $\mathrm{X}_{\boxed{2}}$ which is discontinuous on the source and the target side. Each rule gives raise to a translation unit:

1. ⟨ Forschung ; research ⟩
2. ⟨ es , noch weitere , notwendig ; more , needed ⟩
3. ⟨ "," , sei , . ; that , is , . ⟩

The *holes* or *gaps* in the translation units correspond to non-terminals and argument boundaries in the SLCFRS rules.

The three translation units are intertwined as shown in the alignment in figure 67. Note how *noch weitere* is treated as one unit for alignment here, even though classically the two words would be aligned

8 To distinguish the terminal comma from the comma as the argument separator in this rule and in translation unit notation, we put the first between quotes.

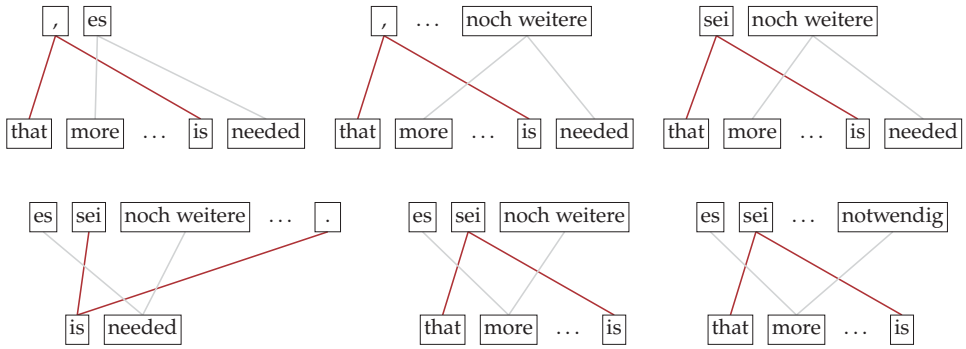


Figure 68: Four CDTUs and two bonbon configurations from the alignment in figure 67

separately. This would however artificially inflate the number of complex alignment configurations. The displayed alignment features 14 bonbon configurations and 42 CDTUs, all of them induced by just rules #2 and #3. Some of those complex configurations are depicted individually in figure 68.

Across the experiments, IO alignments and IO-DTUs are the most frequent complex alignment configurations, followed by CDTUs and bonbon configurations, see table 12. The average ratio of translation units which are part of some complex alignment configuration ranges between 0.22% for DISCODEC(1,2) and 3.99% for DISCODEC(2,2). Again, the systems which outperform the SCFG-based baseline are close to each other concerning this characteristics: 1.19% with DISCODEC(2,1), 1.27% with DISCODEC(2,2)-g_t and 1.45% with DISCODEC(2,2)-g_s-g_t.

As we can also see from table 12, across all experiments, the number of sentences whose alignments feature complex configurations is very roughly half the number of sentences with discontinuous constituents. For instance, with DISCODEC(2,2)-g_t, on average 6.47% of the translations in the test set feature at least one discontinuous constituent, and on average 3.13% of the translations incorporate at least one complex alignment configuration. While discontinuous constituents are necessary to generate complex alignment configurations, those numbers show that they are also used in the derivations of the test set without inducing any complex configuration.

Finally, we also look into the distribution of the generated gaps. Figure 69 shows how many gaps of a certain *size* (cf. def. 2.9, p. 20) are generated on the source side and on the target side for each system. In

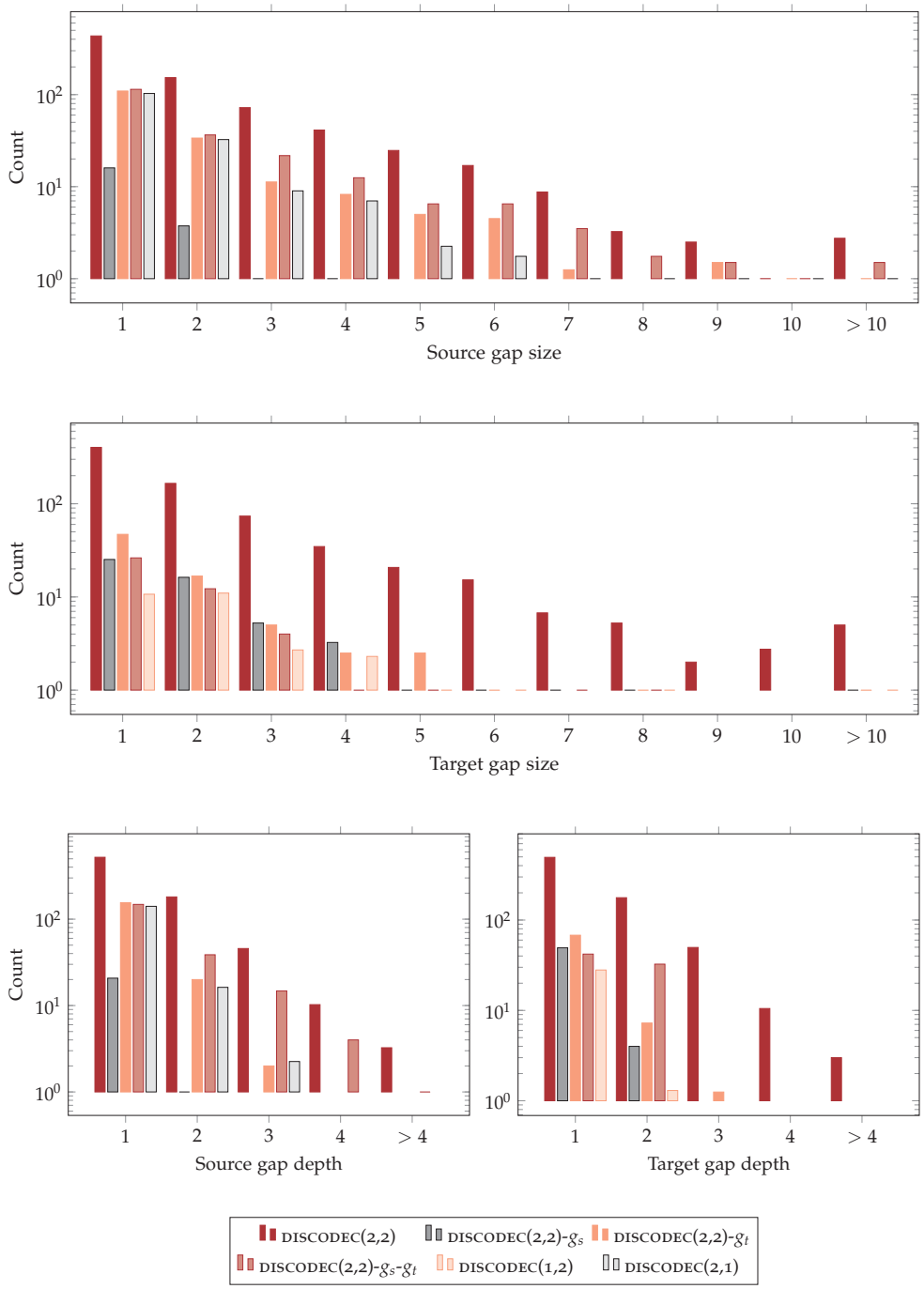


Figure 69: Analysis of the translations in the test set: distribution of sizes and depths of source and target gaps. The numbers are averaged over the systems obtained by multiple tunings.

addition, it is also depicted how deep the gaps are for the individual systems (cf. def. 2.15, p. 21), i. e. whether a gap is directly filled with material in the yield of the parent of the node that generated the gap (depth of 1) or whether the gap is maintained further towards the root of the tree. A general observation is that all distributions are roughly Zipfian. Note the log scale of the y-axes of the plots. I. e. most gaps are small and/or filled immediately, while large, deep gaps are rare.

As before, the successful systems DISCODEC(2,2)-g_t, DISCODEC(2,2)-g_s-g_t and DISCODEC(2,1) show a similar source gap size distribution, emphasizing again the importance of gaps on the source side. On the target side, DISCODEC(2,2)-g_t and DISCODEC(2,2)-g_s-g_t show a similar behaviour, while DISCODEC(2,1) obviously does not generate any target gaps. The absolute numbers of target discontinuities of DISCODEC(2,2)-g_t and DISCODEC(2,2)-g_s-g_t are however small anyway (table 12).

