# SYNTACTIC ANALYSIS IN MACHINE TRANSLATION

Petr Homola


ÚSTAV FORMÁLNÍ
A APLIKOVANÉ LINGVISTIKY

***STUDIES IN COMPUTATIONAL AND THEORETICAL LINGUISTICS***

Petr Homola

# SYNTACTIC ANALYSIS IN MACHINE TRANSLATION

Published by Institute of Formal and Applied Linguistics
as the 6th publication in the series
Studies in Computational and Theoretical Linguistics.



This book has been printed with the support of the project MSM0021620838 of The Ministry of Education of the Czech Republic.



ISBN 978-80-904175-7-1

# Contents

# List of Figures

# List of Tables

# Acknowledgement

I am indebted to my supervisor Vladislav Kuboň for his guidance and support. I am also indebted to many reviewers—native speakers of the languages I have researched—for their feedback and help with evaluation. Finally, I would like to thank to everyone who helped me during my Ph.D. studies.

# 1
# Introduction

Natural language processing (NLP) is a comparatively new and rapidly growing discipline in the borderland of theoretical linguistics on the one side and applied mathematics, especially graph theory and statistics, on the other. Machine translation (MT) is a kind of king's discipline of NLP and there has been long and extensive research in the area of rule-based formalisms as well as of statistical approaches to MT. One subcategory of MT is the translation between related languages which is being researched since the late 1980's of the 20th century. This thesis focuses on MT among Balto-Slavic languages.

## 1.1 The Significance of Machine Translation

The goal of machine translation is to automatically transfer a discourse (in MT usually in written form) from a source language to a target language while preserving its meaning and stylistic characteristics. When building an MT system, a natural requirement is to develop it with as little effort as possible. As the complexity of an MT system depends on the similarity of the source and the target language, the knowledge of different strategies for various degrees of language similarity can minimize the effort and guarantee an acceptable quality.

We mainly focus on Baltic and Slavic languages although most of the discussed aspects are valid in general. The mentioned language family has been chosen since it is an ideal 'playground' due to its typology and different degrees of similarity which allows to investigate MT among related languages in detail. Moreover, for many of these languages linguistic resources (such as morphological analyzers, synthesizers, taggers, corpora etc.) are available, thus it is comparatively easy to perform practical experiments to approve or falsify theoretical hypotheses. Also, the typology of these languages, mainly the extremely free word order at the level of actants, is very interesting from the viewpoint of formal theories as it cannot be directly processed by means of formalisms based on context-free rules. Last but not least, the importance of MT among these languages has grown since the accession of several Baltic and Slavic nations to the European Union.

It is obvious that MT between related languages is generally easier than between, for example, Guaraní and Georgian, but what is still unclear is what we have to focus on in the complex MT process so that we can effectively maximize the translation quality. This thesis attempts to explore the contribution of syntactic analysis to the MT in the context of the Balto-Slavic language family, and our additional experiments

with another language group, Romance, show that most of the conclusions are valid not only for Baltic and Slavic.

## 1.2 Structure of the Thesis

The thesis can be roughly split into three parts. Chapters 2, 3 and 4 define basic notions, give an overview of Baltic and Slavic languages and review older MT systems for related languages. Chapters 5, 6, 7 and 8 focus on the properties of the researched languages and on the implementation of an MT framework for them. Finally, Chapter 9 is dedicated to the statistical part of the framework, the ranker, and to the evaluation of our experiments.

There are many approaches to rule-based NLP such as the categorial grammar, HPSG, LFG etc. Our framework is loosely based on the Lexical Functional Grammar and on the theory behind the Prague linguistic school, which is described, along with the other used theoretical background, in Chapter 2. In Chapter 3, we give an overview of Baltic and Slavic languages and present the most notable facts about them. Chapter 4 gives a brief overview of MT systems for related languages that have been developed in the last decades.

In Chapter 5, we focus on the relationship between Balto-Slavic languages and we identify the various free-rides as well as substantial differences among them which are crucial for MT and NLP in general. Chapter 6 focuses on the most important syntactic features of the Balto-Slavic languages at the shallow and deep level. Chapters 7 and 8 describe the implementation of the partial parser and shallow transfer, respectively.

Chapter 9 is dedicated to the statistical ranker which is crucial to the framework since it is the only module that deals with the non-determinism of all other modules of the framework. Furthermore, we use the most notable methods of automatic evaluation of translation quality to evaluate our framework and to compare it to the shallow-transfer based MT system Apertium.

The concluding chapter provides a broader perspective on the problematics of MT between related languages and summarizes the contribution of the thesis to this particular area of NLP.

