Efficient Morphological Parsing with a Weighted Finite State Transducer

Damir Ćavar
University of Zadar, Croatia
E-mail: dcavar@unizd.hr and http://personal.unizd.hr/˜dcavar/

Ivo-Pavao Jazbec and Siniša Runjaić
Institute of Croatian Language and Linguistics, Croatia
E-mail: {ipjazbec,srunjaic}@ihjj.hr and www.ihjj.hr

Keywords: weighted finite state transducer, morphological analysis, croatian

Received: October 31, 2008

This article describes a highly optimized algorithm and implementation of a deterministic weighted finite state transducer for morphological analysis. We show how various functionalities can be integrated into one machine, without sacrificing performance or flexibility, and and still maintaining applicability to various languages. The annotation schema used in this implementation maximizes interoperability and compatibility by using a direct mapping of tags from the GOLD ontology of linguistic concepts and features, providing possible extended processing scenarios.

Povzetek: Opisana je morfološka analiza za hrvaški jezik.

1 Introduction

For the majority of natural languages, the s.c. low density languages, appropriate linguistic data and language processing tools do not exist, neither enough raw language data (e.g. text or audio recordings). For some languages with much higher language resource density the appropriate language technology is missing that would help in creating necessary and valuable quantitative and qualitative linguistic information, essential not just for research purposes. Thus, even languages that do not face the low density problem, still lack crucial resources. For many languages information as for example contained in CELEX [7] is not available. Thus, for the majority of languages the distributional and quantitative models of phonetic, phonemic, morphological and syntactic properties do not exist.

In recent years the amount of available linguistic data was growing. Recordings and transcriptions, dictionaries and textual corpora build the basis for an impressive amount of empirical linguistic research, as well as language technology for various application domains. However, the resources face crucial problems. On the one hand, we see a growing number of specific purpose data, with limited value for a wide range of research and development domains, representing snapshots of a specific state of a language at a specific time, often based on easily available raw data like newspapers or books. Language change and dynamic aspects of languages require permanent creation and adaptation of existing resources. This task cannot be accomplished without the technologies, e.g. adaptive tools or appropriate machine learning algorithms.

Linguistic annotation of corpora is limited in another way. The choice of part-of-speech (PoS) tags is theory driven, and thus in general restricted to a specific view or framework, with a likely limited value for other research and development purposes.

Focusing on morphology as one of the levels of linguistic representation and grammar, corpus annotations tend to be lexeme and word-form oriented, PoS-tags for lexemes in the corpus, rather than segmentation of word-forms into morphemes and allomorphs with their particular feature annotation. Even the notion of *morphological information* is used inconsistently in the literature, e.g. associated exclusively with lexeme and PoS information. Thus existing resources, e.g. the documented Croatian morphological lexicon [14], do not provide information about the morphological structure and specific feature annotations of single morphemes, but rather word-forms and lexemes with PoS-annotation. Specific research questions, on the other hand, require detailed morphological analyses of lexical tokens in a corpus. On the basis of the Croatian Language Corpus [6], as one of our major data sources, needs to be annotated for subsequent analysis.

1.1 Central goals

Our specific goal was the development of a system that parses lexemes into morphological structure. The desired output information, as far as morphology is concerned, requires a morphological lexicon and morphological corpus annotation to include parsed lexemes on the morphological level, with annotations and explicit feature bundles associated with each single morpheme or allomorph, as shown in table 1 for the word *pročitamo* (Croatian, “to read(out)”).

Table 1: Example of a morphological parse

token	*pročitamo*		
parse	*pro*	*čita*	*mo*
	stem		inflectional
	prefix	root	suffix
	aspect	verb	1^{st}
	perfective	transitive	plural
			present

This type of a morphological parse is already simplified. Certain potentially required and theoretically motivated information is excluded. For example a hierarchical tree structure for morpheme relations is not displayed, although it might be useful to reveal scope ambiguities of semantic properties. The shown parse represents just linear segmentations that include a quasi-hierarchical dependency with for example the prefix and root being contained in the stem, as shown in table 1. In general, we expect ambiguities to occur, and in fact we are interested in all possible parses that lead to a complete analysis of a morphological complex word. For research purposes in the first stage we do not intend to disambiguate the parses.