As for the *gap depth*, all systems show the tendency to close the gaps rather immediately. This is expected given that propagating the gap up requires the usage of additional (mixed) discontinuous rules which (a) might not have been learned in the first place, and (b) whose usage is penalized given the weights of the log-linear model. Furthermore, the complex alignment configurations can be derived with only gaps of depth 1. Nevertheless, the successful systems DISCODEC(2,2)-g_t, DISCODEC(2,2)-g_s-g_t and DISCODEC(2,1) also generate a few derivations with gaps of depth >1. Consider the derivation in figure 66 as an example.

6.4 DISCUSSION

Context to Related Work

In the previous chapter, which described the novel approach to machine translation using SLCFRS as the translation grammar formalism, we have worked out that two strands of work are similar to ours in that

a) they as well support discontinuities (to a certain degree) and thus allow to model (some of the) complex alignment configurations beyond SCFG,

b) they consider string-to-string translation, i. e. tree-based translation without syntactic information and

c) they implement a statistical machine translation (SMT) system including model training on large-scale data sets (e. g. *Europarl*).

Those approaches are *discontinuous phrase-based translation* (Galley and Manning, 2010) (pp. 155, 195ff.), the non-hierarchical counterpart to our approach, and *tree-based translation using Multi Bottom-up Tree Transducers (MBOTs)* (Braune et al., 2013; Seemann et al., 2015a,b) (pp. 165ff., 199ff.), which only allows for discontinuities on the target side. We already compared the approaches in terms of motivation and alignment capacity in section 4.4.2 and described the methodologies and experiments in section 5.4.

In Galley and Manning (2010), the discontinuous phrase-based approach is evaluated on a Chinese-to-English translation task. An improvement in BLEU of 0.77 over a phrase-based and of 1.03 over a hierarchical phrase-based baseline are reported. It would be desirable to see how our hierarchical *and* discontinuous phrase-based system performs in comparison. Unfortunately, we could not evaluate directly against their approach. The current documentation of *Phrasal*, where discontinuous phrase-based translation had been implemented, does not mention the discontinuous phrases anymore, which precludes training such a system on our data.[9] Neither could we obtain the data sets that have been used in Galley and Manning (2010): some of them had only been released for competitions and are not generally available.[10]

Tree-based translation using non-contiguous Synchronous Tree Sequence Substitution Grammar (STSSG) is not in the scope for further comparison since it only satisfies a) and half of c). As their motivation is to loosen the constraints imposed by SCFG-based tree-to-tree translation, Sun et al. (2009) exclusively consider tree-to-tree translation . They furthermore train on a rather small data set, which leads us to thinking that their approach or implementation does not scale.

The work on MBOT-based translation follows the same motivation as the approach using STSSG, namely to improve tree-to-tree and

9 `https://nlp.stanford.edu/wiki/Software/Phrasal`, accessed on June 27, 2015 and on November 1, 2017. Furthermore, C. Manning stated in February 2016 on the java-nlp-user mailing list that discontinuous phrases in *Phrasal* are not maintained anymore: `https://mailman.stanford.edu/pipermail/java-nlp-user/2016-February/007481.html`, accessed on November 1, 2017.

10 Galley and Manning (2010) use the same data as in Wang et al. (2007), including LDC databases which are *evaluation releases*, such as LDC2006E26 and LDC2005E83.

string-to-tree translation. In the corresponding experiments, language pairs which are expected to benefit from the modeling of target discontinuities are chosen, as MBOTs are asymmetrical and do not allow discontinuities on the source side. Besides the work on tree-based translation using syntactic information, Seemann et al. (2015b) also report on experiments for string-to-string translation for English-to-German and English-to-Chinese. While their string-to-string systems overall achieve the best results (better than the systems using syntactic information), the MBOT-based hierarchical system does not quite reach the translation performance of the SCFG-based hierarchical baseline system. This is unexpected in the light of the results of Galley and Manning (2010) who find that, for Chinese-to-English translation, source discontinuities are the major factor for a successful system, thus raising hope that English-to-Chinese translation would benefit from target discontinuities. Possible reasons for the unexpected result have been pointed out in section 5.4.2.

Our results for SLCFRS-based translation for German-to-English experiments in section 6.2.1 provide further evidence. Recall that formally MBOT and $(1 + v_t)_{1|v_t}$-SLCFRS, i. e. SLCFRS of source fan-out $v_s = 1$, are weakly equivalent. This means that our system setup for DISCODEC(1,2) corresponds roughly to the MBOT string-to-string setup in terms of discontinuities.[11] While there are many obvious differences between the two systems, e. g. the type of extracted rules, the value of the gap degree feature, the parameters for the rule extraction and the decoder, we still use our results to shed some more light onto the importance of source and target discontinuities in general.

From the systems in our experiments, DISCODEC(1,2) shows the worst translation performance for German-to-English translation according to automatic metrics. It is not able to improve over the SCFG-based baseline, while DISCODEC(2,1) and DISCODEC(2,2) with gap degree features modestly do so. Relating this finding with Seemann et al.'s MBOT results for English-to-German, as well as relating their result for English-to-Chinese with the results of Galley and Manning's Chinese-to-English experiments, one might say that it becomes apparent that, in (hierarchical) phrase-based translation, the characteristics

11 Obviously, while MBOT does not constrain the number of target tree fragments, the target fan-out of DISCODEC(1,2) is $v_t = 2$. However, Seemann et al. (2015b) present numbers which show that the number of MBOT rules with more than two target tree fragments is negligible in string-to-string systems.

of the languages themselves play only a minor role, if at all, when it comes to the usefulness of discontinuous phrases. Across all referenced experiments, source discontinuities consistently boost the translation performance (as measured by BLEU). In contrast to that, target discontinuities contribute little to the overall translation performance or they turn out to be rather detrimental.

To gain more insights about the role of *gaps* on the target side, Galley and Manning (2010) experiment with removing rules from their hierarchical baseline system which feature a non-terminal that represents a gap between two terminals. As they find that the BLEU score drops only slightly, they speculate that making effective use of gaps on the target side, in the form of discontinuities as well as non-terminals, is generally an issue. After all, generation is a more difficult problem than the mere matching of rules on the source side. As the generation of the translation is majorly influenced by the language model, this speculation goes hand in hand with a note we took earlier, see section 5.5. The wide-spread n-gram language model scoring is not optimal for hypotheses which do not expand from left to right, in particular discontinuous phrases are penalized with unreliable language model estimates in comparison to their continuous competitors. This might mean that target discontinuities will only be able to show their full potential with better generation mechanisms and/or improved language model scoring.

Translation Times

For the experiments reported on in this chapter, the processes were heavily parallelized. As they were furthermore performed on different compute architectures and on compute clusters shared with other users and their processes, an in-depth analysis of training and decoding times and a fair comparison is virtually impossible. Very roughly, for decoding the complete test set including grammar loading, DISCODEC(1,2) takes about twice as much time as DISCODEC(1,1), while DISCODEC(2,1) and DISCODEC(2,2) are by factor 3.5 and 55 slower than DISCODEC(1,1) respectively.

Given the theoretical time complexity results of section 5.3.2, it has been clear that allowing fan-outs $v > 2_{1|1}$ would lead to increased translation times. Whether an improvement in translation quality justifies increased decoding times, needs to be carefully evaluated in a

specific context. From the limited experimental results in this work, it seems that, from the trained *discodec* systems, DISCODEC(2,1) provides the best trade-off between speed and quality.

It is also worth noting that the primary aim of the implementation of *discodec* is functional correctness in order to provide a proof-of-concept of SLCFRS as the core of an SMT system. Efficiency and speed could surely be improved.

Further Experiments

Besides the fan-out v_s and v_t, also the number of (applicable) rules influences the decoding time. This number obviously grows with the more powerful translation models. One technique to limit this factor, which was not applied in the experiments, is to early prune the translation model itself by loading only a certain number of translations rules.