# 7

# Partial Parser for Baltic and Slavic Languages

In this chapter, we describe the parser module and grammar architecture for shallow processing of Baltic and Slavic languages.[1]

## 7.1 Tasks of the Parser

There was no syntactic parser in the original system *Česílko*. This module has been added to the translation process to deliver information about the sentence structure to the transfer module so that language specific structural properties could be handled and translated properly. Without the parser, morphological differences have only been considered, which is, of course, not sufficient in general. Hence, the parser provides an add-on value which is supposed to improve the translation. If the source sentence is left untouched by the parser (because it is too short or too complex), the system translates it as if there was no syntactic parsing.

The parser uses a hand-written grammar which consists of a set of context-free rules that are written in a declarative form. The output of the parser is a set of c-forests.[2] It is important to mention that a c-forest does not represent the structure of the sentence as such but a concrete rule application sequence. Before being passed to the transfer module, c-forests are automatically converted to d-forests.[3] Thus the final result of the parser is a d-forest or a set of d-forests if the parsed sentence is ambiguous.

The parser is not supposed to parse whole sentences. Of course, if the syntactic structure of the sentence is quite simple, the result will be one tree (or set of trees) covering the whole sentence. Nevertheless, in most cases, the result is a set of trees which only represent fragments of the sentence. One reason for such behavior may be the non-projectivity which occurs quite often in languages with free word order. But projective sentences also may only be parsed partially since the grammar focuses on the level of noun and prepositional phrases. The coverage of verbal phrases is rather

[1] Of course, the parser can be used for other language families as well, with appropriate grammar rules.

[2] By a forest, we mean a set of constituent trees which represent fragments of the parsed sentence and span it completely.

[3] A d-forest is a set of dependency trees which have been created by contracting the vertical edges of a c-tree.

small, the rules on this level are only meant to capture syntactic constructions which may cause serious problems in the target sequence.

### 7.1.1 The Computational Formalism

We use a transformational formalism which is based on a chart parser similar to Q-Systems, designed and first implemented by Colmerauer (1969). What is very important is the fact that the derivational process is context-free (in the sense of Chomsky's hierarchy) which has the crucial consequence for Slavic languages that it is not capable of dealing with non-projective constructions (at least not directly).

The input of the parser can be morphologically ambiguous. In such a case, the parser tries to use all provided data to construct a complete tree. If it succeeds, all complete trees comprise the result set whereas all input items which are not contained in a complete tree, are discarded.

Theoretically, it would be necessary to parse the whole sentence in order to disambiguate it morphologically. Even then, some words may keep more than one morphological tag (due to case syncretism). In case of shallow parsing only, the morphological ambiguity seems to be one of the most serious problems. The best case scenario would be to get a disambiguated input. Unfortunately, at the moment the only possibility is to use a stochastic tagger which introduces errors and makes it impossible for the parser to recognize some dependencies. As has been shown by Žáčková (2002), it is not possible to disambiguate Czech texts by means of shallow rules only.

## 7.2 Main Principles of Parsing Rules

As usual in unification-based grammars, each rule is associated with a condition (constraint) on feature structures and the rule applies only if this condition is satisfied.

A typical example of a linguistically motivated condition is the agreement of morphological categories between the governor and its dependant. For example, an adjective which depends on a noun has to agree with it in gender, case and number. We understand the term *agreement* in broader sense, i.e., a dependant agrees with its governor if a set of conditions, which are defined for the particular type of syntactic construcion, is satisfied. In most cases, the conditions are simply equivalences of category values, as in the following phrase:

(7.1) *mladší* *sestře*
younger-FEM,SG,DAT sister-FEM,SG,DAT

"to the younger sister" (Cze)

Nevertheless, the condition may be more complicated sometimes, for instance, in Polish noun phrases if the governor is in dual form:

(7.2) *czarnymi* *oczyma*
black-NEUT,PL,INS eye-NEUT,DUAL,INS

"with black eyes" (Pol)

(7.3) *w* *swoim* *ręku*
in his-FEM,SG,LOC hand-FEM,DUAL,LOC

"in own hands" (Pol)

Another example can be found in Russian:

(7.4) *два* *больших* *города*
two-MASC,NOM big-MASC,PL,GEN town-MASC,SG,GEN

"two big cities" (Rus)

Another example concerning non-trivial agreement between subject and verb (possible, for example, in Slovenian):

(7.5) *Slovenci* *volimo…*
Slovenians-MASC,PL,NOM vote-1PL,PRES

"we Slovenians vote for…" (Slo)

Apart from rules used to build syntactic trees, we use some tricks in our parser. The aim of these tricks is to modify the chain graph or to control the parsing process. Two such rules are described in the following subsections.