In addition to the morphological parse, the output should ideally also contain information about the lexical lemma (i.e. base-form) and the related root lemma. Once the morphological segmentation is available, the generation of lemmata 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, the root-lemma is generated by picking the rightmost root morpheme and append to it the canonical inflectional suffix. We annotate individual word forms 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.

1.2 Scope and problems

The morphological parse and annotation is supposed to cope with raw language data from various time periods and dialects, i.e. synchronic and diachronic data, focusing for the time being on Croatian. Such textual data is problematic since e.g. different orthography standards have been and are still used. The lexical environment was and is not static, with lexical 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.

Given these conditions, it is obvious that 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, i.e. the annotations of morphemes should be extensible.

2 Technical realization

In general, the technical realization of the described annotator appears to be feasible, using a very simple, and nevertheless efficient technical solution, i.e. finite state transducers [3, 4]. Specific attention is given to efficiency and adaptability (to variants and dialects, as well as other languages). In the following we describe the algorithmic specification of a morphological parser for the Croatian standard, and synchronic and diachronic variants.

2.1 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. [18, 2, 15], further links to literature and implementations can be found in the context of the OpenFst library [1].

For Croatian there are various descriptions of the formalization and computational modeling of morphology in terms of finite state methods [19, 12]. 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 [17] and Morphisto [21] 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 [11] system.

Common problems of some of the existing tools are the lack of efficient handling of ambiguities in natural languages, and in particular different code-pages for character sets. The common strategy is to replace input symbols in the machine compilation process with a fixed mapping to integer values. This implies that every input has to be transcoded using this mapping before analysis, which represents a performance penalty. Ambiguity is usually dealt with by using non-deterministic finite state models and relying on backtracking, or alternative strategies, which again comes with a performance penalty.

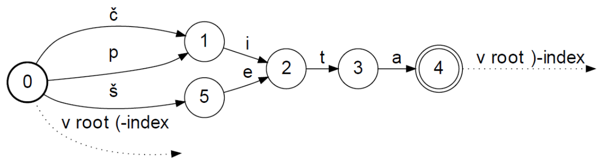
3 FST-based segmentation

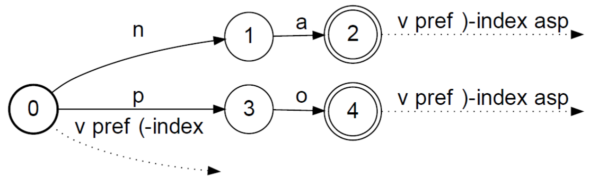
While we decided to stick to the approach and implementation strategy of TAGH, we apply our own experimental libraries and development environment.[1] Following the TAGH-approach [11], we model Croatian morphology by referring exclusively to morphotactic regularities, using morpheme and allomorph sets and regular morphological rules, such that a deterministic finite state transducer (FST) can be generated.

For the compilation of a FST morpheme lists are required. In the initial modeling step morphemes are grouped on the bases of specific criteria. The main criteria are a. morphemes having the same feature specification, and b. being subject to the same morphological rules, where morphological rules are purely distributional and morphotactic, not derivational in the sense of e.g. lexical phonology [16].

Each morpheme group in the Croatian morphological parser represents one deterministic and acyclic finite state transducer (DFST), comparable to the Mealy [13] or Moore machine [5]. 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 with a specific end-bit set. Thus morphemes are marked with a start and end index, as well as the corresponding feature bundle, representing the desired annotation. Morpheme analyses consist of a pair of emission tuples on a stack. Redundancies are avoided by limiting the placement on the stack to one occurrence only. Only complete analyses that span the complete input string are returned in the final output.