In addition, *slcfrs-extract* as well as *discodec* offer a set of hyperparameters which could be tuned further and experimented with. For our experiments, they were set according to the best of the community's knowledge as described in section 6.1.

Apart from that, experiments on other language pairs should be performed to evaluate our approach. As the results from related work suggest, Chinese-to-English would be a promising candidate.

6.5 CONCLUSION

Evaluating our hierarchical approach to machine translation which allows for discontinuous phrases on a German-to-English translation task revealed a modest improvement in BLEU and METEOR score over the SCFG-based baseline. Human evaluators showed a slight preference for translations produced by the SLCFRS system.

Our analysis of the translation results showed that it seems to be important to allow discontinuities on the source side while limiting the number of discontinuous constituents which are generated on the target side. This might be a preliminary result and worth revisiting in case of advancements in language-model scoring and generation for translation in general. Overall, the number of discontinuous constituents and generated complex alignment configurations as well as the ratio of concerned sentences is rather small. This corresponds to

the expectations, given the motivational analysis in section 3.2, and to some extend explains the small differences in the evaluation results.

7 CONCLUSION

7.1 CONTRIBUTIONS

In this thesis, a novel approach to tree-based statistical machine translation (SMT) has been explored. It makes use of a grammar formalism beyond Context-Free Grammar (CFG) for translation modeling. In this context, we have built the first SMT system based on Linear Context-Free Rewriting System (LCFRS). To the best of our knowledge, it is also the first hierarchical phrase-based system which allows for discontinuities on the source and on the target side.

To formalize translation modeling beyond CFG, we have defined Synchronous Linear Context-Free Rewriting System (SLCFRS), an extension to Synchronous Context-Free Grammar (SCFG) in which nonterminals span pairs of tuples of strings instead of just pairs of single strings and can thus model discontinuities. This has required the extension of well-established procedures for model training and decoding devised for SCFG to SLCFRS translation models. We have provided a practical implementation and an evaluation comparing several SLCFRS systems to each other and their SCFG-based counterpart. The analysis of the results has revealed a small, but significant improvement over the baseline when allowing for source discontinuities and constraining the number of target gaps.

We have motivated the move from SCFG to SLCFRS with the limitation of the space of alignment configurations which can be generated with SCFG. To that end, we have contributed a hierarchical aligner and an empirical investigation concerning the adequacy of SCFG and SLCFRS with respect to the alignment configurations which occur in a wide range of manually aligned data. A follow-up manual qualitative investigation revealed that only very few alignment configurations beyond SCFG can be attributed to alignment errors. In our investigations, we have furthermore discovered a class of alignment configurations which had not been described in the literature before.

LCFRS and its formal equivalents are well-studied formalisms for modeling the syntax of natural languages. They have also demon-

strated their potential for parsing discontinuous constituents in a probabilistic, data-driven setup. However, literature on using LCFRS for translation modeling has been sparse and remained theoretical. The presented work thus represents an actual application of LCFRS and LCFRS parsing to SMT. In addition, an extensive overview of related approaches has been provided in order to put our approach into context.

7.2 FUTURE WORK

A lot of work could still be done. Shortcomings of the implemented string-to-string SLCFRS translation system have been pointed out in section 5.5. Suggestions for additional experiments are given in section 6.4. Further directions of future work will be described in the following.

USE OF SYNTACTIC STRUCTURES WHICH SUPPORT DISCONTINUITIES Translation modeling within this work has not made use of monolingual syntactic information so far: the presented hierarchical phrase-based translation model is syntactic only in a formal sense. For SCFG-based translation models, various approaches for string-to-tree, tree-to-string and tree-to-tree models have been worked out in the literature, building syntactic trees on the target side, the source side or both respectively during decoding (see p. 86ff. for some references). Generally, these approaches make use of a probabilistic parser which produces context-free constituency structures for the sentences in the training corpus. It is however known that certain phenomena in the syntax of natural languages cannot be modeled adequately with CFG and that more expressive modeling mechanisms, e. g. mildly context-sensitive grammar formalisms like LCFRS, are necessary (see section 2.1.3). We suppose that more adequate monolingual modeling would lead to better translational correspondences and thus to translations of better quality.

SLCFRS is a natural candidate for translation modeling in string-to-tree, tree-to-string and tree-to-tree approaches where the trees represent LCFRS derivations. It should be possible to extend the approaches which have been established for SCFG-based systems rather directly to SLCFRS, i. e. such that they make use of constituency trees supporting discontinuous constituents. Note that with Multi Bottom-up Tree

Transducer (MBOT) (p. 165) and Synchronous Tree Sequence Substitution Grammar (STSSG) (p. 162), this is not directly possible. Even though they are expressive enough to generate (some of the) complex alignment configurations and to model discontinuities, their synchronous parse trees exclusively feature continuous constituents, thus making grammar extraction from trees with discontinuities non-trivial. The tree-based approaches which have been developed using those formalisms for translation modeling (p. 199ff.) all rely on CFG parse trees.[1]

Tree-based approaches which rely on tree fragments instead of flat rules, e. g. by using Synchronous Tree-Substitution Grammar (STSG), can also be extended to support the modeling of discontinuous constituents. In the context of Data-Oriented Parsing (DOP), *Discontinuous Tree-Substitution Grammar (DTSG)* has been defined and used (van Cranenburgh et al., 2016), which would be suitable for translation modeling by extending it to its synchronous version. A Synchronous DTSG would also be an instance of a *tree-rewriting* SLCFRS.

For the monolingual parsing, one would make use of recent advances in monolingual parsing of discontinuous constituents. It is now possible to parse large amounts of text, e. g. an SMT training corpus, producing constituency trees which allow for discontinuous constituents with good accuracy and speed (see p. 65 for the references).

NEURAL MACHINE TRANSLATION BEYOND CFG On a final note, let us glance at the new big player in the field, neural machine translation (NMT). Extensions to the plain sequence-to-sequence learning have been proposed just recently which make use of syntactic structure in the form of context-free constituency trees on the source or the target side in some way (Eriguchi et al., 2016, 2017; Aharoni and Goldberg, 2017). In this context, one can reason in the same way as for SMT models: in order to adequately describe the syntax of the individual languages, constituency structures which support discontinuous constituents should be used, thus also pushing NMT beyond context-freeness.

1 Seemann and Maletti (2015) even remove the non-projective dependencies, which would lead to discontinuous constituents, in their data preparation procedure for MBOT.

A

ACRONYMS

A.1 FORMALISMS

Many grammar formalisms and frameworks have been mentioned in this thesis. They are listed in the following, together with the acronyms that have been used.

CFG	Context-Free Grammar
DTSG	Discontinuous Tree-Substitution Grammar
GMTG	Generalized Multitext Grammar
ITG	Inversion Transduction Grammar
LCFRS	Linear Context-Free Rewriting System
MBOT	Multi Bottom-up Tree Transducer
MCFG	Multiple Context-Free Grammar
MCTAG	Multi-Component Tree-Adjoining Grammar
NF-ITG	Normal-form Inversion Transduction Grammar
NF-SCFG	Normal-form Synchronous Context-Free Grammar
NF-SLCFRS	Normal-form Synchronous Linear Context-Free Rewriting System
PMCFG	Parallel Multiple Context-Free Grammar
RCG	Range Concatenation Grammar
RCT	Range Concatenation Transducer
SCFG	Synchronous Context-Free Grammar
SCFTG	Synchronous (Simple) Context-Free Tree Grammar

SDTG Syntax-Directed Transduction Grammar

SLCFRS Synchronous Linear Context-Free Rewriting System

SRCG Simple Range Concatenation Grammar

SRCT Simple Range Concatenation Transducer

STAG Synchronous Tree-Adjoining Grammar

STIG Synchronous Tree-Insertion Grammar

STSG Synchronous Tree-Substitution Grammar

STSSG Synchronous Tree Sequence Substitution Grammar

TAG Tree-Adjoining Grammar

TIG Tree-Insertion Grammar

TSG Tree-Substitution Grammar

A.2 MISCELLANEOUS

The following list contains other acronyms which have been used.

