Interoperability and Rapid Bootstrapping of Morphological Parsing and Annotation Automata

Damir Ćavar,[1] **Ivo-Pavao Jazbec,**[2] **Siniša Runjaić**[2]

[1]University of Zadar
Linguistics Department
Obala kralja Petra Krešimira IV. 2, 23000 Zadar, Croatia
dcavar@unizd.hr

[2]Institute of Croatian Language and Linguistics
Ul. Republike Austrije 16, 10000 Zagreb, Croatia
{ipjazbec,srunjaic}@ihjj.hr

Abstract

We discuss the design and development of a finite state transducer for morphological segmentation, annotation, and lemmatization that allows for merging of three major functionalities into one high-performance monolithic automaton. It is designed to be flexible, extensible, and applicable to any language that allows for purely morphotactic modeling on the lexical level of morphological structure. The annotation schema used in an initial Croatian language model is a direct mapping from the GOLD ontology of linguistic concepts and features, which increases the potential for interoperability, but also opens up advanced possibilities for a DL-based post-processing.

1. Introduction

Quantitative and qualitative information about morphological properties of languages is hard to come by. For many languages information as for example contained in CELEX (Burnage, 1990) is not available. For many research questions, most of the available information about distributional properties of morphemes and their feature makeup is not sufficient.

Corpus annotations tend to be lexeme and word-form oriented, providing part-of-speech (PoS) tags for tokens in the corpus, rather than segmentation of word-forms into morphemes and allomorphs with their particular feature annotation. The notion of *morphological information* is used inconsistently in the literature, e.g. associated with lexeme and PoS information only. The documented Croatian morphological lexicon (Oliver and Tadić, 2004) for example does not provide information about the morphological structure and specific feature annotations of single morphemes, but rather word-forms and lexemes with PoS-annotation.

On the other hand, specific research questions, require detailed morphological analyses of lexical tokens in a corpus. In our particular case, the Croatian Language Corpus (Brozović-Rončević and Ćavar, 2008), as one of our major data sources needs to be annotated for subsequent analysis.

To be more precise, our understanding of a morphological lexicon and morphological corpus annotation, i.e. our desired type of information, consists of parses of lexemes on the morphological level, and explicit feature bundles associated with each single morpheme or allomorph, as shown in table 1 for the word *popijemo* (Croatian, "to drink (up)").

token	*popijemo*		
parse	*po*	*pije*	*mo*
	stem		suffix
	prefix	root	inflectional
	aspect	verb	1^{st}
	perfective	transitive	plural

Table 1: Morphological parse example

We refrain from providing a hierarchical tree structure for morpheme relations, although it might be useful to reveal scope ambiguities of semantic properties.[1] Parses with just a linear segmentation, including a quasi-hierarchical dependency with for example the prefix and root being contained in the stem, as shown in table 1, provide enough valuable information for some advanced research questions.

For specific research purposes, in particular questions about morphological ambiguity load, the desired morphological analysis of a corpus should provide all possible parses and feature bundles for each morpheme, be it a morphological root or affix (potentially a null-affix), as partially shown in table 2.

Looking at textual data from Croatian synchronic and diachronic dialects and variants, as well as the standard language, we are facing various problems, as for example:

– Different orthography standards have been, and are still used.

– The lexical environment is not static, with lexical

We are grateful to Thomas Hanneforth, Adrian Thurston, and Darko Veberič for their comments and response related to the code and technical realization, and to our colleagues at the IHJJ for lexical material and advice. Furthermore, we thank several anonymous reviewers for helpful hints and comments.

[1]An elegant description of the ambiguity of the word *unlockable* for example could be based on a hierarchical representation as [*un* [*lock able*]] or [[*un lock*] *able*]. A flat and linear segmentation representation is not as intuitive.

token	*kocka*	
parse 1	*kocka*	Ø
	root	suffix
	noun	singular
	feminine	nominative
parse 2	*kocka*	Ø
	root	suffix
	verb	3^{rd}
	intransitive	singular
	denominal	
...		

Table 2: Morphologically ambiguous parse example

items emerging and disappearing, their semantic properties changing etc.

– Lexical changes occurred, some might have affected the morphological makeup of individual word-forms (including changes in paradigms), some might be related to different feature bundles associated with them.