The following graph shows a simplified example of an acyclic DFST for verbal roots and for example aspectual prefixes:

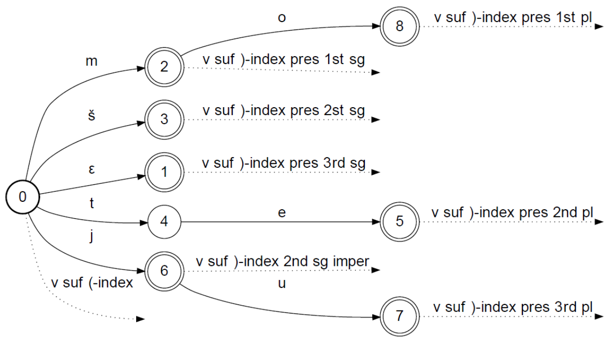




To clarify the semantics of the emission arrows (dotted line), one should keep in mind that the initial emission tuple is placed on the stack only when the initial state is left, while the final emission tuple is placed there when the final state is reached.

All affixes are organized in the same way, e.g. the verbal inflectional paradigm, as in the following graph:

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


o
v suf)-index pres 1st pl
m
v suf)-index pres 1st sg
š
v suf)-index pres 2st sg
ε
v suf)-index pres 3rd sg
t
e
v suf)-index pres 2nd pl
j
v suf)-index 2nd sg imper
u
v suf (-index
v suf)-index pres 3rd pl

Since the model is based on purely morphotactic distributional regularities, potential phonological phenomena are expressed using exclusively allomorphic variations, i.e. alternative sets of allomorphs. Consider for example the allomorphic variation in the case of *banka*(nominative singular of “the bank”) with the dative/locative singular form *banci*. In this case the allomorph *banc* is grouped with all such allomorphs that occur in the context of the instrumental singular suffix *i*, while the root *bank* is grouped with all other nominal roots that have an allomorphic variant in this particular case. This way specific paradigms for such nouns can be defined, that lack the instrumental singular suffix for all such root morphemes, and combine with all the other case specific suffixes, and so on.

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 DFST for the verbal aspectual prefixes, the verbal roots and 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 [20] 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[2] deterministic transducer.

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.

[2]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. Independent of the theoretical question whether such type of recursion indeed exists, or is conceptually necessary, for the analyzer it is empirically irrelevant, and has no impact on its properties, except of size optimization.

The advantage of such a representation is not only that the resulting morphological representation is maximally 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 symbol sequence as input emit a different output tuple), or ambiguity is mapped on the emission of multiple annotation tuples. For Croatian, 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.

3.1 Interoperability and annotation standard

Annotated language data plays an important role in various domains, be it language technology development, or linguistic research. Many different annotations were developed, for various purposes, with different goals in mind. Due to the diversity of encoding and annotation standards, current language resources face a problem related to issues of interoperability and annotation compatibility. The various different tag-sets that are used for different and particular languages 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. Annotation of language data, however, is an expensive task, as well as the change and adaptation of existing data to a new or specific annotation standard.

Thus, the question of annotation standards is crucial for the conceptualization and development of new language resources and language processing tools. In principle, two major options exist. Either a certain annotation standard is promoted and agreed upon, or a specific standard is chosen that maximizes interoperability and compatibility with other existing standards.

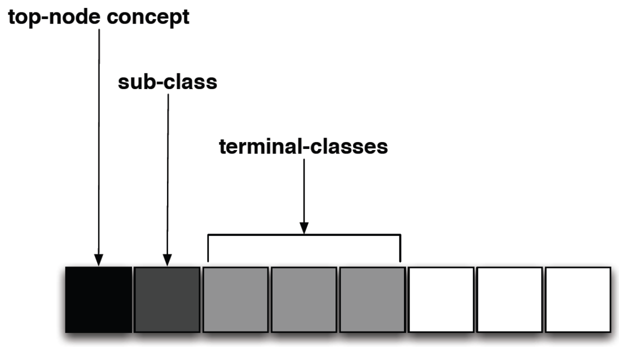
For our purposes here we decided not to promote a specific annotation standard, but rather to offer maximal interoperability in the resulting corpus annotation, as well as in the annotation tool as such, by using a tag-set that appears to be maximally compatible with existing tag-sets, as language specific as necessary, and at the same time maximally extensible. The General Ontology for Linguistic Description (GOLD) [9, 10, 8] *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. The current specification of GOLD is not complete, many concepts are missing, various might change, and need further specification. Nevertheless, the defined development process of GOLD handles current insufficiencies by providing language specific extensions, as well as general extensions with features necessary for the description of a wider language group.