### 7.2.1 Chain Link (shackle)

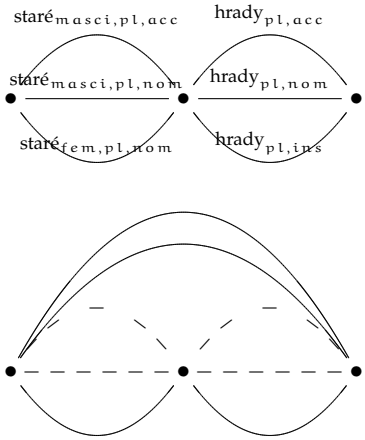



**Figure 7.1:** Example of NP analysis without a shackle

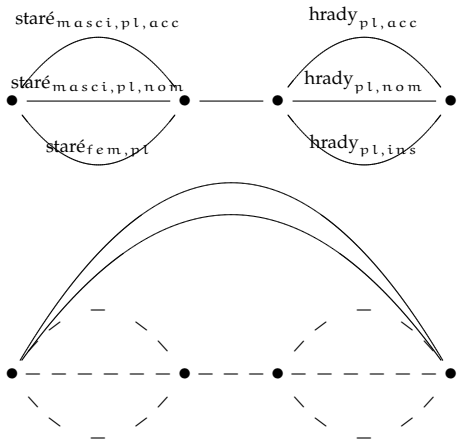



**Figure 7.2:** Example of NP analysis with a shackle

As has been already mentioned, the input of the parser is often morphologically highly ambiguous. One of the tasks of the parser is to disambiguate the sentence (or at least to lower the ambiguity). Let us consider the sentence *Starý hrad se tyčí nad řekou* "The old castle towers over the river". The phrase *starý hrad* is morphologically ambiguous (nominative and accusative). If this phrase has been recognized as the subject of the main verb, we know that the case is nominative in this context. And since there is no other reading where it would be accusative, we want to remove this wrong reading. In fact, it is removed automatically by the algorithm of the parser. But what would have happened if we had the bare phrase *staré hrady*? There are two possible readings (nominative and accusative) which cannot be resolved due to lack of context. Nevertheless, there are still other meanings for each of the words independently (disregarding the dependence between them). In this case, these edges will not be removed although the parser has analyzed the phrase. This is one negative property of the parser framework which has to be solved explicitely. We use a simple workaround: between edges which represent one word of the input sentence, we insert a new edge (*shackle*) that links bunches of edges. If there is at least one analysis which connects two words, the parser marks the shackle as used, i.e., it will be removed during the cleaning phrase (see Section 7.3). As an effect of this, the 'wrong' edges do not lie on a valid path in the multigraph any more and will be deleted as well,

as can be seen in Figure 7.2 (the adjective would have more morphological meanings; for the sake of simplicity, the multigraph contains only one edge with different gender).

It is obvious that if we modify the multigraph by adding 'shackles' between edges labelled with feature structures, we also have to modify all rules accordingly.

### 7.2.2 Elimination of Identical Results

The application of rules to the multigraph is non-deterministic. As a result, the application of several different sequences of rules may lead to identical results, as illustrated in the following example:

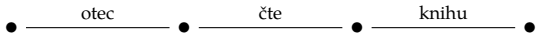



**Figure 7.3:** Example of a sentence with duplicate parses

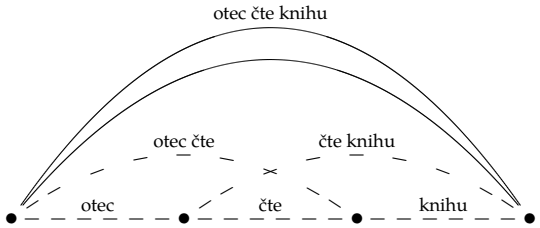



**Figure 7.4:** Chain graph with new edges

There are two possible parses:

1. The rule identifying direct objects is applied first, the rule identifying subjects is applied afterwards.
2. The rule identifying subjects is applied first, the rule identifying direct objects is applied afterwards.

Theoretically, we would get two edges spanning the whole sentence and labelled with identical syntactic structures (see Figure 7.4). In our implementation of the parser, this kind of duplicity is recognized automatically to avoid exponential explosion.