CDT Copenhagen Dependency Treebank

CDTU cross-serial discontinuous translation unit

CNF Chomsky Normal Form

DOP Data-Oriented Parsing

DTU discontinuous translation unit

IO inside-out

IO-DTU inside-out discontinuous translation unit

LHS left-hand side

NLP natural language processing

NMT neural machine translation

RHS	right-hand side
SMT	statistical machine translation
SVO	subject-verb-object
TUER	translation unit error rate

BIBLIOGRAPHY

Abeillé, A., Schabes, Y., and Joshi, A. K. (1990). Using lexicalized TAGs for machine translation. In *Proceedings of the 13th Conference on Computational linguistics*. Association for Computational Linguistics.

Abeillé, A. (1988). Parsing French with tree adjoining grammar: Some linguistic accounts. In *Proceedings of the 12th Conference on Computational Linguistics - Volume 1*, COLING '88, pages 7–12.

Aharoni, R. and Goldberg, Y. (2017). Towards string-to-tree neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Short Papers)*.

Aho, A. V. and Ullman, J. D. (1972). *The theory of parsing, translation, and compiling*. Prentice-Hall, Inc.

Arnold, A. and Dauchet, M. (1982). Morphismes et bimorphismes d'arbres. *Theoretical Computer Science*, 20(1):33 – 93.

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*.

Becker, T., Joshi, A. K., and Rambow, O. (1991). Long-distance scrambling and Tree-Adjoining Grammars. In *Proceedings of the Fifth Conference of the European Chapter of the Association for Computational Linguistics*, pages 21–26, Berlin, Germany. Association for Computational Linguistics.

Becker, T., Rambow, O., and Niv, M. (1992). The derivational generative power of formal systems or scrambling is beyond LCFRS. IRCS report 92-38, University of Pennsylvania, Philadelphia, PA.

Bellman, R. (1957). *Dynamic Programming*. Princeton University Press, Princeton.

Bertsch, E. and Nederhof, M.-J. (2001). On the complexity of some extensions of RCG parsing. In *Proceedings of the Seventh International Workshop on Parsing Technologies*, pages 66–77, Beijing, China.

Booth, T. L. and Thomson, R. A. (1973). Applying probability measures to abstract languages. *IEEE Transactions on Computers*, C-22(5):442–450.

Boullier, P. (1998). Proposal for a Natural Language Processing syntactic backbone. Technical Report 3342, INRIA.

Boullier, P. (1999). Chinese numbers, MIX, scrambling, and Range Concatenation Grammars. In *Proceedings of the Ninth Conference of the European Chapter of the Association for Computational Linguistics*, pages 53–60, Bergen, Norway. Association for Computational Linguistics.

Boullier, P. (2000). Range Concatenation Grammars. In *Proceedings of the Sixth International Workshop on Parsing Technologies*, pages 53–64, Trento, Italy.

Boyd, A. (2007). Discontinuity revisited: An improved conversion to context-free representations. In *Proceedings of The Linguistic Annotation Workshop (LAW) at ACL 2007*, pages 41–44, Prague, Czech Republic. Association for Computational Linguistics.

Brants, S., Dipper, S., Hansen, S., Lezius, W., and Smith, G. (2002). The TIGER Treebank. In Hinrichs, E. W. and Simov, K., editors, *Proceedings of the 1st Workshop on Treebanks and Linguistic Theories*, pages 24–42, Sozopol, Bulgaria.

Braune, F. (2015). *Decoding Strategies for Syntax-based Statistical Machine Translation*. PhD thesis, Stuttgart, Germany.

Braune, F., Seemann, N., Quernheim, D., and Maletti, A. (2013). Shallow local multi-bottom-up tree transducers in statistical machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 811–821. Association for Computational Linguistics.

Bresnan, J., Kaplan, R. M., Peters, S., and Zaenen, A. (1982). Cross-serial dependencies in Dutch. *Linguistic Inquiry*, 13(4):613–635. Reprinted in Savitch et al. (1987).

Brown, P. F., Cocke, J., Pietra, S. A. D., Pietra, V. J. D., Jelinek, F., Lafferty, J. D., Mercer, R. L., and Roossin, P. S. (1990). A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85.

Brown, P. F., Pietra, V. J. D., Pietra, S. A. D., and Mercer, R. L. (1993). The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

Buch-Kromann, M., Korzen, I., and Müller, H. H. (2009). Uncovering the 'lost' structure of translations with parallel treebanks. *Copenhagen Studies in Language*, 38:199–224.

Burbank, A., Carpuat, M., Clark, S., Dreyer, M., Fox, P., Groves, D., Hall, K., Hearne, M., Melamed, I. D., Shen, Y., et al. (2005). Final report of the 2005 language engineering workshop on statistical machine translation by parsing. Technical report, Johns Hopkins University.

Burden, H. and Ljunglöf, P. (2005). Parsing Linear Context-Free Rewriting Systems. In *Proceedings of the Ninth International Workshop on Parsing Technology*, pages 11–17, Vancouver, BC. Association for Computational Linguistics.

Carreras, X. and Collins, M. (2009). Non-projective parsing for statistical machine translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 200–209. Association for Computational Linguistics.

Chappelier, J. and Rajman, M. (1998). A generalized CYK algorithm for parsing stochastic CFG. In *Proceedings of the First Workshop on Tabulation in Parsing and Deduction (TAPD98)*, pages 133–137.

Charniak, E. (1997). Statistical parsing with a Context-Free Grammar and word statistics. In *Proceedings of the 14th National Conference on Artificial Intelligence*, pages 598–603.

Charniak, E. (2000). A maximum-entropy-inspired parser. In *Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 132–139, Seattle, WA. Association for Computational Linguistics.

Chiang, D. (2004). Uses and abuses of intersected languages. In *Proceedings of the Seventh International Workshop on Tree Adjoining Grammar and Related Formalisms (TAG+7)*, pages 9–15, Vancouver, Canada.

Chiang, D. (2005). A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 263–270. Association for Computational Linguistics.

Chiang, D. (2007). Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Chiang, D. (2010). Learning to translate with source and target syntax. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1443–1452. Association for Computational Linguistics.

Cho, K., van Merriënboer, B., Bahdanau, D., and Bengio, Y. (2014). On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar. Association for Computational Linguistics.

Chomsky, N. (1956). Three models for the description of language. *Information Theory, IEEE Transactions*, 2(3):113–124.

Clark, J. H., Dyer, C., Lavie, A., and Smith, N. A. (2011). Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 176–181, Portland, Oregon, USA. Association for Computational Linguistics.

Coavoux, M. and Crabbé, B. (2017). Incremental discontinuous phrase structure parsing with the gap transition. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1259–1270, Valencia, Spain. Association for Computational Linguistics.

Cocke, J. and Schwartz, J. T. (1970). Programming languages and their compilers: Preliminary notes. Technical report, Courant Institute of Mathematical Sciences, New York University.

Collins, M. (1999). *Head-Driven Statistical Models for Natural Language Parsing*. PhD thesis, University of Pennsylvania, Philadelphia, PA.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2001). *Introduction to Algorithms*. MIT Press, second edition.

Crego, J. M., Costa-Jussà, M. R., Mariño, J. B., and Fonollosa, J. A. R. (2005). N-gram-based versus phrase-based statistical machine translation. In *Proceedings of the International Workshop on Spoken Language Technology (IWSLT'05)*, pages 167–174, Pittsburgh, PA, USA.

Crego, J. M. and Yvon, F. (2009). Gappy translation units under left-to-right SMT decoding. In *Proceedings of the annual meeting of the European Association for Machine Translation (EAMT)*, pages 66–73, Barcelona, Spain.

DeNeefe, S. and Knight, K. (2009). Synchronous Tree Adjoining machine translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 727–736. Association for Computational Linguistics.

DeNero, J., Bansal, M., Pauls, A., and Klein, D. (2009). Efficient parsing for transducer grammars. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL '09)*, pages 227–235, Stroudsburg, PA, USA. Association for Computational Linguistics.