Since various domains of lexical and morphological properties and features in our particular case are still subject to ongoing research, the set of features is necessarily open and unspecified from the outset. We expect in particular semantic properties, new feature types that result from linguistic conceptual necessities, or marking of linguistic origin and cultural background to emerge during future studies. Also, identifiers for named entities appear to be natural extensions of the feature set. The annotations should be extensible with respect to these properties. Any extension of the morphological feature set should directly be integrated into the annotations of an entire corpus.

Once the morphological segmentation is available, further annotations and analyses can be incorporated in a trivial way. For example, the generation of lemmata for segmented word-forms can be achieved by appending the canonical inflectional suffix to the identified base, and potentially applying the necessary allomorphic change to the root. Furthermore, for establishing associations of word-forms to semantic fields, i.e. identifying the semantic root of a complex word-form, the lemma of the root provides a useful additional annotation information. For most Slavic and Germanic languages the rightmost root in a word-form is the semantic head of a complex morpheme. Thus, a general strategy for the identification of the root-lemma is to pick the rightmost root morpheme and append to it the canonical inflectional suffix.

This strategy is on the one hand attractive, because by identification of the lexical root one can relate complex word-forms to their underlying semantic concept. On the other hand, many word-forms are not strictly compositional to allow this. Consider the word-form *neprijatelj* ("enemy"):

– Surface form: *neprijatelj*

– Segmentation: *ne* <prefix> – *prijatelj* <root>

– Root lemma: *prijatelj*

– Base lemma: *neprijatelj*

The root-lemma in this case is not a direct derivation or composition of the negation operator and the lexical root. The meaning of ¬`prijatelj` is not `neprijatelj`, although ¬`prijatelj` might be one of many conceptually true properties of `neprijatelj`. Nevertheless, providing the root-lemma even in this case allows for associations of lexical items with fundamental concepts and word classes, in particular, if derivational morphology is involved in the word-formation process of a surface form. Our goal is thus to annotate the corpus for both lemma types, i.e. the root- and the base-lemma. The latter is achieved by inclusion of all prefixes in the lemma formation rule that are part of the morphological base.

To achieve this, a software solution is necessary. Manual annotation of large corpora is not feasible, it would lack consistency on the large scale, i.e. it is error prone. A software solution allows for systematic and consistent annotation of large corpora. Errors in the annotation should be systematically corrected in the grammar and formalism of some algorithmic annotation solution, rather than corrected manually in a corpus.

To sum up, a software solution should have the following properties:

a. It should provide parses of word-forms into morphemes.

b. It should provide annotations (feature bundles) for each single morpheme.

c. It should be extensible, wrt. the feature-bundles associated to morphemes, as well as to the list of morphemes as such.

A software solution, however, to our knowledge, does not exist for the languages we are interested in.

The specification might look like an all-in-one device for every purpose. However, its development appears to be feasible, with a very simple, and nevertheless efficient technical solution, i.e. finite state transducers (Berstel, 1979; Berstel and Reutenauer, 1988). In the following we describe the algorithmic specification of CroMo, the morphological parser, annotator and lemmatizer, developed for the Croatian standard, and synchronic and diachronic variants.

1.1. Previous approaches

Finite state methods for computational modeling of natural language morphology are wide-spread and well understood. Various commercial and open-source FSA-based development environments, libraries and tools exist for modeling of natural language morphology. A detailed discussion of their properties and application for various languages would be beyond the scope of this article. Some overview can be found in recent literature, e.g. (Sproat, 2000; Beesley and Karttunen, 2003; Roark and Sproat, 2007), further links to literature and implementations can be found in the context of the OpenFst library (Allauzen et al., 2007).

For Croatian there are various descriptions of the formalization and computational modeling of morphology in terms of finite state methods (Tadić, 1994; Lopina, 1999). However, an implemented testable application is not available.

Some solutions that have been implemented for example for German come close to the system requirements specified above. The SMOR (Schmid et al., 2004) and Mor-

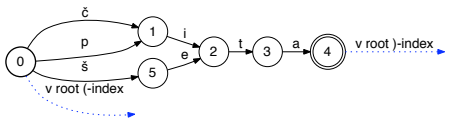
phisto (Zielinski and Simon, 2008) systems partially represent such a type of computational morphology application. An almost complete overlap of features and properties can be found in the implementation of the German morphology as described in the TAGH (Geyken and Hanneforth, 2005) system.