## 7.3 Multigraph Clean-up and Further Optimization

As long as a rule can be applied to the multigraph, edges are added to it but no existing edge is removed. The new edges represent (are labelled with) intermediary feature structures that may be used in further parsing or be candidates for the final result. Once the multigraph cannot be extended by any rule (according to the used grammar), the intermediary edges need to be discarded from the multigraph since we want only the most complex feature structures to be processed in the transfer phase. This clean-up is somewhat similar to garbage collection in programming languages with automatic memory management.

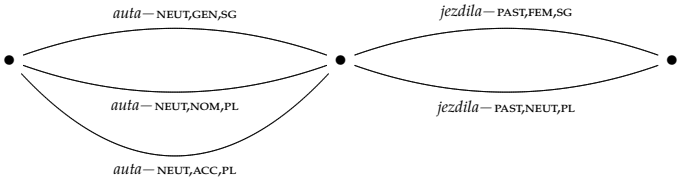
As an example, let us consider the following Czech verb phrase as the input of the parser:

(7.6) *auta* *jezdila*
cars-NEUT,NOM,PL move-LPART,NEUT,PL

"The cars moved/were moving." (Cze)

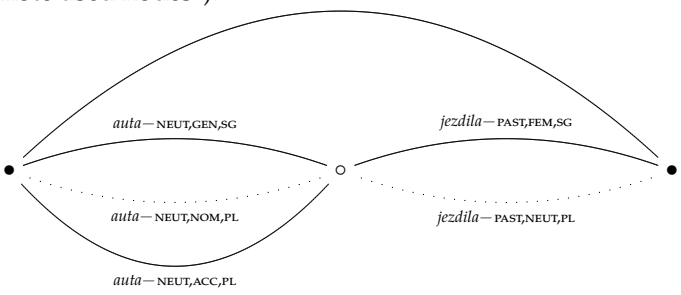
The input of the parser is the following morphologically preprocessed multigraph (the multisets of edges between the same pair of nodes reflect the morphological ambiguity of a word form):

(7.7)




One rule will be applied to this multigraph. Namely the one that attaches a noun in nominative (the subject) to its predicate (a resultative participle in this case). The following multigraph is the result of the syntactic analysis (dotted lines denote used edges, circles denote used nodes[4]):

(7.8)




[4]We define the used node as a node that has at least one used edge to the left and at least one used edge to the right.

Now we need to get rid of all obsolete edges:

1. First of all, we remove all used edges (denoted by dotted lines).
2. We remove all edges which start or end in a used node (i.e., the edges that reflect morphological variants of a used edge which are morphologically misanalyzed in the given context according to the used grammar).
3. For each path $p$ from the initial node to the end node, we calculate the number $u(p)$ of used edges it contains. Then we assign each edge $e$ the score $s(e) = \min_{p \in P} u(p)$. The score for the whole graph is defined as $s = \min_{e \in E} s(e)$. Finally, we remove all edges where $s(e) > s$.[5]

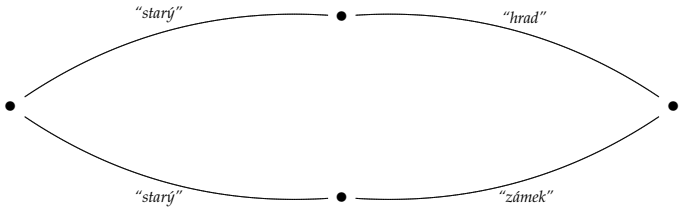
The last step ensures that every edge which remains in the multigraph lies on a path from the initial node to the end node. The resulting graph will be processed by the transfer module and at the same time, all complex feature structures (that represent syntactic trees) are syntactically synthesized (the transfer is described in Chapter 8).

Processing long sentences may result in very large multigraphs with the number of edges growing exponentially. If we had to translate the Russian phrase *старый замок* "old castle" into Czech, the transfer would give the following two features structures:

(7.9) $$\begin{bmatrix} \text{"замок"} \\ \text{ADJ} \quad [\text{"старый"}] \end{bmatrix} \rightarrow \left\{ \begin{bmatrix} \text{"hrad"} \\ \text{ADJ} \quad [\text{"starý"}] \end{bmatrix}, \begin{bmatrix} \text{"zámek"} \\ \text{ADJ} \quad [\text{"starý"}] \end{bmatrix} \right\}$$

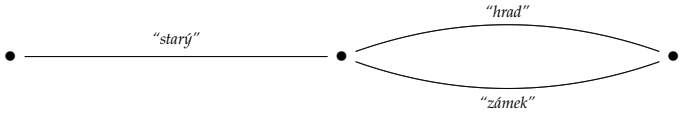
The syntactically synthesized multigraph would be as follows:

(7.10)




As the two edges with the feature structure for the adjective *starý* are identical, we can optimize the spatial complexity of the multigraph by contracting identical edges that have at least one common node. For the discussed example, we would get:

(7.11)




We call this process *compacting* the multigraph. It is obvious that in complex multigraphs, the number of edges can be lowered significantly. Immediately before mor-

[5] If there is at least one path from the initial node to the end node consisting only from unused edges then the algorithm is equal to the one described in (Colmerauer, 1969), i.e., all used edges are deleted as well as edges that do not belong to a path from the initial node to the end node.

phological synthesis, the optimization can be even more efficient if we do not contract only edges with identical feature structures but also with identical surface form in the target language (since there is an extensive syncretism in Slavic laguages).

## 7.4 Using the Parser in a Production Environment

The parser described in the previous sections of this chapter is written in a high-level language (Objective-C) which is more comfortable for the developer to use since the focus lies on linguistics. For grammar development and testing, the performance and resource consumption of the compiled code is not an issue. However, the performance is important for the processing of large texts while the resource consumption (the memory footprint) is crucial for the use of the parser on resource-restricted devices such as PDAs and smartphones. In this section, we briefly discuss a possible optimization of the parser.

We have tried to optimize the parsing process in the way that the rules are indexed by type signature, i.e., the concatenation of type names of all feature structures on the left-hand side of the rule. This optimization saved approximately 50% of processing time because the parser did not try to apply all rules on each subchain of the graph (only rules taken from the index for the particular subchain were considered to be applicable). Nevertheless, we wanted a much faster optimization and also a lower memory footprint. It turned out that transforming the grammar and the input into the Q-Systems format is a good solution.

The Q-Systems are significantly faster than the FS-based implementation of the parser mainly due to the different data structure used in unification. While the FS-based implementation unifies general feature structures, the Q-Systems use trees, thus the unification is similar to the unification of compound predicates in Prolog which makes it significantly faster.

Feature structures in grammar rules and in the input must meet several conditions in order to be transformable to the Q-Systems format. First of all, they must be typed and each type must be assigned a set of attributes the feature structure can contain. Another condition is that the order of attributes declared for a type is fixed. Finally, variables used as attribute values in feature structures may only contain atomic values or embedded feature structures.

Each feature structure is converted to a tree. The root of the tree is labelled by the type name of the feature structure while the sons of the root correspond to attribute values. The order of these nodes is the same as the order of attributes in the declaration for the particular type and all its supertypes. The structure of the rules remains the same including the 'shackles' (see Section 7.2.1). Attributes declared for a type that are not contained in a feature structure (and thus behave like free variables in Prolog) are represented by unique variables in the corresponding Q-Systems rule. It is obvious that type names and atomic attribute values must conform to the syntactic

rules of the Q-Systems. Variables are directly converted to tree-like variables in the corresponding tree and they get the same name.[6]

Let us consider the following type declarations (taken from a grammar for named entity recognition):

```
type sign
end

type shortdate
    prototype sign
    atomic day
    atomic month
end

type date
    prototype shortdate
    atomic year
end

type dateshorttime
    prototype date
    atomic hour
    atomic minute
end

type datetime
    prototype dateshorttime
    atomic second
end

type precisetime
    prototype datetime
    atomic millis
end
```

Each type has a unique name and a prototype (i.e., its supertype, except for the most general type "sign"). The type is assigned a list of attributes containing all attributes of its supertype followed by the declared (additional) attributes. The order of the attributes is not significant for the person who is writing a grammar, it is used only for the transformation of the feature structures. It is obvious that the same type declaration must be used to transform the rules and the input.

[6]The used implementation of the Q-Systems allows for using named variables (see below) while the originally Q-Systems designed by Colmerauer (1969) only allowed for indexed variables.

Let us consider the following feature structure of the aforementioned type *date*.

$$\begin{bmatrix} \textit{date} & \\ \text{DAY} & 23 \\ \text{MONTH} & 5 \\ \text{YEAR} & 2008 \end{bmatrix}$$

This feature structure would be automatically translated to the following Q-tree:[7]

```
DATE(23, 5, 2008)
```

If the structure would have the same content but the type *datetime*, it would be transformed to (the identifiers starting with I∗ are variables):

```
DATETIME(23, 5, 2008, I*ANONYMOUS1, I*ANONYMOUS2, I*ANONYMOUS3)
```

Since the attributes HOUR, MINUTE and SECOND are not listed in the feature structure, they are considered to be underspecified and we have to introduce anonymous variables to represent their values so that the unification works correctly. The name of the anonymous variable is generated automatically so that it is unique.