Denkinger, T. (2015). A Chomsky-Schützenberger representation for weighted multiple context-free languages. In *Proceedings of the 12th International Conference on Finite-State Methods and Natural Language Processing (FSMNLP 2015)*.

Denkowski, M. and Lavie, A. (2011). Meteor 1.3: Automatic metric for reliable optimization and evaluation of machine translation systems. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 85–91. Association for Computational Linguistics.

Dienes, P. and Dubey, A. (2003). Antecedent recovery: Experiments with a trace tagger. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 33–40, Sapporo, Japan. Association for Computational Linguistics.

Dras, M. (1999). A meta-level grammar: Redefining Synchronous TAG for translation and paraphrase. In *Proceedings of the 37th Annual*

Meeting of the Association for Computational Linguistics on Computational Linguistics, pages 80–87.

Dreyer, M., Hall, K., and Khudanpur, S. (2007). Comparing reordering constraints for SMT using efficient BLEU oracle computation. In *Proceedings of SSST, NAACL-HLT 2007 / AMTA Workshop on Syntax and Structure in Statistical Translation*, pages 103–110. Association for Computational Linguistics.

Earley, J. (1970). An efficient context-free parsing algorithm. *Communications of the Association for Computing Machinery*, 13(2):94–102.

Eisele, A. and Chen, Y. (2010). MultiUN: A multilingual corpus from United Nation documents. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, Valletta, Malta. European Languages Resources Association (ELRA).

Eisner, J. (2003). Learning non-isomorphic tree mappings for machine translation. In *The Companion Volume to the Proceedings of 41st Annual Meeting of the Association for Computational Linguistics (ACL '03)*, pages 205–208.

Engelfriet, J., Lilin, E., and Maletti, A. (2009). Extended multi bottom-up tree transducers: Composition and decomposition. *Acta Inf.*, 46(8):561–590.

Eriguchi, A., Hashimoto, K., and Tsuruoka, Y. (2016). Tree-to-sequence attentional neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 823–833, Berlin, Germany. Association for Computational Linguistics.

Eriguchi, A., Tsuruoka, Y., and Cho, K. (2017). Learning to parse and translate improves neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Short Papers)*, pages 72–78. Association for Computational Linguistics.

Evang, K. and Kallmeyer, L. (2011). PLCFRS parsing of English discontinuous constituents. In *Proceedings of the 12th International Conference on Parsing Technologies (IWPT)*, pages 104–116, Dublin, Ireland. Association for Computational Linguistics.

Federmann, C. (2012). Appraise: An open-source toolkit for manual evaluation of machine translation output. *The Prague Bulletin of Mathematical Linguistics*, 98:25–35.

Fernández-González, D. and Martins, A. (2015). Parsing as reduction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 1523–1533. Association for Computational Linguistics.

Gaifman, H. (1965). Dependency systems and phrase-structure systems. *Information and Control*, 8:304–337.

Galley, M., Graehl, J., Knight, K., Marcu, D., DeNeefe, S., Wang, W., and Thayer, I. (2006). Scalable inference and training of context-rich syntactic translation models. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 961–968. Association for Computational Linguistics.

Galley, M., Hopkins, M., Knight, K., and Marcu, D. (2004). What's in a translation rule? In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*. Association for Computational Linguistics.

Galley, M. and Manning, C. D. (2008). A simple and effective hierarchical phrase reordering model. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 848–856. Association for Computational Linguistics.

Galley, M. and Manning, C. D. (2010). Accurate non-hierarchical phrase-based translation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 966–974, Los Angeles, California. Association for Computational Linguistics.

Gallo, G., Longo, G., Pallottino, S., and Nguyen, S. (1993). Directed hypergraphs and applications. *Discrete Applied Mathematics*, 42(2):177 – 201.

Gazdar, G., Klein, E., Pullum, G. K., and Sag, I. (1985). *Generalized Phrase Structure Grammar*. Basil Blackwell, Oxford.

Gildea, D. (2012). On the string translations produced by multi bottom–up tree transducers. *Computational Linguistics*, 38(3):673–693.

Gómez-Rodríguez, C., Kuhlmann, M., Satta, G., and Weir, D. (2009). Optimal reduction of rule length in Linear Context-Free Rewriting Systems. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 539–547, Boulder, CO. Association for Computational Linguistics.

Goodman, J. (1999). Semiring parsing. *Computational Linguistics*, 25(4):573–605.

Goutte, C., Yamada, K., and Gaussier, E. (2004). Aligning words using matrix factorisation. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04)*, pages 502–509, Barcelona, Spain.

Graça, J., Pardal, J. P., Coheur, L., and Caseiro, D. A. (2008a). Building a golden collection of parallel multi-language word alignment. In *Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC'08)*.

Graça, J., Pardal, J. P., Coheur, L., and Caseiro, D. A. (2008b). Multi-language word alignments annotation guidelines. Technical report, Spoken Language Systems Laboratory, Lisboa, Portugal.

Graehl, J., Knight, K., and May, J. (2008). Training tree transducers. *Computational Linguistics*, 34(3):391–427.

Green, S., Cer, D., and Manning, C. D. (2014). Phrasal: A toolkit for new directions in statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 114–121. Association for Computational Linguistics.

Grune, D. and Jacobs, C. (2008). *Parsing Techniques. A Practical Guide*. Monographs in Computer Science. Springer, second edition.

Hall, J. and Nivre, J. (2008). Parsing discontinuous phrase structure with grammatical functions. In Nordström, B. and Ranta, A., editors, *Advances in Natural Language Processing*, pages 169–180. Springer Berlin Heidelberg.

Harbusch, K. and Poller, P. (2013). Structural translation with Synchronous Tree-Adjoining Grammars in Verbmobil. Technical report verbmobil, Universität Koblenz-Landau/DFKI GmbH.

Hoang, H. and Koehn, P. (2010). Improved translation with source syntax labels. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 409–417. Association for Computational Linguistics.

Höhle, T. (1983). *Topologische Felder*. PhD thesis, Universität zu Köln.

Holmqvist, M. and Ahrenberg, L. (2011). A gold standard for English–Swedish word alignment. In *Proceedings of the 18th Nordic Conference of Computational Linguistics NODALIDA 2011*, pages 106–113.

Hopcroft, J. E. and Ullman, J. D. (1979). *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley.

Hopkins, M. and Langmead, G. (2010). SCFG decoding without binarization. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 646–655. Association for Computational Linguistics.

Huang, L. and Chiang, D. (2005). Better k-best parsing. In *Proceedings of the Ninth International Workshop on Parsing Technology*, pages 53–64. Association for Computational Linguistics.

Huang, L. and Chiang, D. (2007). Forest rescoring: Faster decoding with integrated language models. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 144–151, Prague, Czech Republic. Association for Computational Linguistics.

Huang, L., Knight, K., and Joshi, A. (2006). Statistical syntax-directed translation with extended domain of locality. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas (AMTA)*, pages 66–73. The Association for Machine Translation in the Americas.

Jelinek, F., Lafferty, J. D., and Mercer, R. L. (1992). Basic methods of probabilistic context free grammars. In Laface, P. and De Mori, R., editors, *Speech Recognition and Understanding: Recent Advances, Trends and Applications*, pages 345–360. Springer Berlin Heidelberg.

Johnson, M. (2002). A simple pattern-matching algorithm for recovering empty nodes and their antecedents. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 136–143, Philadelphia, PA. Association for Computational Linguistics.

Joshi, A. K. (1985). Tree adjoining grammars: How much context-sensitivity is necessary for characterizing structural descriptions? In Dowty, D., Karttunen, L., and Zwicky, A., editors, *Natural language processing: Theoretical, computational and psychological perspectives*, pages 206–250. Cambridge University Press, New York.

Joshi, A. K., Levy, L. S., and Takahashi, M. (1975). Tree Adjunct Grammars. *Journal of Computer and System Science*, 10(1):136–163.

Junczys-Dowmunt, M., Dwojak, T., and Hoang, H. (2016). Is neural machine translation ready for deployment? A case study on 30 translation directions. In *Proceedings of the 9th International Workshop on Spoken Language Translation (IWSLT)*, Seattle, WA.