2. FST for morphological segmentation

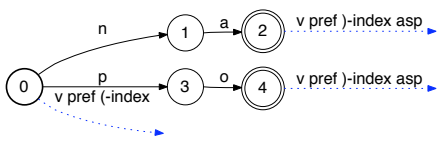
For various reasons, we decide to stick to the approach and implementation strategy of TAGH, while we apply our own experimental libraries and development environment.[2] Following the TAGH-approach (Geyken and Hanneforth, 2005), we model Croatian morphology by referring exclusively to morphotactic regularities, using morpheme and allomorph sets and regular morphological rules, such that a deterministic finite state transducer (FST) can be generated.

The initial modeling step consists of grouping of morphemes. While each application might involve different considerations about how morphemes have to be grouped, in general if should be based on criteria like (a.) having the same feature specification, and (b.) being subject to the same morphological rules.

In CroMo each morpheme group represents one deterministic and acyclic finite state transducer (DFST). The design is similar to the Mealy (Mealy, 1955) or Moore machine (Black, 2004). Every morpheme DFST emits on entry a tuple of the byte-offset in the input string, and the feature bundle that is associated with the DFSA path. In every final state the DFST emits the same tuple. This way morphemes are marked with a start and end index, as well as the corresponding feature bundle, representing the desired annotation. The following graph shows a simplified example of an acyclic DFST for verbal roots:



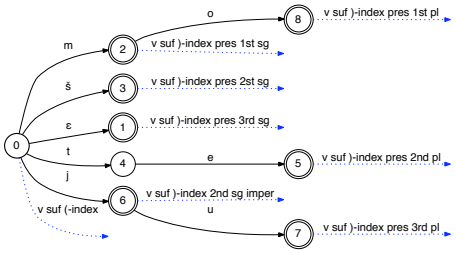
In the same way verbal (e.g. aspectual) prefixes are organized into acyclic DFSTs, as shown in the following graph:



The verbal inflectional paradigm is organized in the same way. Since the model is based on purely morphotactic distributional regularities, potential phonological phenomena are expressed using exclusively allomorphic variations.

[2]The automata and grammar definitions we use are compatible with several existing systems and libraries.

The following graph shows a simplified network for verbal suffixes:

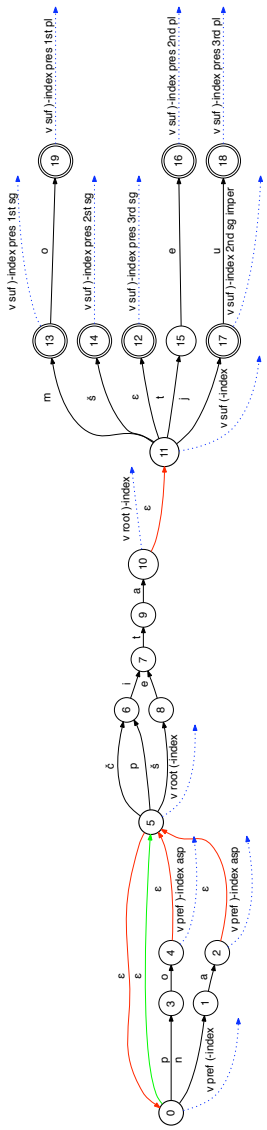


Once all morphemes are grouped into DFSTs, and the appropriate emission symbols (the annotations) are assigned to each entry and final state of the DFST, each morpheme group is assigned an arbitrary variable name, which is used in the definition of rules. A rule that makes use of the automata above could be defined as follows:

```
vAspectPref* . vAtiRoots . vInflSuf
```

This rule describes the concatenation of the verbal root DFST with the DFST for the verbal inflectional paradigm, using common regular expression notation. In this case we use the regular expression syntax as defined for the Ragel (Thurston, 2006) state machine compiler. Additionally, the prefixes are defined as optional and potentially recursive prefixes concatenated with the verbal root DFST. This definition generates a cyclic[3] deterministic transducer with every final state of beginning and intermediate sub-DFSTs into a non-final state, linked to the initial state of the concatenated DFSTs via an ϵ-transmission, as shown in the following graph:

[3]Cyclicality in this particular case leads to more compact automata. In principle, the depth of recursion of such prefixes could be limited (empirically and formally), and formalized using the appropriate regular expression syntax.



Such a DFST emits a tuple containing the byte-offset and the corresponding annotation symbols at the initial state, and at each morpheme boundary (former initial and final states of the sub-DFSTs).

Using this approach, all lexical classes are defined as complex (potentially cyclic) DFSTs, and combined, together with the closed class items, as one monolithic DFST.

The advantage of such a representation is not only that the resulting morphological representation is compressed, but also that it is processed in linear time, with the identification of morpheme boundaries and corresponding feature bundles being restricted by contextual rules.