For the purposes here, i.e. morphological and morphosyntactic annotation, the existing definition of GOLD is extended with three additional concepts. All other concepts are covered in GOLD 2008. 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, i.e. morphological properties of morphemes (e.g. prefix, suffix, root), morpho-syntactic properties (e.g. case, number), and morpho-semantic 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. We developed example mappings to specific tag-sets, e.g. alignments to the MULTEXT-East tag-set, with the loss of features that are not defined in MULTEXT-East.

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, and additionally 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. All emitted tags are pre-compiled into the binary representation of the machine. Converting the emission tuples (i.e. individual bit-vectors) into literal string representations is achieved efficiently, once an input string is analyzed completely. This output mapping is optional such that post-processing components can consume the bit-vector representation for subsequent optimal analysis, e.g. in syntactic parsing.

3.2 Implementation

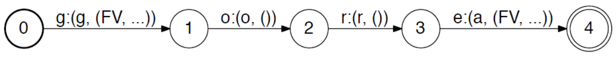
The morphological analyzer consists of two sets of code-bases. The first component converts a lexical base into a formal automaton definition. The second compiles together with the automaton definitions into a binary application.

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. The current implementation 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 [20] 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 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[3]). 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 declarative paradigm of FSTs, and is just temporary. 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, as shown in the graph below. Thus every emission will be a tuple that contains tuples of output characters and optional annotation bit-vectors.



The parser expects a token list as input. The code-page of the lexical base for machine compilation has to match the input tokens. Otherwise there is no restriction on a specific code-page or character encoding, since the automaton processes strings by consuming bytes in the binary representation.

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. Thus, no hypotheses of sub-morphemes are generated, and the number of irrelevant hypotheses is radically reduced.

The significant implementation features that differentiate our implementation from other solutions, are that the code-base is platform independent and open-source, based on free and open tools like GCC and Ragel. Furthermore, the fact that doesn't transcode the lexical base or the input words, it can be based on any encoding. The binary processing strategy allows even for mixed encoding of the lexical base and input tokens, without major consequences for the size and efficiency of the resulting machine.

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.

4 Evaluation

The evaluation version of the implementation for Croatian 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.

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

We selected randomly 10,000 tokens with an average morpheme length of 2.5 morphemes. The parser processes in average approx. 50,000 tokens per second (real 10,000 tokens per 150 millisec.), including runtime instantiation in RAM, 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.

The current implementation doesn't include 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. The described machine is not disambiguating the generated output. For disambiguation the transitional probabilities (and thus the likelihood of a given parse for one lexeme) might be useful. In general we are convinced that disambiguation necessarily has to rely on contextual information, and thus must include some sort of parser or contextual language model, i.e. be part of a more complex analysis component.

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 cannot perform such an evaluation, neither a recall evaluation on a predefined evaluation corpus. Future availability of reference corpora should enable us to provide such extremely relevant evaluation results.

For evaluation and potential application to other languages, the source code is made available on the web site `http://personal.unizd.hr/~dcavar/CroMo/`.

References

[1] Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri. 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, 2007.

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

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

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

[5] Paul E. Black. Dictionary of algorithms and data structures. Online publication: U.S. National Institute of Standards and Technology, Available online, December 2004.

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

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

[8] Scott O. Farrar. *An Ontology for Linguistics on the Semantic Web*. PhD thesis, The University of Arizona, Tucson, Arizona, 2003.

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

[10] Scott O. Farrar, William D. Lewis, and D. Terence Langendoen. 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, 2002.

[11] Alexander Geyken and Thomas Hanneforth. 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 2005.

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

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

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

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

[16] Jerzy J. Rubach. *Cyclic and Lexical Phonology. The Structure of Polish*. Foris Publications, Dordrecht, 1984.

[17] Helmut Schmid, Arne Fitschen, and Ulrich Heid. 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, 2004.

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

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

[20] Adrian D. Thurston. 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 2006.

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