The interpreter of Q-Systems is implemented in C++ and it is equivalent to the original Q-Systems designed by Colmerauer (1969) except for the following extensions:

- The variables can be named, while in the original Q-Systems they could only be indexed. The name must be alphanumeric.
- If a rule has been successfully applied, the interpreter does not add the new subchain to the graph if there already is an identical subchain at the same position.
- The result of the parser is an empty graph if there is no path from the initial node to the end node in the final graph, after all used edges have been removed (the result of the original Q-Systems was the initial graph instead).

We have tested the aforementioned optimization on 1,000 text documents (most of them containing more than 200 words) with a grammar for named entity recognition. The processing time improved from 33 minutes to less than 4 minutes with a ten times smaller memory consumption.

[7] The interpreter of Q-Systems is not case sensitive thus we can use capitals to denote types in the Q-grammar.

# Summary

This thesis explores the contribution of syntactic analysis to the machine translation (MT) between related languages and it also attempts to explore the limits of shallow MT methods. We focus on one group of languages, the Balto-Slavic language family, and one MT architecture, namely hybrid systems with prevalently rule-based modules.

First, we present related work for Slavic, Scandinavian, Turkic, Celtic and Romance languages. We review different approaches of MT between related languages including the MT system for Slavic languages *Česílko* which constitutes the basis of our system.

Second, we suggest a modification of the commonly used shallow-transfer approach. We describe in detail the implementation of the proposed framework, namely the partial parser, shallow transfer and stochastic ranker, and evaluate the improved architecture on three language pairs using several well-known metrics such as WER, BLEU and NIST.

Third, we examine how our architecture behaves if we couple two MT systems to obtain a new translation pair as compared to a simple pipe of two MT language pairs. This experiment enlightens some aspects of the relationship between deterministic and non-deterministic approaches to morphological analysis, parsing and transfer.

In the concluding chapter, we provide a broader perspective on hybrid methods in MT between related languages and finally, we summarize the contribution of the thesis.

# Bibliography


Lars Ahrenberg and Maria Holmqvist. Back to the Future? The Case for English-Swedish Direct Machine Translation. In *Proceedings of Recent Advances in Scandinavian Machine Translation*, Uppsala, Sweden, 2005.

Kemal Altintas and Ilyas Cicekli. A Machine Translation System between a Pair of Closely Related Languages. In *Proceedings of the 17th International Symposium on Computer and Information Sciences (ISCIS 2002)*, pages 192–196, Orlando, Florida, 2002.

Vytautas Ambrazas. *Dabartinės lietuvių kalbos gramatika*. Mokslo ir enciklopedijų leidykla, Vilnius, 1996.

Vytautas Ambrazas, Aleksas Girdenis, Kazys Morkūnas, Algirdas Sabaliauskas, Vincas Urbutis, Adelė Valeckienė, and Aleksandras Vanagas. *Lietuvių kalbos enciklopedija*. Mokslo ir enciklopedijų leidybos institutas, Vilnius, 1999.

Carme Armentano-Oller, Rafael C. Carrasco, Antonio M. Corbí-Bellot, Mikel L. Forcada, Mireia Ginestí-Rosell, Sergio Ortiz-Rojas, Juan Antonio Pérez-Ortiz, Gema Ramírez-Sánchez, Felipe Sánchez-Martínez, and Miriam A. Scalco. Open-source Portuguese-Spanish machine translation. In *Proceedings of the 7th International Workshop on Computational Processing of Written and Spoken Portuguese*, Rio de Janeiro, Brasil, 2006.

Eckhard Bick and Lars Nygaard. Using Danish as a CG Interlingua: A Wide-Coverage Norwegian-English Machine Translation System. In *Proceedings of NODALIDA*, Tartu, Estonia, 2007.

Hadumod Bußmann. *Lexikon der Sprachwissenschaft*. Alfred Kroener Verlag, Stuttgart, 2002.

Alevtina Bémová, Karel Oliva, and Jarmila Panevová. Some Problems of Machine Translation Between Closely Related Languages. In *Proceedings of the 12th conference on Computational linguistics*, volume 1, pages 46–48, Budapest, Hungary, 1988.

Chris Callison-Burch, Miles Osborne, and Philipp Koehn. Re-evaluating the Role of BLEU in Machine Translation Research. In *Proceedings of the 11th conference of the European Chapter of the Association for Computational Linguistics*, pages 249–256, 2006.