In order to cope with morphological ambiguity, this approach is extended. In principle there are two major approaches to deal with ambiguity, either one has to allow for non-deterministic automata (two different transitions with the same input emit a different output tuple), or ambiguity is mapped on the emission of multiple annotation tuples. In the case of CroMo, the latter option is used in the modeling. Every emission is a tuple of length 0 to n, such that e.g. orthographically ambiguous nominal suffixes like *a* (genitive singular or plural) are modeled as a single transition in a DFST with the final state emitting two annotation tuples that contain the specific case and number features.

2.1. Interoperability and annotation standard

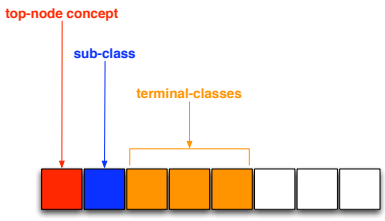
Current language resources face a serious problem, related to issues of interoperability and annotation compatibility. Various different tag-sets are used for particular languages, and some of those tend not to be straight-forward compatible. In the same way, linguistic annotation tools do not necessarily make use of some standardized tag-set, and such a tag-set actually does not even exist.

For our purposes here we decided to offer maximal interoperability in the resulting corpus annotation, as well as in the annotation tool as such, one that is maximally compatible with existing tag-sets, as language specific as necessary, and at the same time maximally extensible. The General Ontology for Linguistic Description (GOLD) (Farrar and Langendoen, 2003; Farrar et al., 2002; Farrar, 2003) *was originally envisioned as a solution to the problem of resolving disparate markup schemes for linguistic data*. GOLD specifies basic linguistic concepts and their interrelations, and can be used, to a certain extent, as a description logic for linguistic annotation.

We make use of three core concept classes in GOLD, and the necessary sub-concepts, i.e. MorphoSemanticProperty, MorphosyntacticProperty, and LinguisticExpression. The concepts defined therein relate to the notions that are expected to be emitted by CroMo, i.e. morphological properties of morphemes (e.g. prefix, suffix, root), morphosyntactic properties (e.g. case, number), and morphosemantic properties (e.g. aspect, mood, tense).

By using the labels for concepts as defined in GOLD, we should be able to maintain maximal compatibility with other existing tag-sets. While the logic of GOLD would burden a morphological parsing algorithm, the reference to the concepts doesn't seem problematic. Representing the concepts as pure emission strings associated to the emission states, as discussed above, might decrease memory and performance benefits of a DFST-based analyzer. To maximize the performance, the GOLD-concepts and relations are mapped on a bit-vector. Encoding of the relevant concepts can be achieved with bit-vectors of less than 64 bit.

The mapping defines constants that correspond to bit-masks that are pre-compiled into the DFST. The bit-mask for example for `Genitive` might be defined as one that corresponds to set first and second bits of the terminal-class bit-field, plus the corresponding bits that indicate that the sub-class `CaseProperty` is set, as well as the bit for the corresponding top-node class `MorphosyntacticProperty`, as shown in the following graphic:



In a limited way, via definitions of constants and mapping of linguistic annotation in the morpheme dictionaries, one can maintain implicatures and inheritance relations, as defined in the ontology, via bit-vector representations and appropriate bit-masks.

For the morphological analyzer this does not imply any additional processing load, i.e. the emission tuples consist of bit-vectors in form of 4-byte numerical integer values. Converting the emission tuples (i.e. individual bit-vectors) into literal string representations can be achieved efficiently, once an input string is analyzed completely.

2.2. Implementation

CroMo consists of two sets of code-bases. The first component converts a lexical base into a formal automaton definition. The second compiles together with the automaton definitions into a binary application.

The lexical base is kept either in database tables, spreadsheets, or textual form. The different formats allow us to maintain a minimally invasive lexical coding approach. Linguists or lexicologists are not required to learn a formal language for DFST definitions. Furthermore, they are free to use their individual way of annotation, being guided by GOLD concepts, but free to define their own, should these not be part of GOLD. CroMo provides guidelines for the data-format, but also the possibility to use individual scripts for data conversion and annotation mappings.

The individual morpheme lists, annotations and rule definitions are compiled into Ragel (Thurston, 2006) automata definitions, as described above. Besides rules that are related to concrete morpheme lists and the corresponding DFSTs, there are also guessing rules that define general properties of nouns, verbs and adjectives. The features that are used, be they specified in GOLD or not, are mapped on bit-vectors, and C-header files with the constant literal and bit-vector mask definitions are generated.