Kaeshammer, M. (2013). Synchronous linear context-free rewriting systems for machine translation. In *Proceedings of the Seventh Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 68–77, Atlanta, Georgia. Association for Computational Linguistics.

Kaeshammer, M. (2015). Hierarchical machine translation with discontinuous phrases. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 228–238, Lisbon, Portugal. Association for Computational Linguistics.

Kaeshammer, M. and Demberg, V. (2012). German and English treebanks and lexica for Tree-Adjoining Grammars. In *Proceedings of the Eighth International Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey. European Language Resources Association (ELRA).

Kaeshammer, M. and Westburg, A. (2014). On complex word alignment configurations. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 1773–1780, Reykjavik, Iceland. European Language Resources Association (ELRA).

Kahane, S., Candito, M.-H., and de Kercadio, Y. (2000). An alternative description of extractions in TAG. In *Proceedings of the Fifth International Workshop on Tree Adjoining Grammar and Related Frameworks (TAG+5)*, pages 115–122, Paris, France.

Kallmeyer, L. (2010). *Parsing beyond context-free grammars*. Springer, Berlin, Heidelberg.

Kallmeyer, L. and Maier, W. (2009). An incremental Earley parser for Simple Range Concatenation Grammar. In *Proceedings of the 11th International Conference on Parsing Technologies (IWPT'09)*, pages 61–64, Paris, France. Association for Computational Linguistics.

Kallmeyer, L. and Maier, W. (2013). Data-driven parsing using Probabilistic Linear Context-Free Rewriting Systems. *Computational Linguistics*, 39(1).

Kanazawa, M. (2009). The pumping lemma for well-nested multiple context-free languages. In Diekert, V. and Nowotka, D., editors, *Developments in Language Theory*, pages 312–325, Berlin, Heidelberg. Springer.

Kasami, T. (1965). An efficient recognition and syntax-analysis algorithm for context-free languages. Scientific report AFCRL-65-758, Air Force Cambridge Research Lab, Bedford, MA.

Kato, Y., Seki, H., and Kasami, T. (2006). Stochastic multiple Context-Free Grammar for RNA pseudoknot modeling. In *Proceedings of the Eighth International Workshop on Tree Adjoining Grammar and Related Formalisms (TAG+8)*, pages 57–64. Association for Computational Linguistics.

Klein, D. and Manning, C. D. (2001). Parsing and hypergraphs. In *Proceedings of the Seventh International Workshop on Parsing Technologies*, Beijing, China.

Knight, K. (1999). Decoding complexity in word-replacement translation models. *Computational Linguistics*, 25(4):607–615.

Knight, K. and Graehl, J. (2005). An overview of probabilistic tree transducers for natural language processing. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 1–24. Springer.

Knuth, D. E. (1977). A generalization of Dijkstra's algorithm. *Information Processing Letters*, 6(1).

Koehn, P. (2004). Pharaoh: A beam search decoder for phrase-based statistical machine translation models. In *Proceedings of the 6th Conference of the Association for Machine Translation in the Americas (AMTA) - Machine Translation: From Real Users to Research*, pages 115–124. Springer Berlin Heidelberg.

Koehn, P. (2005). Europarl: A parallel corpus for statistical machine translation. In *Proceedings of the tenth Machine Translation Summit*, pages 79–86.

Koehn, P. (2010). *Statistical Machine Translation*. Cambridge University Press.

Koehn, P., Axelrod, A., Birch, A., Callison-Burch, C., Osborne, M., Talbot, D., and White, M. (2005). Edinburgh system description for the 2005 IWSLT speech translation evaluation. In *Proceedings of the International Workshop on Spoken Language Translation (IWSLT) 2005*, pages 68–75.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., and Herbst, E. (2007). Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180. Association for Computational Linguistics.

Koehn, P., Och, F. J., and Marcu, D. (2003). Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 48–54. Association for Computational Linguistics.

Kracht, M. (2003). *The Mathematics of Language*. Mouton de Gruyter, Berlin.

Kuhlmann, M. (2007). *Dependency Structures and Lexicalized Grammars*. Doctoral dissertation, Saarland University, Saarbrücken, Germany.

Kuhlmann, M. and Satta, G. (2009). Treebank grammar techniques for non-projective dependency parsing. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 478–486, Athens, Greece. Association for Computational Linguistics.

Lambert, P., Gispert, A., Banchs, R., and Mariño, J. B. (2005). Guidelines for word alignment evaluation and manual alignment. *Language Resources and Evaluation*, 39:267–285.

Levenberg, A., Dyer, C., and Blunsom, P. (2012). A Bayesian model for learning SCFGs with discontiguous rules. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 223–232, Jeju Island, Korea.

Levy, R. (2005). *Probabilistic Models of Word Order and Syntactic Discontinuity*. PhD thesis, Stanford University.

Levy, R. and Manning, C. (2004). Deep dependencies from context-free statistical parsers: Correcting the surface dependency approximation. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 327–334, Barcelona, Spain. Association for Computational Linguistics.

Lewis, II, P. M. and Stearns, R. E. (1968). Syntax-directed transduction. *Journal of the ACM*, 15(3):465–488.

Lilin, E. (1981). *Proprietes de cloture d'une extension de transducteurs d'arbres deterministes*, pages 280–289. Springer Berlin Heidelberg, Berlin, Heidelberg.

Liu, Y., Liu, Q., and Lin, S. (2006). Tree-to-string alignment template for statistical machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, pages 609–616, Sydney, Australia. Association for Computational Linguistics.

Liu, Y., Liu, Q., and Lü, Y. (2011). Adjoining tree-to-string translation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1278–1287. Association for Computational Linguistics.

Ljunglöf, P. (2004). *Expressivity and Complexity of the Grammatical Framework*. PhD thesis, Göteborg University, Gothenburg, Sweden.

Lopez, A. (2007). Hierarchical phrase-based translation with suffix arrays. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 976–985.

Lopez, A. (2008). Statistical machine translation. *ACM Computing Surveys*, 40(3).

Macken, L. (2010). An annotation scheme and gold standard for Dutch-English word alignment. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*. European Language Resources Association (ELRA).

Maier, W. (2010). Direct parsing of discontinuous constituents in German. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 58–66. Association for Computational Linguistics.

Maier, W. (2013). *Parsing Discontinuous Structures*. PhD thesis, Tübingen, Germany.

Maier, W. (2015). Discontinuous incremental shift-reduce parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1202–1212, Beijing, China. Association for Computational Linguistics.

Maier, W., Kaeshammer, M., and Kallmeyer, L. (2012). PLCFRS parsing revisited: Restricting the fan-out to two. In *Proceedings of the Eleventh International Conference on Tree Adjoining Grammars and Related Formalisms (TAG+11)*, pages 126–134, Paris, France.

Maier, W. and Kallmeyer, L. (2010). Discontinuity and non-projectivity: Using mildly context-sensitive formalisms for data-driven parsing. In *Proceedings of the Tenth International Workshop on Tree Adjoining Grammar and Related Formalisms (TAG+10)*, pages 119–126, New Haven, CT.

Maier, W. and Lichte, T. (2011). Characterizing discontinuity in constituent treebanks. In *Formal Grammer 2009, Revised Selected Papers*, volume 5591 of *LNAI*. Springer.

Maier, W. and Lichte, T. (2016). Discontinuous parsing with continuous trees. In *Proceedings of the Workshop on Discontinuous Structures in Natural Language Processing*, pages 47–57, San Diego, California. Association for Computational Linguistics.

Maier, W. and Søgaard, A. (2008). Treebanks and mild context-sensitivity. In de Groote, P., editor, *Proceedings of the 13th Conference on Formal Grammar (FG-2008)*, pages 61–76, Hamburg, Germany. CSLI Publications.

Maillette de Buy Wenniger, G., Khalilov, M., and Sima'an, K. (2010). A toolkit for visualizing the coherence of tree-based reordering with word-alignments. *The Prague Bulletin of Mathematical Linguistics*, 94:97–106.

Maillette de Buy Wenniger, G. and Sima'an, K. (2013). A formal characterization of parsing word alignments by synchronous grammars with empirical evidence to the ITG hypothesis. In *Proceedings of the Seventh Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 58–67, Atlanta, Georgia. Association for Computational Linguistics.