Alain Colmerauer. Les systèmes Q ou un formalisme pour analyser et synthétiser des phrases sur ordinateur. Technical report, Mimeo, Montréal, 1969.

Antonio Corbi-Bellot, Mikel Forcada, Sergio Prtiz-Rojas, Juan Antonie Perez/Ortiz, Gema Remirez-Sanchez, Felipe Sanchez Martinez, Inaki Alegria, Aingeru Mayor, and Kepa Sarasola. An Open-Source Shallow-Transfer Machine Translation Engine for the Romance Languages of Spain. In *Proceedings of the 10th Conference of the European Association for Machine Translation*, Budapest, 2005.

Łukasz Dębowski, Jan Hajič, and Vladislav Kuboň. Testing the limits — adding a new language to an MT system. *Prague Bulletin of Mathematical Linguistics*, 78, 2002.

George Doddington. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the ARPA Workshop on Human Language Technology*, 2002.

Helge Dyvik. Exploiting Structural Similarities in Machine Translation. *Computers and Humanities*, 28:225–245, 1995.

Tomaž Erjavec. MULTEXT-East Version 3: Multilingual Morphosyntactic Specifications, Lexicons and Corpora. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation*, Paris, 2004.

Hans-Werner Eroms. *Syntax der deutschen Sprache*. Walter de Gruyter, Berlin, 2000.

Stefano Federici, Simonetta Montemagni, and Vito Pirrelli. Shallow Parsing and Text Chunking: A View On Underspecification in Syntax. In *Workshop on Robust Parsing, 8th ESSLLI*, pages 35–44, 1996.

M. Forcada, A. Garrido, R. Canals, A. Iturraspe, S. Montserrat-Buendia, A. Esteve, S. Ortiz Rojas, H. Pastor, and P.M. Pérez. The Spanish-Catalan machine translation system interNOSTRUM. *0922-6567 - Machine Translation*, VIII:73–76, 2001.

Berthold Forssman. *Lettische Grammatik*. Verlag J.H. Roell, Dettelbach, 2001.

Victor Friedman. *Macedonian*. SEELRC, 2001.

L. T. F. Gamut. Logic, language and meaning 2: Intensional logic and logical grammar. *University of Chicago Press, Chicago*, 1991.

Jan Hajič. An MT System Between Closely Related Languages. In *Proceedings of the third conference of the European Chapter of the Association for Computational Linguistics*, pages 113–117, Copenhagen, Denmark, 1987.

Jan Hajič and Vladislav Kuboň. Tagging as a Key to Successful MT. In *Proceedings of the Malý informatický seminář*, Josefův Důl, 2003.

Jan Hajič, Jan Hric, and Vladislav Kuboň. Machine translation of very close languages. In *Proceedings of the 6th Applied Natural Language Processing Conference*, pages 7–12, Seattle, Washington, USA, 2000.

Jan Hajič, Eva Hajičová, Petr Pajas, Jarmila Panevová, Petr Sgall, and Barbora Vidová-Hladká. Prague depedency treebank 1.0. *CDROM, CAT: LDC2001T10, ISBN 1-58563-212-0*, 2001.

Jan Hajič, Petr Homola, and Vladislav Kuboň. A simple multilingual machine translation system. In *Proceedings of the MT Summit IX*, New Orleans, 2003.

Petr Homola and Vladislav Kuboň. Improving machine translation between closely related Romance languages. In *Proceedings of the EAMT*, Hamburg, 2008.

Petr Homola and Erika Rimkutė. Artimų kalbų mašininis vertimas. *Kalbų studijos*, 6:77–81, 2004.

Pětr Janaš. *Niedersorbische Grammatik*. Domowina-Verlag, Bautzen, 1976.

Frederick Jelinek. *Statistical methods for speech recognition*. Massachusetts Institute of Teechnology, 1997.

Ronald M. Kaplan and Joan Bresnan. Lexical-Functional Grammar: A formal system for grammatical representation. In Joan Bresnan, editor, *Mental Representation of Grammatical Relations*. MIT Press, Cambridge, 1982.

Lauri Kartunnen. D-PATR: A development environment for Unification-based Grammars. In *Proceedings of Coling*, pages 74–80, 1986.

Natalia Klyueva and Ondřej Bojar. UMC 0.1: Czech-Russian-English Multilingual Corpus. In *Proceedings of the Conference "Korpusnaja lingvistika — 2008"*, pages 188–195, Sankt-Peterburg, Russia, 2008. ISBN 978-5-288-04769-5.

Blaže Koneski. *Историја на македонскиот јазик [History of the Macedonian languages]*. Kočo Racin, Skopje, 1965.