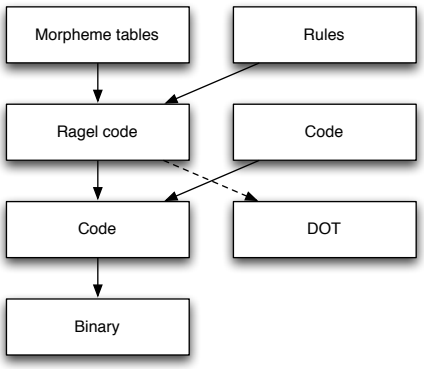
Ragel generates a monolithic DFST as C-code, using highly efficient C-jump code (`goto`-statements), as well as a DOT-file for visualization of the resulting automaton (using e.g. Graphviz[4]). The generated code is wrapped in a C++ class that handles input and output, and controls the program logic.

In the current version the generation of the root- and the base-lemma is encoded in the emission bit-vector. One byte is reserved to mark the reverse offset for string concatenation, while two bytes are reserved to point to an element in a string array with the corresponding string that needs to be appended. The form *čitamo* would be associated with an offset of -2 and a corresponding suffix *ti*. This solution doesn't match the general paradigm, and is just temporary.

[4]See `http://www.graphviz.org/` for details.

In the next release the output characters of the corresponding lemma will be integrated in the emission of the transducer, associated with each single transition. Thus every emission will be a tuple that contains tuples of output characters and optional annotation bit-vectors.

The following graphic shows the general automaton compilation workflow, implemented in a single shell-command:



CroMo expects a token list as input. Tokens are processed sequentially. For each token, all emitted tuples are collected in a stack. Only matching start- and end-tuples are returned, if there are compatible sub-morpheme analyses that span over the complete input token length.

The significant implementation features that differentiate CroMo from other solutions, are:

1. The code-base is platform independent and free open-source, using only free and open tools like GCC and Ragel for compilation.

2. CroMo doesn't depend on any particular encoding standard. The lexical base can be encoded in any common encoding, since the automaton processing is purely binary (i.e. byte-based). The character encoding of the morpheme definitions and the input tokens should match. In principle it is possible to use multiple different encodings.

The extension of the morphological base is kept trivial, along the lines of the requirements specified above, i.e. the necessity to be able to add newly identified morphemes or paradigms from diachronic and synchronic variants.

3. Evaluation

The evaluation version of CroMo contains approx. 120,000 morphemes in its morpheme-base, using UTF-8 character encoding. The number of strings it can recognize is infinite, due to cyclic sub-automata. Unknown word-forms can be analyzed due to incorporated guessing rules.

For the following evaluation results we used a 2.4 GHz 64-bit Dual-Core CPU. In the evaluation version only a single core is used during runtime of the FST, while both CPU cores are used during compilation.

Compilation of the morphology requires min. 4 GB of RAM using GCC 4.2. This is expected due to the monolithic architecture, and since the Ragel-generated C-code of the transducer gets very large. The compilation process takes less than 5 minutes, using both CPU cores. The resulting binary footprint is less than 5 MB of size.

The final automaton consists of approx. 150,000 transitions and 25,000 states.

We selected randomly 10,000 tokens with an average morpheme length of 2.5 morphemes. CroMo processes in average approx. 50,000 tokens per second (real 10,000 tokens per 150 millisec.), including runtime instantiation in memory, mapping of the analysis bit-vectors to the corresponding string representations, generation of lemmata, and output redirection to a log-file. An extension of the morpheme base has no significant impact on memory instantiation time, neither on the runtime behavior. The memory instantiation can be marginalized for a large processing sample.

CroMo doesn't implement transitional or emission-probabilities, due to missing quantitative information from training data. Once an annotated corpus is available, these weights can trivially be implemented as additional weights in the emission tuple.

A relevant evaluation result is the coefficient of the ratio between all and relevant emissions, i.e. the percentage of relevant (possible) morpheme analyses and all generated ones. Due to certain limitations, we did not perform such an evaluation, neither a recall evaluation on a predefined evaluation corpus. The results of these evaluations, together with the source code, will be made available on CroMo's web site `http://personal.unizd.hr/~dcavar/CroMo/`.

4. Comments

CroMo manifests a highly efficient morphological segmentation and annotation algorithm, with little margin for efficiency improvement in the code base.