Maletti, A. (2010a). A tree transducer model for synchronous tree-adjoining grammars. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1067–1076. Association for Computational Linguistics.

Maletti, A. (2010b). Why synchronous tree substitution grammars? In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 876–884. Association for Computational Linguistics.

Maletti, A. (2011). How to train your multi bottom-up tree transducer. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 825–834. Association for Computational Linguistics.

Marcus, M., Kim, G., Marcinkiewicz, M. A., Macintyre, R., Bies, A., Ferguson, M., Katz, K., and Schasberger, B. (1994). The Penn Treebank: Annotating predicate argument structure. In *Proceedings of the Workshop on Human Language Technology*, pages 114–119.

Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. (1993). Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330. Special Issue on Using Large Corpora: II.

Mariño, J. B., Banchs, R. E., Crego, J. M., de Gispert, A., Lambert, P., Fonollosa, J. A., and Costa-Jussà, M. R. (2006). N-gram-based machine translation. *Computational Linguistics*, 32(4):527–549.

Martin, J., Mihalcea, R., and Pedersen, T. (2005). Word alignment for languages with scarce resources. In *Proceedings of the ACL Workshop on Building and Using Parallel Texts*, pages 65–74. Association for Computational Linguistics.

Melamed, I. D. (1998). Annotation style guide for the Blinker Project. Technical Report IRCS-98-06, University of Pennsylvania.

Melamed, I. D. (2003). Multitext grammars and synchronous parsers. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 79–86.

Melamed, I. D. (2004). Statistical machine translation by parsing. In *Proceedings of the 42nd Annual Conference of the Association for Computational Linguistics (ACL-04)*.

Melamed, I. D., Satta, G., and Wellington, B. (2004). Generalized multitext grammars. In *Proceedings of the 42nd Annual Conference of the Association for Computational Linguistics (ACL-04)*.

Menezes, A. and Quirk, C. (2007). Using dependency order templates to improve generality in translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 1–8, Prague, Czech Republic. Association for Computational Linguistics.

Menezes, A. and Quirk, C. (2008). Syntactic models for structural word insertion and deletion during translation. In *Proceedings of the*

2008 Conference on Empirical Methods in Natural Language Processing, pages 735–744. Association for Computational Linguistics.

Mihalcea, R. and Pedersen, T. (2003). An evaluation exercise for word alignment. In *Proceedings of the HLT-NAACL 2003 Workshop on Building and Using Parallel Texts: Data Driven Machine Translation and Beyond*, pages 1–10.

Nederhof, M.-J. (2003). Weighted deductive parsing and Knuth's algorithm. *Computational Lingustics*, 29(1).

Nederhof, M.-J. and Vogler, H. (2012). Synchronous context-free tree grammars. In *Proceedings of the Eleventh International Conference on Tree Adjoining Grammars and Related Formalisms (TAG+11)*, pages 55–63, Paris, France.

Nelder, J. A. and Mead, R. (1965). A simplex method for function minimization. *Computing Journal*, 7(4):308–313.

Nesson, R., Rush, A., and Shieber, S. (2006). Induction of probabilistic Synchronous Tree-Insertion Grammars for machine translation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas*, pages 128–137. Association for Machine Translation in the Americas.

Och, F. J. (2003). Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 160–167. Association for Computational Linguistics.

Och, F. J. and Ney, H. (2000). Improved statistical alignment models. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 440–447. Association for Computational Linguistics.

Och, F. J. and Ney, H. (2002). Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 295–302.

Och, F. J. and Ney, H. (2003). A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Och, F. J. and Ney, H. (2004). The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–449.

Och, F. J., Tillmann, C., and Ney, H. (1999). Improved alignment models for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC)*.

Padó, S. and Lapata, M. (2006). Optimal constituent alignment with edge covers for semantic projection. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 1161–1168, Sydney, Australia. Association for Computational Linguistics.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318.

Pereira, F. C. N. and Warren, D. H. D. (1983). Parsing as deduction. In *Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics*, pages 137–144, Cambridge, MA. Association for Computational Linguistics.

Petrov, S. (2009). *Coarse-to-Fine Natural Language Processing*. PhD thesis, University of California at Berkeley, Berkeley, CA.

Popović, M. (2016). CHRF deconstructed: β parameters and n-gram weights. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pages 499–504, Berlin, Germany. Association for Computational Linguistics.

Quirk, C., Menezes, A., and Cherry, C. (2005). Dependency treelet translation: Syntactically informed phrasal SMT. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 271–279. Association for Computational Linguistics.

Rambow, O. and Satta, G. (1999). Independent parallelism in finite copying parallel rewriting systems. *Theoretical Computer Science*, 223(1-2):87–120.

Ranta, A. (2004). Grammatical Framework, a typetheoretical grammar formalism. *Journal of Functional Programming*, 14(2):145–189.

Rosenkrantz, D. J. and Lewis, P. M. (1970). Deterministic left-corner parsing. In *Proceedings of the 11th Annual Symposium on Switching and Automata Theory (SWAT 1970)*, pages 139–152, Washington, DC. IEEE Computer Society.

Saers, M., Wu, D., and Quirk, C. (2011). On the expressivity of linear transductions. *Proceedings of the 13th Machine Translation Summit (MT Summit XIII)*, pages 431–438.

Sagot, B. (2005). Linguistic facts as predicates over ranges of the sentence. In Blache, P., Stabler, E., Busquets, J., and Moot, R., editors, *Proceedings of Logical Aspects of Computational Linguistics, 5th International Conference (LACL 2005)*, volume 3492, pages 271–286, Bordeaux, France. Springer.

Satta, G. and Peserico, E. (2005). Some computational complexity results for synchronous context-free grammars. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 803–810.

Savitch, W. J., Bach, E., Marsh, W., and Safran-Naveh, G., editors (1987). *The Formal Complexity of Natural Language*. Springer Netherlands, Dordrecht.

Schabes, Y. and Joshi, A. K. (1988). An Earley-type parsing algorithm for Tree Adjoining Grammars. In *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics*, pages 258–269, Buffalo, NY. Association for Computational Linguistics.

Seemann, N., Braune, F., and Maletti, A. (2015a). String-to-tree multi bottom-up tree transducers. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 815–824, Beijing, China. Association for Computational Linguistics.

Seemann, N., Braune, F., and Maletti, A. (2015b). A systematic evaluation of MBOT in statistical machine translation. In *Proceedings of the 15th MT Summit*, pages 200–214.

Seemann, N. and Maletti, A. (2015). Discontinuous statistical machine translation with target-side dependency syntax. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 239–247, Lisbon, Portugal. Association for Computational Linguistics.

Seki, H., Matsumura, T., Fujii, M., and Kasami, T. (1991). On Multiple Context-Free Grammars. *Theoretical Computer Science*, 88(2):191–229.

Sennrich, R. (2015). Modelling and optimizing on syntactic n-grams for statistical machine translation. *Transactions of the Association for Computational Linguistics*, 3:169–182.

Shannon, C. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, 27.

Shen, L., Xu, J., and Weischedel, R. (2010). String-to-dependency statistical machine translation. *Computational Linguistics*, 36(4):649–671.

Shieber, S. (2006). Unifying synchronous tree-adjoining grammars and tree transducers via bimorphisms. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics.

Shieber, S. M. (1985). Evidence against the context-freeness of natural language. *Linguistics and Philosophy*, 8:333–343.

Shieber, S. M. (1994). Restricting the weak generative capacity of Synchronous Tree-Adjoining Grammars. *Computational Intelligence*, 10(4):371–385.

Shieber, S. M. (2004). Synchronous grammars as tree transducers. In *Proceedings of the Seventh International Workshop on Tree Adjoining Grammar and Related Formalisms (TAG+7)*, pages 88–95.

Shieber, S. M. (2007). Probabilistic Synchronous Tree-Adjoining Grammars for machine translation: The argument from bilingual dictionaries. In *Proceedings of the NAACL-HLT 2007/AMTA Workshop on Syntax and Structure in Statistical Translation*, pages 88–95. Association for Computational Linguistics.