Vladislav Kuboň. Problems of robust parsing of Czech. *Ph.D. thesis, Institute of Formal and Applied Linguistics, Faculty of Mathematics and Physics, Charles University, Praha*, 2001.

Jerzy Kuryłowicz. Le problème du classement des cas. *Bulletin de la Société Polonaise de Linguistique IX*, pages 20–43, 1949.

James E. Lavine. Subject properties and ergativity in North Russian and Lithuanian. In Gerbert Coats Katarzyna Dziwirek and Cynthia Vakareliyska, editors, *Formal Approaches to Slavic Linguistics 7*, pages 307–328. Michigan Slavic Publications, 1999.

James E. Lavine. The morphosyntax of Polish and Ukrainian -no/-to. *Journal of Slavic Linguistics*, 13(1):75–117, 2005.

Dmitry Levinson. Aspect in Negative Imperatives and Genitive of Negation: A Unified Analysis of Two Phenomena in Russian. 2005. URL `http://www.stanford.edu/~dmitryle/Levinson2005_ImperfectiveAndGenitiveOfNegation.pdf`.

Willy Mayerthaler, Günther Fliedl, and Christian Winkler. *Lexikon der natürlichkeitstheoretischen Syntax und Morphosyntax*. Stauffenberg Verlag, Tübingen, 1998.

Jarmila Panevová. *Formy a funkce ve stavbě české věty*. Academia, Praha, 1980.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, 2001.

Ernesta Račienė. Zur Frage des Aspekts und der Aktionsarten im Deutschen und Litauischen. *Žmogus ir žodis, Vilniaus pedagoginis universitetas*, 1, 1999.

Kevin P. Scannell. Machine translation for closely related language pairs. In *Proceedings of the Workshop Strategies for developing machine translation for minority languages*, Genoa, Italy, 2006.

Petr Sgall, Eva Hajičová, and Eva Buráňová. *Aktuální členění věty v češtině*. Academia, Praha, 1980.

Petr Sgall, Eva Hajičová, and Jarmila Panevová. *The Meaning of the Sentence in Its Semantic and Pragmatic Aspects*. D. Reider Publishing Company, 1986.

Manfred Starosta. *Niedersorbisch schnell und intensiv 2*. Ludowe nakładnistwo Domowina, Bautzen, 1992.

Ralf Steinberger, Bruno Pouliquen, Anna Widiger, Camelia Ignat, Tomaž Erjavec, Dan Tufis, and Daniel Varga. The JRC-Acquis: A multilingual aligned parallel corpus with 20+ languages. Sep 2006. URL `http://arxiv.org/abs/cs.CL/0609058`.

Charles E. Townsend and Laura A. Janda. *Gemeinslavisch und Slavisch im Vergleich. Einführung in die Entwicklung von Phonologie und Flexion*. Verlag Otto Sagner, München, 2003.

Valentin I. Trubinskij. *Очерки русского диалектного синтаксиса*. Izdatěľstvo Leningradskogo universiteta, Leningrad, 1984.

Nikolaos H. Trunte. *Altkirchenslavisch*, volume 1 of *Словенскъи языкъ. Ein praktisches Lehrbuch des Kirchenslavischen in 30 Lektionen. Zugleich eine Einführung in die slavische Philologie*. Verlag Otto Sagner, München, 2005.

Aloyzas Vidugiris. *Zietelos lietuvių šnekta*. Presvika, Vilnius, 2004.

Jernej Vičič. Rapid development of data for shallow transfer RBMT translation systems for highly inflective languages. In *Proceedings of 6th Language Technologies Conference 2008*, 2008.

Rasuolė Vladarskienė. Iliatyvo vartojimas dabartinėje lietuvių kalboje. *Kalbos kultūra*, 76:47–57, 2003.

Eva Žáčková. *Parciální syntaktická analýza (češtiny)*. PhD thesis, Fakulta informatiky Masarykovy univerzity, Brno, 2002.

Harald Weydt and Alicja Kaźmierczak. Gibt es ein Perfekt im modernen Polnisch? *Linguistik online*, 4(3), 1999.

Dan Zeman. Neprojektivity v Pražském závislostním korpusu (PDT). Technical Report 22, Univerzita Karlova, Praha, 2004.

Zigmas Zinkevičius. *Lietuvių kalbos dialektologija*. Mokslo ir enciklopedijų leidybos institutas, Vilnius, 1994.

Zigmas Zinkevičius. *The history of the Lithuanian language*. Mokslo ir enciklopedijų leidybos institutas, Vilnius, 1998.

# Index