The use of GOLD as an annotation standard has been shown to be feasible, on the implementation level. However, GOLD is still under development. Many necessary concepts have not yet been implemented. A benefit of using a DL for annotation has yet to be shown. Once necessary syntactic concepts are specified in GOLD, syntax-based disambiguation would be feasible. GOLD mappings to different tag-sets have still to be established, to fulfill the promise of interoperability and annotation compatibility.

5. References

Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri. 2007. OpenFst: A general and efficient weighted finite-state transducer library. In *Proceedings of the Ninth International Conference on Implementation and Application of Automata, (CIAA 2007)*, pages 11–23. Springer-Verlag.

Kenneth R. Beesley and Lauri Karttunen. 2003. *Finite State Morphology*. CSLI Publications, Stanford, April.

Jean Berstel and Christophe Reutenauer. 1988. *Rational Series and Their Languages*. EaTCS Monographs on Theoretical Computer Science. Springer-Verlag, Berlin, December.

Jean Berstel. 1979. *Transductions and Context-Free Languages*. Teubner Studienbücher, Stuttgart.

Paul E. Black. 2004. Dictionary of algorithms and data structures. Online publication: U.S. National Institute of Standards and Technology, Available from http://www.nist.gov/dads/HTML/mooreMachine.html, December.

Dunja Brozović-Rončević and Damir Ćavar. 2008. Hrvatska jezična riznica kao podloga jezičnim i jezičnopovijesnim istraživanjima hrvatskoga jezika. In *Vidjeti Ohrid*, Hrvatska sveučilišna naklada, pages 173–186, Zagreb. Hrvatsko filološko društvo.

Gavin Burnage. 1990. CELEX - A guide for users. Technical report, Centre for Lexical Information, University of Nijmegen, Nijmegen.

Scott O. Farrar and D. Terence Langendoen. 2003. A linguistic ontology for the semantic web. *Glot International*, 7(3):1–4, March.

Scott O. Farrar, William D. Lewis, and D. Terence Langendoen. 2002. A common ontology for linguistic concepts. In N. Ide and C. Welty, editors, *Semantic Web Meets Language Resources: Papers from the AAAI Workshop*, pages 11–16. AAAI Press, Menlo Park, CA.

Scott O. Farrar. 2003. *An Ontology for Linguistics on the Semantic Web*. Ph.D. thesis, The University of Arizona, Tucson, Arizona.

Alexander Geyken and Thomas Hanneforth. 2005. TAGH: A complete morphology for german based on weighted finite state automata. In Anssi Yli-Jyrä, Lauri Karttunen, and Juhani Karhumäki, editors, *FSMNLP*, volume 4002 of *Lecture Notes in Computer Science*, pages 55–66. Springer, September.

Vjera Lopina. 1999. Strojna obrada imenične morfologije u pisanome hrvatskom jeziku. Ma thesis, Centar za postdiplomske studije Dubrovnik, Dubrovnik, October.

George H. Mealy. 1955. A method for synthesizing sequential circuits. *Bell System Technical Journal*, 34(5):1045—1079, September.

Antoi Oliver and Marko Tadić. 2004. Enlarging the croatian morphological lexicon by automatic lexical acquisition from raw corpora. In *Proceedings of LREC 2004*, volume IV, pages 1259–1262, Lisbon, May. ELRA.

Brian Roark and Richard Sproat. 2007. *Computational Approaches to Syntax and Morphology*. Oxford University Press, Oxford.

Helmut Schmid, Arne Fitschen, and Ulrich Heid. 2004. SMOR: A german computational morphology covering derivation, composition, and inflection. In *Proceedings of the IVth International Conference on Language Resources and Evaluation (LREC 2004)*, pages 1263–1266, Lisbon, Portugal.

Richard Sproat. 2000. *A Computational Theory of Writing Systems*. AT&T Bell Laboratories, New Jersey, July.

Marko Tadić. 1994. *Računalna obradba morfologije hrvatskoga književnog jezika*. doctoral dissertation, Filozofski fakultet Sveučilišta u Zagrebu, Zagreb, Croatia.

Adrian D. Thurston. 2006. Parsing computer languages with an automaton compiled from a single regular expression. In *11th International Conference on Implementation and Application of Automata (CIAA 2006)*, volume 4094 of *Lecture Notes in Computer Science*, pages 285–286, Taipei, Taiwan, August.

Andrea Zielinski and Christian Simon. 2008. Morphisto – an open-source morphological analyzer for german. In *Proceedings of FSMNLP 2008*, Ispra, Italy, September.