Shieber, S. M. and Schabes, Y. (1990). Synchronous Tree-Adjoining Grammars. In *Proceedings of the 13th Conference on Computational Linguistics*, pages 253–258. Association for Computational Linguistics.

Shieber, S. M., Schabes, Y., and Pereira, F. C. N. (1995). Principles and implementation of deductive parsing. *Journal of Logic Programming*, 24(1&2):3–36.

Sikkel, K. (1997). *Parsing Schemata*. Texts in Theoretical Computer Science. Springer, Berlin, Heidelberg, New York.

Simard, M., Cancedda, N., Cavestro, B., Dymetman, M., Gaussier, E., Goutte, C., Yamada, K., Langlais, P., and Mauser, A. (2005). Translating with non-contiguous phrases. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 755–762.

Skut, W., Krenn, B., Brants, T., and Uszkoreit, H. (1997). An annotation scheme for free word order languages. In *Proceedings of the 5th Applied Natural Language Processing Conference*, pages 88–95, Washington, DC.

Smith, N. A. and Johnson, M. (2007). Weighted and probabilistic context-free grammars are equally expressive. *Computational Linguistics*, 33(4):477–491.

Søgaard, A. (2008a). Learning context-sensitive synchronous rules. In *Proceedings of EAMT'08*, pages 168–173.

Søgaard, A. (2008b). Range concatenation grammars for translation. In *Proceedings of Coling 2008: Companion volume: Posters*, pages 103–106.

Søgaard, A. (2009). On the complexity of alignment problems in two synchronous grammar formalisms. In *Proceedings of the Third Workshop on Syntax and Structure in Statistical Translation (SSST-3) at NAACL HLT 2009*, pages 60–68. Association for Computational Linguistics.

Søgaard, A. (2010). Can inversion transduction grammars generate hand alignments? In *Proccedings of the 14th Annual Conference of the European Association for Machine Translation (EAMT)*.

Søgaard, A. (2011). A $\mathcal{O}(|G|n^6)$ time extension of inversion transduction grammars. *Machine Translation*, 25(4):291–315.

Søgaard, A. and Kuhn, J. (2009). Empirical lower bounds on aligment error rates in syntax-based machine translation. In *Proceedings of the Third Workshop on Syntax and Structure in Statistical Translation (SSST-3) at NAACL HLT 2009*, pages 19–27. Association for Computational Linguistics.

Søgaard, A. and Wu, D. (2009). Empirical lower bounds on translation unit error rate for the full class of inversion transduction grammars. In *Proceedings of the 11th International Conference on Parsing Technologies (IWPT'09)*, pages 33–36. Association for Computational Linguistics.

Sudhoff, S. (2010). *Focus Particles in German: Syntax, Prosody, and Information Structure*. Benjamins.

Sun, J., Zhang, M., and Tan, C. L. (2009). A non-contiguous tree sequence alignment-based model for statistical machine translation. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 914–922. Association for Computational Linguistics.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems (NIPS 2014)*, pages 3104–3112.

Telljohann, H., Hinrichs, E. W., Kübler, S., Zinsmeister, H., and Beck, K. (2012). Stylebook for the Tübingen Treebank of Written German (TüBa-D/Z). Technical report, Seminar für Sprachwissenschaft, Universität Tübingen.

Tillmann, C. (2004). A unigram orientation model for statistical machine translation. In *Proceedings of HLT-NAACL 2004: Short Papers*, pages 101–104. Association for Computational Linguistics.

van Cranenburgh, A. (2012). Efficient parsing with Linear Context-Free Rewriting Systems. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 460–470. Association for Computational Linguistics.

van Cranenburgh, A. and Bod, R. (2013). Discontinuous parsing with an efficient and accurate DOP model. In *Proceedings of the International Conference on Parsing Technologies (IWPT 2013)*.

van Cranenburgh, A., Scha, R., and Bod, R. (2016). Data-oriented parsing with discontinuous constituents and function tags. *Journal of Language Modelling*, 4(1):57–111.

van Cranenburgh, A., Scha, R., and Sangati, F. (2011). Discontinuous data-oriented parsing: A mildly context-sensitive all-fragments grammar. In *Proceedings of the Second Workshop on Statistical Parsing of Morphologically Rich Languages (SPMRL 2011)*, pages 34–44, Dublin, Ireland. Association for Computational Linguistics.

Versley, Y. (2014). Experiments with easy-first nonprojective constituent parsing. In *Proceedings of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages*, pages 39–53, Dublin, Ireland.

Vijay-Shanker, K., Weir, D., and Joshi, A. K. (1987). Characterizing structural descriptions used by various formalisms. In *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics*.

Villemonte de la Clergerie, E. (2002). Parsing Mildly Context-Sensitive Languages with Thread Automata. In *Proceedings of COLING 2002: The 19th International Conference on Computational Linguistics*, Taipei, Taiwan.

Viterbi, A. J. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269.

Vičič, J. and Brodnik, A. (2008). Parse tree based machine translation for less-used languages. *Metodološki Zvezki*, 5(1):65–80.

Wang, C., Collins, M., and Koehn, P. (2007). Chinese syntactic reordering for statistical machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 737–745.

Weir, D. (1988). *Characterizing Mildly Context-Sensitive Grammar Formalisms*. PhD thesis, University of Pennsylvania, Philadelphia, PA.

Wellington, B., Waxmonsky, S., and Melamed, I. D. (2006). Empirical lower bounds on the complexity of translational equivalence. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 977–984, Sydney, Australia. Association for Computational Linguistics.

Wetherell, C. S. (1980). Probabilistic languages: A review and some open questions. *ACM Computing Surveys*, 12(4):361–379.

Williams, P., Sennrich, R., Post, M., and Koehn, P. (2016). *Syntax-based Statistical Machine Translation*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Wu, D. (1997). Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403.

Xia, F. (2001). *Investigating the Relationship between Grammars and Treebanks for natural languages*. Doctoral dissertation, University of Pennsylvania, Philadelphia, PA.

XTAG Research Group (2001). A Lexicalized Tree Adjoining Grammar for English. Technical report, Institute for Research in Cognitive Science, University of Pennsylvania, Philadelphia, PA.

Yadav, R. K. and Gupta, D. (2010). Annotation guidelines for Hindi-English word alignment. In *Asian Language Processing (IALP)*, pages 293–296.

Yamada, K. and Knight, K. (2002). A decoder for syntax-based statistical MT. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*.

Younger, D. H. (1967). Recognition and parsing of Context-Free Languages in time n^3. *Information and Control*, 10(2):189–208.

Zaidan, O. (2009). Z-mert: A fully configurable open source tool for minimum error rate training of machine translation systems. *The Prague Bulletin of Mathematical Linguistics*, 91:79–88.

Zens, R. and Ney, H. (2003). A comparative study on reordering constraints in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 144–151. Association for Computational Linguistics.

Zhang, H., Gildea, D., and Chiang, D. (2008a). Extracting synchronous grammar rules from word-level alignments in linear time. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 1081–1088. Association for Computational Linguistics.

Zhang, H., Huang, L., Gildea, D., and Knight, K. (2006). Synchronous binarization for machine translation. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 256–263. Association for Computational Linguistics.

Zhang, J., Zhai, F., and Zong, C. (2011). Augmenting string-to-tree translation models with fuzzy use of source-side syntax. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 204–215. Association for Computational Linguistics.

Zhang, M., Jiang, H., Aw, A., Li, H., Tan, C. L., and Li, S. (2008b). A tree sequence alignment-based tree-to-tree translation model. In *Proceedings of ACL-08: HLT*, pages 559–567. Association for Computational Linguistics.

Zhang, M., Jiang, H., Li, H., Aw, A., and Li, S. (2008c). Grammar comparison study for translational equivalence modeling and statistical machine translation. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 1097–1104. Association for Computational Linguistics.

Zollmann, A. and Venugopal, A. (2006). Syntax augmented machine translation via chart parsing. In *Proceedings on the Workshop on Statistical Machine Translation*, pages 138–141. Association for Computational Linguistics.